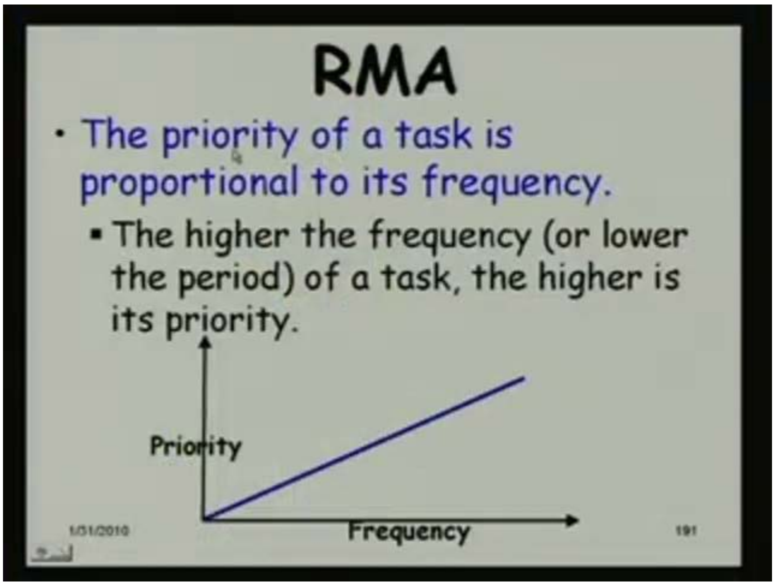
Real-Time Systems

Prof. Dr. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module No. # 01
Lecture No. # 10
RMA Scheduling: Further Issues

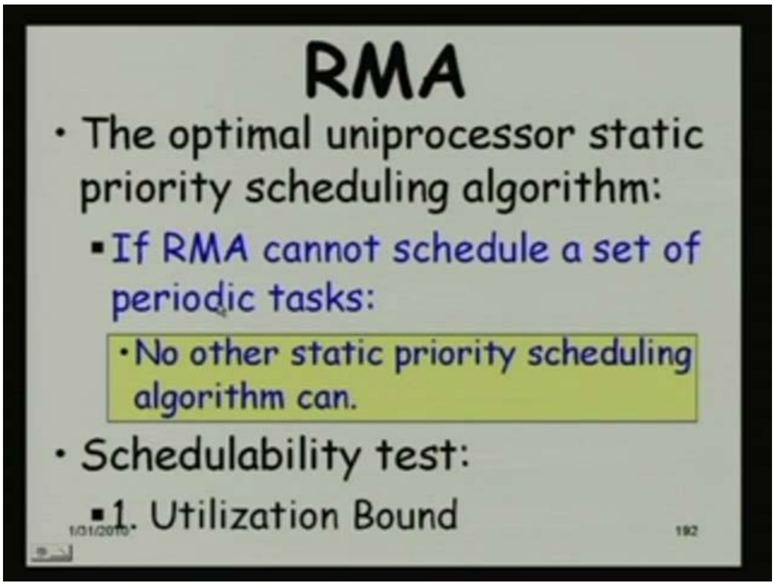
Good morning, let us get started, we will first recapitulate; what we are doing last time. We are saying that, rate monotonic scheduling is a very popular scheduling algorithm used in almost all non-trivial applications and we had discussed some very basic issues in rate monotonic scheduling. So, we will just review that very briefly for a couple of minutes and then we will proceed from there on. So, just recollect that we are saying that, the algorithm for this is very simple. Said that,

(Refer Slide Time: 00:53)



