

Real Time Systems
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 18
Rate Monotonic Schedulability Analysis

Welcome to this lecture. In the last lecture, we were discussing about the rate monotonic scheduler, a very popular scheduler for real time and embedded applications and we were trying to address a very important design problem that is given an application, can this, the task set in the application feasibly run on a universal that is they meet all the deadlines, if the priorities of the tasks are assigned according to the rate monotonic principle.

And we had looked at the necessary condition and the sufficient condition which is the Liu Layland condition and we said that the Liu Layland condition is a bit conservative, if it passes the Liu Layland condition okay no worry the task set will can be run satisfactorily on a uniprocessor. But if it fails, then we need to look further. Specifically, we need to apply the completion time theorem to check that whether really it cannot be run or is there a chance that they are unsatisfactorily on a uniprocessor. So let us proceed from that point.

(Refer Slide Time: 1:39)

RM Analysis -- Example 4

Task set: $T_i = (e_i, p_i)$

① $T_1 = (2,4)$ and $T_2 = (4,8)$ ②

Schedulability check:

$2/4 + 4/8 = 0.5 + 0.5 = 1.0 > 2(\sqrt{2} - 1) = 0.82$ Not Schedulable

Let us try to check schedulability manually...

0 2 3 4 6 8

T_1^1 T_2^1 T_1^2 T_2^1 ...

Some task sets that FAIL the Liu-Layland test may be schedulable under RMS

Let us look at an example. We have a task set consisting of two tasks. The first task takes 2 units of execution time every 4 unit, the second task requires 4 unit of time every 8 units. Now is this task can be feasibly scheduled on a uniprocessor. First, we can try the Liu Layland condition, for

that we need to compute the utilization. Now the utilization due to the two tasks is 100 %. But then the bound given by the Liu Layland is 0.82 and therefore the, according to Liu Layland condition this cannot turn feasibly on a uniprocessor.

But let us check further. Let us try to draw the schedule according to a rate monotonic scheduler and see that whether the task set runs feasibly and all the deadlines are met. So, the task 1 is the highest priority because the period is 4, task 2 has a higher period and therefore its priority is 2 and a time 0 both tasks arrive or in other words their phasing is 0 and initially T₁ runs for 2 units of time and at that point T₂ can run for 2 units of time and T₁ again arrives at 4 and T₁ starts running for 2 units of time and T₂ runs.

So by 8 unit T₂ completes T₁ of course it has completed all its, it has met all its deadline and even T₂ has met its deadline, we can draw the schedule as long as we can. And we will see that all the schedules are met. So can we formalize this and find out whether a task set when we check manually it is schedulable and Liu Layland result says that it is not schedulable. So that we will give in terms of the completion time theorem. So some tasks that failed the Liu Layland test might actually be schedulable under a rate monotonic scheduler on a uniprocessor.

(Refer Slide Time: 4:42)

RMA Schedulability

- Utilization bound 1: $U = \sum_{i=1}^n \frac{e_i}{p_i} = \sum u_i \leq 1$
- Utilization bound 2: $U_2 = \sum u_i \leq n(2^{\frac{1}{n}} - 1)$ (Liu-Layland)
- Schedulability check 3: Liu and Lehoczky's Completion Time Theorem

The slide includes a diagram of a task set utilization space with a green line representing the utilization bound. A red checkmark is next to the first bound, and a red 'X' is next to the second bound. A red circle with '??' is drawn around the second bound. A video inset shows a man speaking.

If we look at the schedulability results so far, the utilization bound 1 it said that the utilization due to tasks ≤ 1 and utilization bound 2 which is the Liu Layland criterion, it provided a bound through an expression consisting of two terms. So, the task set as long as its utilization lies

between U_2 and U_1 , ok as long as it exceeds U_1 it is not schedulable and if it is below U_2 it is schedulable, we have shown that using green, but between U_2 and U_1 that is the question mark.

And that we will check using the Liu and Lehoczky's completion time theorem. If the bound, if the actual utilization lies between U_2 and U_1 , if it is more than U_1 we can say that it is not schedulable but if it exceeds U_2 , but less than U_1 , we need to check further using the Liu Lehoczky's completion time theorem.

(Refer Slide Time: 6:19)

RM Schedulability Test- III

- Completion time theorem (Liu and Lehoczky 1989):
 - 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.