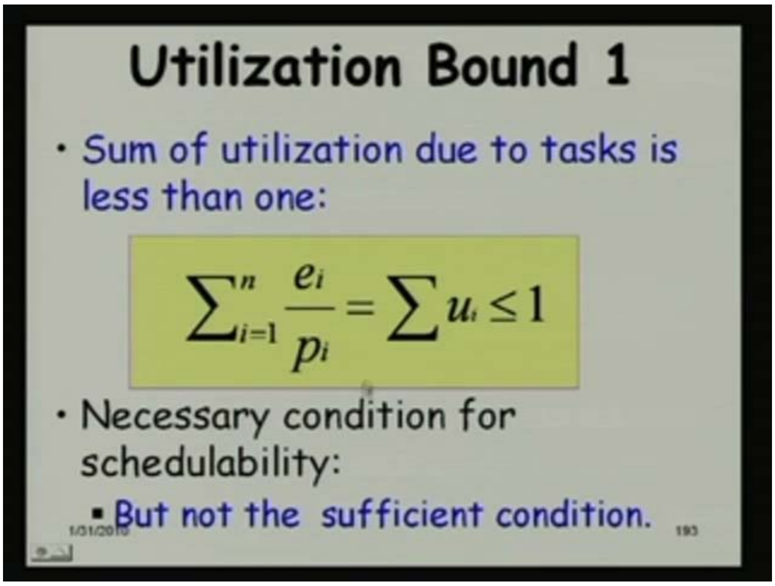
The designer has to assign priority to tasks. That is proportional to the frequency of the tasks. The task having the highest frequency will have the highest priority right. So or in other words, the task having the lowest period will have the highest priority and the task having highest the period will have the lowest priority.

(Refer Slide Time: 01:22)



And then, we were discussing about the property of rate monotonic algorithm. We said that, it is a static optimal, static priority algorithm. Here, the priorities are assigned before the system starts by the programmer during design time and throughout the system run the priority does not run sorry does not change and it is the optimal algorithm in the sense that; if something cannot be scheduled using RMA then, we can we cannot have any other static priority scheduling algorithm which can schedule that same set of tasks. So, in that way this algorithm is very important and we were trying to determine, how it know the schedulability of a set of tasks that are given to us.

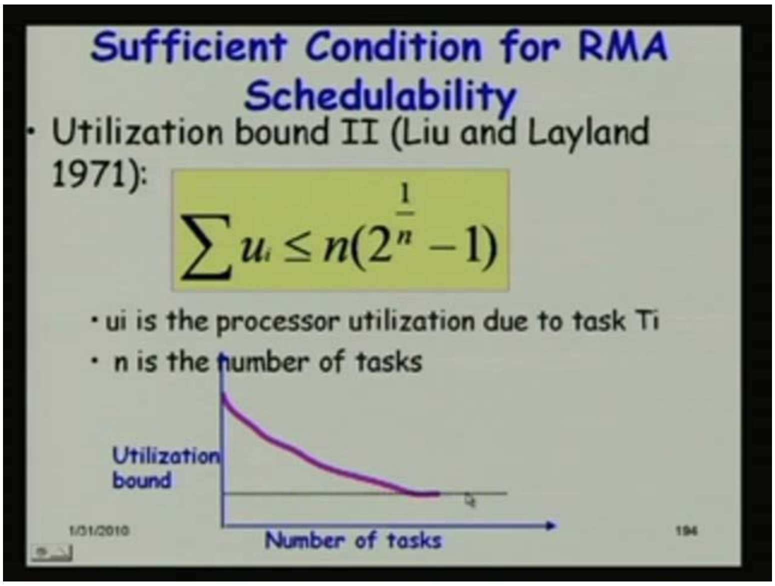
(Refer Slide Time: 02:29)



So, the first one, the first test that we need to do is the utilization bound. Basically the processor cannot be used more than 100 percent very simple condition here. That the utilization due to each task that is the time of the execution time it needs divided by the period of the task; summation of that should be less than 1 or sum of the utilization should be less than 1. This is the basic condition to be satisfied by any scheduler on uniprocessor.

But for EDF, this is both necessary and sufficient condition, but here for RMA; this is a necessary condition. If it is less than 1 then there is a chance that it will be schedulable, but it is not the sufficient condition we will have to test further.

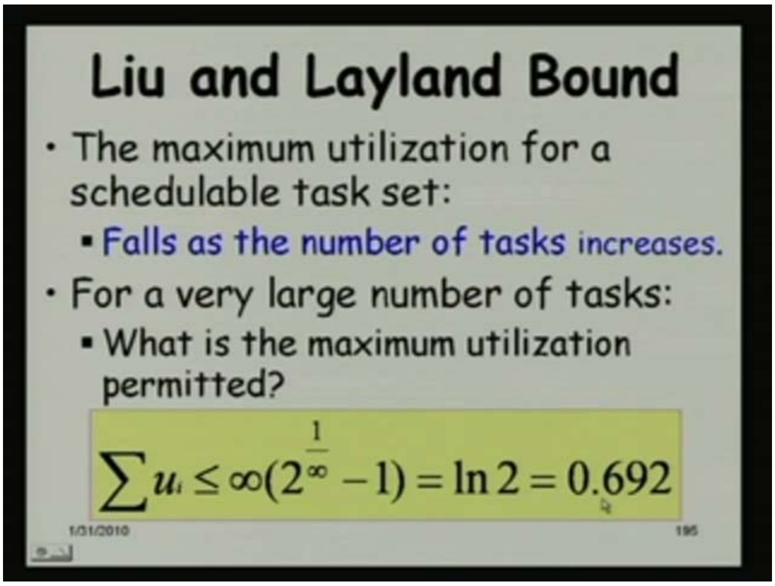
(Refer Slide Time: 03:14)



And we had discussed about, the liu and layland criterion proposed by liu and layland in 1971. Through a lengthy derivation, complicated derivation said that if the utilization due to all the tasks in the set is less than n into 2 to the power 1 by n, minus 1. Where n is the number of tasks that we are trying to schedule. If this is satisfied then the set of tasks will surely be schedulable.

Now, if we try to plot the total utilization due to the tasks that is achievable or the maximum utilization for a set of tasks that is schedulable under RMA, according to the liulayland criterion. We can substitute values for n is equal to 1, 2, 3 etcetera and try to determine this utilization bound. So, if you do that we will see that for number of tasks one utilization is 100 percent and then gradually drops as the number of tasks increases and then, saturates around 0.69.

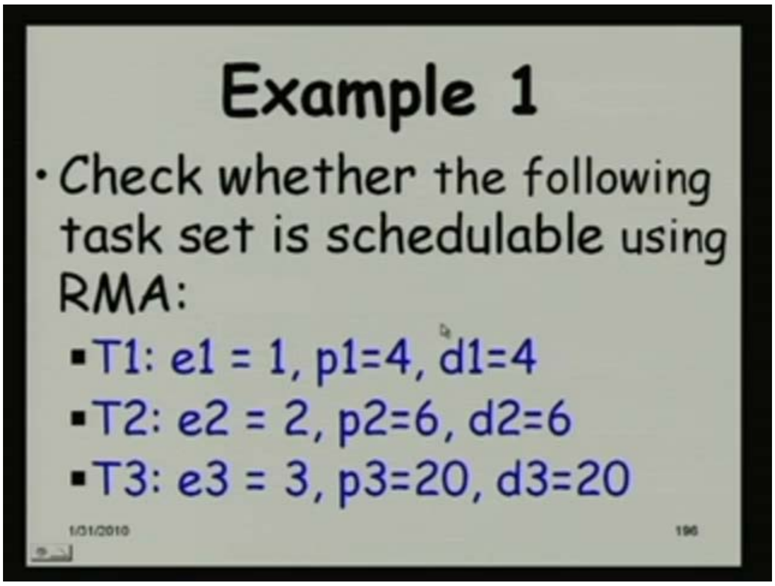
(Refer Slide Time: 04:37)



So, when n becomes very large it becomes indeterminate form infinity into 2 to the power 1 by infinity, minus 1 and by using l hospital's rule we can find the bound to be 0.69.

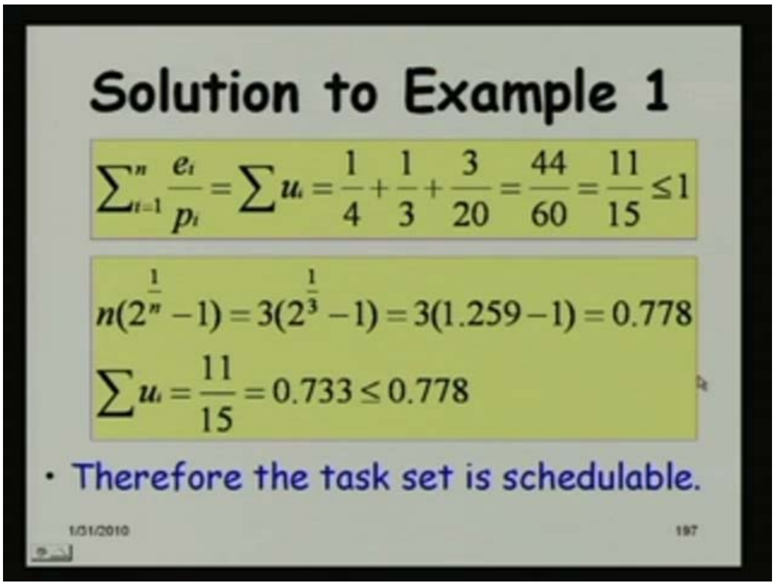
So, that means, if we have large number of tasks the maximum utilization of the C P U is about 0.7. We cannot make the C P U any more busier than this. So, 30 percent of the C P U time will be wasted is not it? But that is the price for meeting the deadlines for the tasks, according to the liu layland criterion.