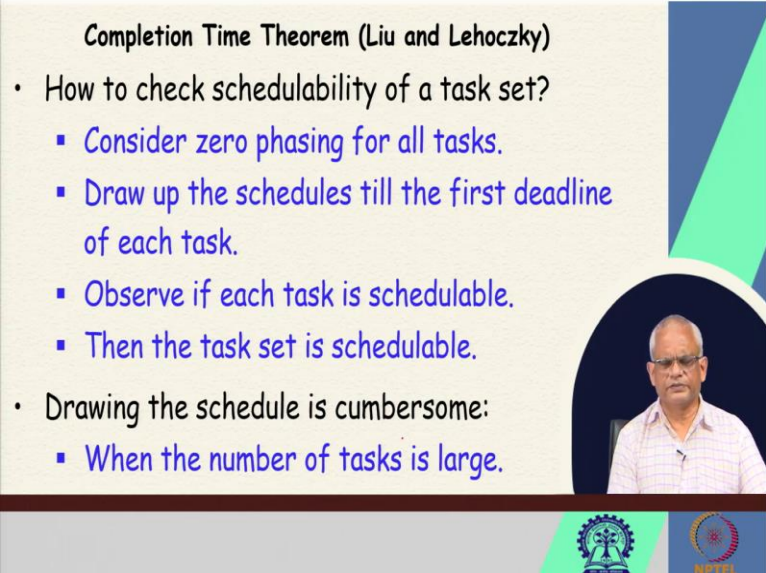
Now, let us try to understand the completion time theorem which was given by Liu Lehoczky in 1989. The theorem states that if we consider all the tasks under zero phasing, so that means the tasks might be arriving, the first instance of task might arrive at any phase. Let us convert all tasks to zero phasing, assume that all tasks have zero phasing.

And under that, if we check the each task individually, and check that they meet their first deadline, we do not have to look further, we just look till the first deadline under zero phasing, then we can confidently say that if the first deadline is met, then the task will keep meeting all its deadline and what is the basis of this theorem, which confidently says that, if the task set meets its deadline, first deadline, just need to check one deadline under zero phasing then it will keep meeting all its deadline.

What is the basis of this statement? The main basis here is that when the tasks, all tasks arrive at zero phasing that is all arrived exactly at zero time. And that is the worst case for all the tasks, any task that needs to miss its deadline will miss here, in their first deadline when all the tasks start at the same time. So, if we can check the worst case and see that all the tasks are meeting their deadlines, then we can confidently say that the tasks will meet any of their deadlines.

So that is the crux here. Let me just repeat that once again. That zero phasing gives the worst case execution time or completion time for all tasks, the zero phasing, under zero phasing every task will have its worst case completion time. And if all tasks meet their deadlines under the worst case completion time, then definitely the task set is schedulable.

(Refer Slide Time: 9:07)



Completion Time Theorem (Liu and Lehoczky)

- How to check schedulability of a task set?
 - Consider zero phasing for all tasks.
 - Draw up the schedules till the first deadline of each task.
 - Observe if each task is schedulable.
 - Then the task set is schedulable.
- Drawing the schedule is cumbersome:
 - When the number of tasks is large.

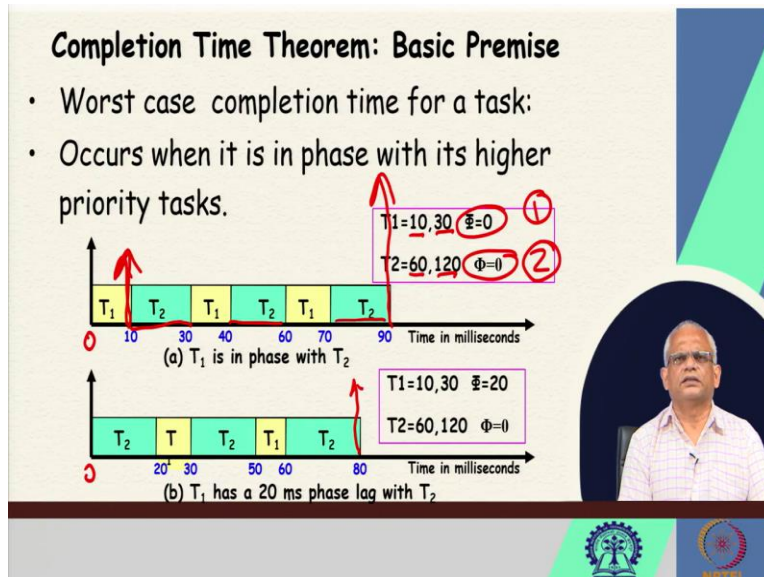
The slide features a video inset of a man in a light-colored shirt speaking. At the bottom, there are two logos: one of a tree and another of a gear.