(Refer Slide Time: 05:25)



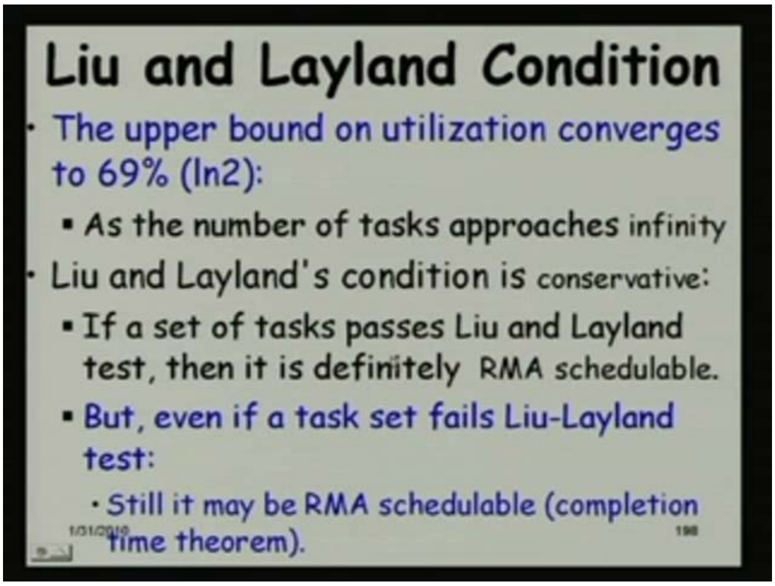
Then, we had looked at some problems, like given a set of tasks how do we determine whether they are schedulable.

(Refer Slide Time: 05:32)



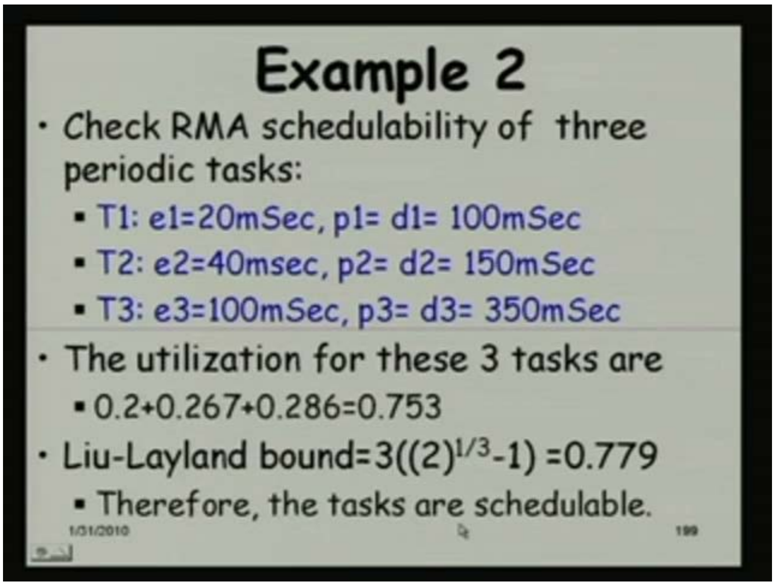
So, we just need to find the utilization due to the set of tasks. So, we find that it is 11 by 15 some of you have done last time and then, check n into 2 to the power 1 by n minus 1. Which is 0.778 and since, 11 by 15 is less than 0.778 the task said, will be schedulable. Simple test instead of put in the formula you get whether it is schedulable or not.