So, to apply this, one possibility is that we consider zero phasing for all tasks, even if some phasing is given for the tasks like 100 or 500 for different tasks, we just ignore that we assume that all tasks arrive at zero, then we just draw the schedules for first deadline using the rate monotonic principle, and then see if all the first deadlines are met, then we will say that the task set is schedulable otherwise not.

But one thing is that if there are a dozen task or two dozen tasks and we are trying to draw the schedule manually for every task not only there is a chance of error to apply the rate monotonic principle, but it is also cumbersome. Can we have a mathematical expression which you can use

and check that whether they satisfy the completion time theorem, so, that we will just try to discuss the mathematical expression for completion time theorems?

(Refer Slide Time: 10:32)



But one thing is the basic premise of the completion time theorem based on which it is founded is that when all the tasks in application arrive at the same time, let us assume at zero time, then the worst case completion time for all tasks results, there is the main idea here. Main observation actually, that when the tasks arrive exactly at the same time, then the tasks will have their worst case completion time, we will just take an example to show that.

Assume that we have a task set, two tasks T_1 has execution time 10 and a period of 30, T_2 execution time 60 and period 120. Now, the phasing of T_2 is 0, so this starts exactly from 0 time. And this also starts exactly at 0 time. So, this is zero phasing. Now, under zero phasing, we can draw manually the schedule and find that T_2 need 60 units. So, it got here 20 20 and 20. So, by 90 T_2 could complete and T_1 of course, it completed at 10 because it is the highest priority.

So, the priority 1, T_2 is priority 2. So T_1 can complete at 10. But if we draw the schedule, you can try that yourself and see that T_2 completes at 90. Now, let us assume that they do not arrive at the same time, but there is a phase difference T_2 has zero phasing. So, it arrives at 0 and T_1 arrives at 20. Now, we can draw here and check that T_2 completes at 80 and T_1 of course completes within 10 of its arrival time.

So, the completion time of T1 still it is 10 but T2 instead of 90 became 80 because they came in different phases, if they come exactly at the same time, their arrival time is exactly on the same time for all tasks, then that is the worst case completion time of the different tasks. So with this example, we are just trying to illustrate that the worst case completion time for a task occurs when it is in phase with its higher priority tasks.

The highest priority task will not be affected even if it is in phase with other tasks because it will anyway run whether it is in phase or not, it will anyway have the highest priority run. So, a lower priority task would have a larger completion time if it is in phase with its higher priority task compared to when it is out of phase with its higher priority task. So that is the basic premise. And therefore it says that check the first deadline under zero phasing and if the task set meet their respective first deadlines, then the task set is feasibly scheduled.

(Refer Slide Time: 14:30)

Liu and Lehoczky's Criterion : Mathematical Formulation

- A task T_i will meet its first deadline if.

$$e_i + \sum_{j=1}^{i-1} \left\lceil \frac{p_i}{p_j} \right\rceil * e_j \leq p_i$$

p_i is the period of task T_i

$Pr(T_1) > Pr(T_2) \dots > Pr(T_n)$

$\lceil \frac{25}{10} \rceil = 3$ $T_2 = 4, 25$

But can we give a mathematical formulation for the Liu Lehoczky's completion time criterion; the criterion is that for a task T_i , we can check its first deadline. It will meet its first deadline. If e_i is the execution time of the task, and for all its higher priority tasks, which are priority 1 to j , $j = i - 1$ we have arranged the tasks such that T_i the higher priority tasks are T_1 to T_{i-1} . Now, we need to check with respect to all the higher priority tasks, we need to compute this term.

So, the execution time of the task T_i is e_i and for each of these tasks, we need to check p_i / p_j where j is between 1 to $i - 1$ and we take a $\lceil \cdot \rceil$ of that, $\lceil p_i / p_j \rceil * e_j$. So, what it means really is

that how many times the higher priority task arrives within the period of p_i . Let us take an example. If we have two tasks, let us say one task is takes 2 units of execution time every 10 units this is T_1 and T_2 let us say takes 4 units of execution time every 25 units.

Now, whenever T_1 arrives, it will preempt T_2 because T_1 is higher priority and T_2 is lower priority but then how many times will T_1 arrive or how many times will T_1 preempt, when T_2 is ready, the T_2 is 25 unit once it arrives here and its deadline is 25 or the period is 25 both period and deadline are 25. So, this is T_2 and T_1 can arrive at most three times because it is period is 10. So, if it arrives here, it can arrive here at 20 sorry, 0 and then 10 and then 20 or if it arrives at 1, 11 and 21.



So, at most it can arrive three times and how do we get three times, three times is $\lceil 25/10 \rceil = 3$. So, like that we compute what is the maximum number of times that a higher priority task can arrive before a lower priority task completes and that is given by $\lceil p_i / p_j \rceil * e_j$ because each time a high priority task arrives it will execute for e_j , we will take some examples and see how to use this result.

But this is a crucial result. If you are developing a real time application using a rate monotonic scheduler, you are very likely to use this result. Now let us therefore look at it with some examples and see how to apply this result.

(Refer Slide Time: 18:38)



Example 5

- Consider three periodic tasks
 - T1: $e_1=20\text{mSec}$, $p_1=100\text{mSec}$
 - T2: $e_2=30\text{mSec}$, $p_2=150\text{mSec}$
 - T3: $e_3=60\text{mSec}$, $p_3=200\text{mSec}$
- Check whether the task set is RMA schedulable.



Solution

- Checking for Liu-Layland criterion:
$$\sum_{i=1}^n \frac{e_i}{p_i} = \sum u_i = \frac{20}{100} + \frac{30}{150} + \frac{60}{200} = \frac{420}{600} = 0.7 < 0.78$$
- Liu-Layland criterion is satisfied:
 - Therefore, the task set is schedulable.



First is, let us take one example application, execution time is 20 and period is 100, task 2 execution time is 30, period 150 and task 3, 60 and 200. And now we check whether the task set is the rate monotonic analysis. We can do the analysis and be sure whether it can be feasibly run on a uniprocessor and given an application like this, the first thing is we compute the utilization and then check the utilization bound 1 which is it is less than 100 if it exceeds 100 we do not have to look further, it is not schedulable.

And if it is less than 1 we check the Liu Layland criterion and if it meets Liu Layland criterion, nothing to look forward it will run, I mean nothing to check further it will learn and if it fails the

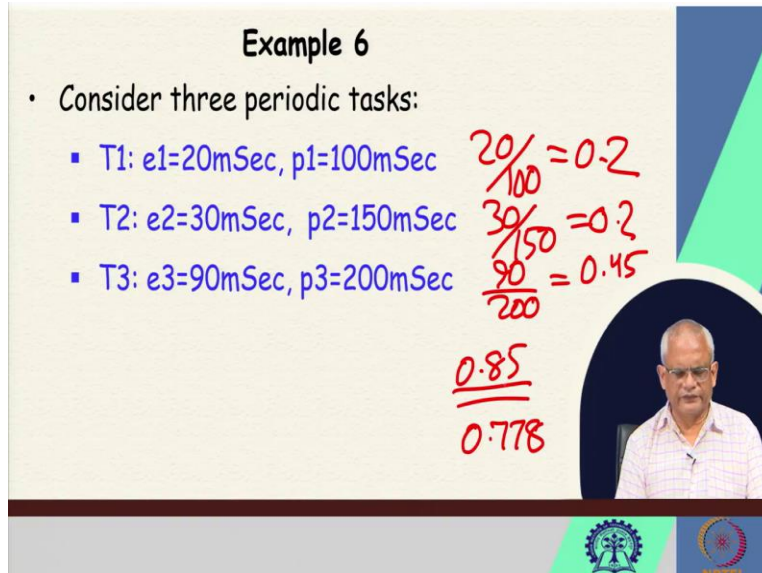
Liu Layland criterion then only we use the completion time theorem. So we first compute the utilization due to the three tasks and it turns out to be 0.7 which is less than 0.78 and therefore the task that passes Liu Layland criterion, do not have to look further.

(Refer Slide Time: 20:13)

Example 6

- Consider three periodic tasks:
 - T1: $e_1=20\text{mSec}$, $p_1=100\text{mSec}$ $\frac{20}{100} = 0.2$
 - T2: $e_2=30\text{mSec}$, $p_2=150\text{mSec}$ $\frac{30}{150} = 0.2$
 - T3: $e_3=90\text{mSec}$, $p_3=200\text{mSec}$ $\frac{90}{200} = 0.45$

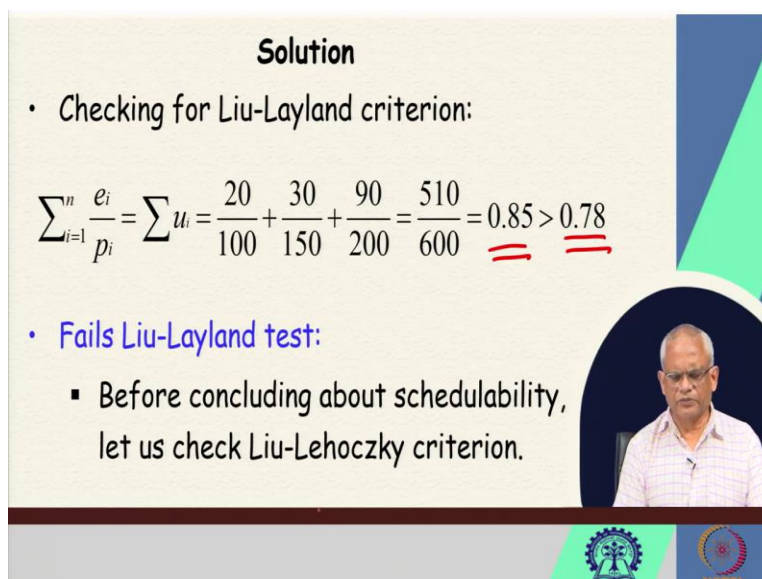
$\frac{0.85}{0.778}$



Solution

- Checking for Liu-Layland criterion:

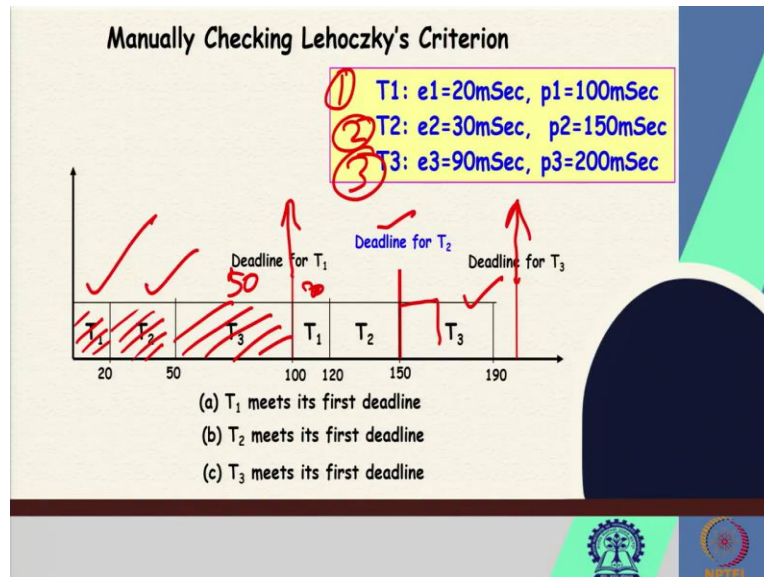
$$\sum_{i=1}^n \frac{e_i}{p_i} = \sum u_i = \frac{20}{100} + \frac{30}{150} + \frac{90}{200} = \frac{510}{600} = 0.85 > 0.78$$
- Fails Liu-Layland test:
 - Before concluding about schedulability, let us check Liu-Lehoczky criterion.



But let us look at this example. Again we have three tasks, 20 is execution time, 100 are the period and deadline, 30 and 150, 90 and 200. Now, let us try to compute the utilization and apply the Liu Layland criterion. So, this is $20 / 100 = 0.2$, $30 / 150 = 0.2$ and $90 / 200 = 0.45$. Now, if we sum it up, it will become 0.85 and three tasks we know that the Liu Layland bound is 0.778 and it is exceeding 0.778, 0.85.

So, 0.85 is the utilization due to the tasks and which is much greater than the Liu Leyland bound of 0.778 and therefore, according to the Liu Leyland criterion, this task set cannot be feasibly scheduled on a uniprocessor using a rate monotonic scheduler. Now, let us try the completion time theorem. Before we say that this task set cannot be physically run because it fails Liu Layland test, let us try to use the completion time theorem at the Liu Lehoczky's criterion.

(Refer Slide Time: 22:15)



One is about manually checking the criterion where we assume zero phasing and check whether they meet their first deadline here T_1 is priority 1, highest priority, T_2 is the second highest priority and T_3 is the third highest priority, initially T_1 will run because it is the highest priority it will run for 20 milliseconds and then T_2 will run once T_1 completes, it will run for 30 milliseconds and T_3 needs 90 millisecond but then at 100 T_1 will again arrive.

So T_3 can run here for 50 millisecond but T_1 again arrives and it runs for 20 and the T_2 okay, T_2 arrives at 150 okay, so, just a small problem here, but we can check here that T_3 will run here and then T_2 arrives. So, T_2 has met its deadline, the T_2 's T_1 's deadline was here it has completed much before the T_1 deadline is met, T_2 's deadline is also met and we can see that the T_3 's deadline is also met. So we can draw the schedule properly. T_2 actually arrives at 150.

So here T_2 will run and we will check that T_1 T_2 T_3 all meet their respective deadlines. Now can we check it with the expression.

(Refer Slide Time: 24:34)



Mathematically Checking for Lehoczky's Criterion

- For T_1 : $20 < 100$ ✓
 - Satisfied
- For T_2 : $30 + 20 * 2 = 70 < 150$ ✓
 - Satisfied
- For T_3 : $90 + 20 * 2 + 30 * 2 = 190 < 200$ ✓
 - Satisfied

$T_1: e_1=20\text{mSec}, p_1=100\text{mSec}$
 $T_2: e_2=30\text{mSec}, p_2=150\text{mSec}$
 $T_3: e_3=90\text{mSec}, p_3=200\text{mSec}$

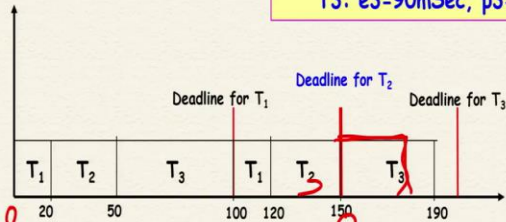
$\lceil \frac{150}{100} \rceil = 2$
 $\lceil \frac{200}{100} \rceil = 2$

$\frac{90}{90} \quad \frac{20}{10} \quad \frac{30}{60}$





Manually Checking Lehoczky's Criterion

$T_1: e_1=20\text{mSec}, p_1=100\text{mSec}$
 $T_2: e_2=30\text{mSec}, p_2=150\text{mSec}$
 $T_3: e_3=90\text{mSec}, p_3=200\text{mSec}$





(a) T_1 meets its first deadline
(b) T_2 meets its first deadline
(c) T_3 meets its first deadline



Mathematically Checking for Lehoczky's Criterion

- For T1: $20 < 100$
 - Satisfied
- For T2: $30 + 20 * 2 = 70 < 150$
 - Satisfied
- For T3: $90 + 20 * 2 + 30 * 2 = 190 < 200$
 - Satisfied
- The task set is schedulable. ✓

T1: $e_1=20\text{mSec}$, $p_1=100\text{mSec}$
 T2: $e_2=30\text{mSec}$, $p_2=150\text{mSec}$
 T3: $e_3=90\text{mSec}$, $p_3=200\text{mSec}$

Now for the first task, there is no higher priority task. So $20 < 100$. So trivially the first task will meet its deadline. The second task is its execution time is 30 and $\lceil 150/100 \rceil = 2$ and the higher priority task is T₁. So $30 + 20 * 2$ which is $70 < 150$. So T₂ will also meet its deadline and for T₃ there are two higher priority tasks. So, its period is 200 and $\lceil 200/100 \rceil = 2$ and $20 * 2$ that much will be required for the first highest priority task and the second priority task that will be $\lceil 200/150 \rceil = 2$.

So 30 for the second priority task $30 * 2 = 60$. And the task itself requires 90. So, $90 + 40 + 60 = 190$. So, it meets its deadline, we can confidently say, but for the manual drawing just see that even after carefully drawing, we had drawn, made a mistake here actually that T₂ arrives at 150. So, once it arrives at 0 and the next time it will arrive at 150. So, T₂ will run here and T₃ will run here.

So, just see that the manual drawing is erroneous, cumbersome, but we can apply the formula and we can easily show that the task set whether it meets its completion time theorem or not. And for this case, it fails the Liu Layland test, but it passes the Liu Lehoczky's criterion or the completion time criterion and therefore, the task set is schedulable. We are at the end of this lecture. We will discuss about some other issues, important issues with the rate monotonic scheduler that is in the next lecture. We will stop here. Thank you.