(Refer slide Time: 06:02)

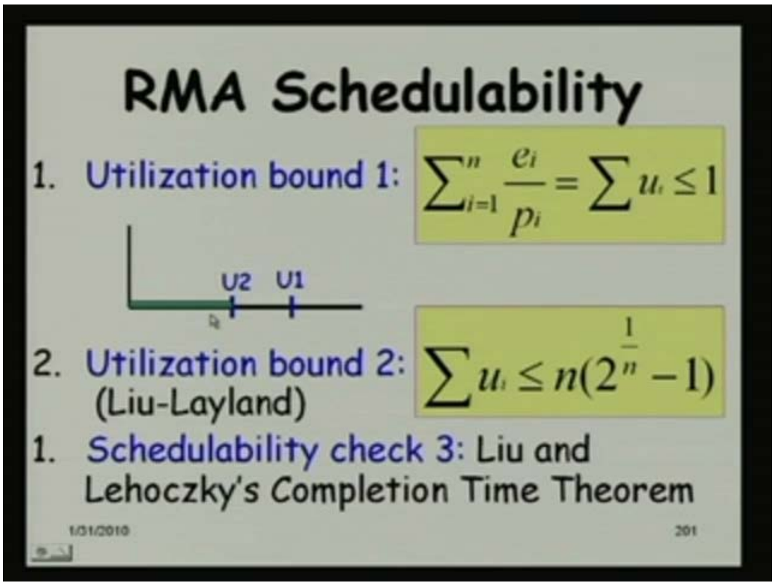


But the thing is that if a set of tasks, fails the liu layland criterion is there a chance that they will be schedulable yes it is. So, because the liu and layland criterion is a pessimistic criterion if it passes, liu layland test then, definitely it will be schedulable, but even if it fails, there is chance that still it will be RMA schedulable and this was given by the completion time theorem by liu and lehoczky.

(Refer Slide Time: 06:39)



(Refer Slide Time: 06:43)



So, today I will discuss about, the liu lehoczky’s criterion and we will proceed from there. So, the first utilization bound is the processor, total processor utilization due to the set of tasks should be less than 1 and then the liu layland criterion, but it is pessimistic if it fails, we need to check further. So, that is given by the completion time theorem it is called as the completion time theorem proposed by liu and lehoczky.

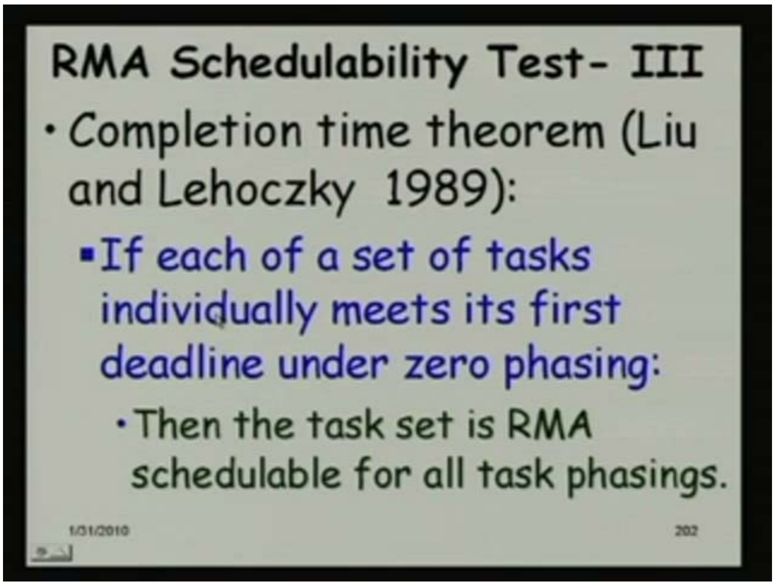
So, this diagram says that, u 1 utilization bound 1 if it is less than u 1 then, there is a chance that the set of tasks, will be schedulable if it is more than u 1 then; obviously, the set of tasks will not be schedulable and if it is less than u 2 that is the utilization bound 2 obtained by this n into 2 to the power 1 n, minus 1 then definitely the task set will be schedulable as long as it is less than u 2 we can positively say that, it will run satisfactorily.

But what if it exceeds u 2 and less than u 1. So, there is a chance that it will be schedulable and that is given by the liu and lehoczky's completion time theorem. We will look at it this theorem is very simple. Actually, unlike the liu and layland which we said, very complex proof here, its straight forward. So, as long as it is less than u 2 less than the liu layland criterion. We can without hesitation say that, it will be schedulable more than u one.

Is it set three perfect bound (( )) lowest case.

The lowest case yeah we will we will discuss about, that the question is that whether the completion time theorem tells us precisely whether it will be schedulable or not to large extent yes, but again we will see that even if it does not really satisfy liu lehoczky criterion that is the third test still there is a small chance we will we will check that let us first discuss about the liu lehoczky's criterion.

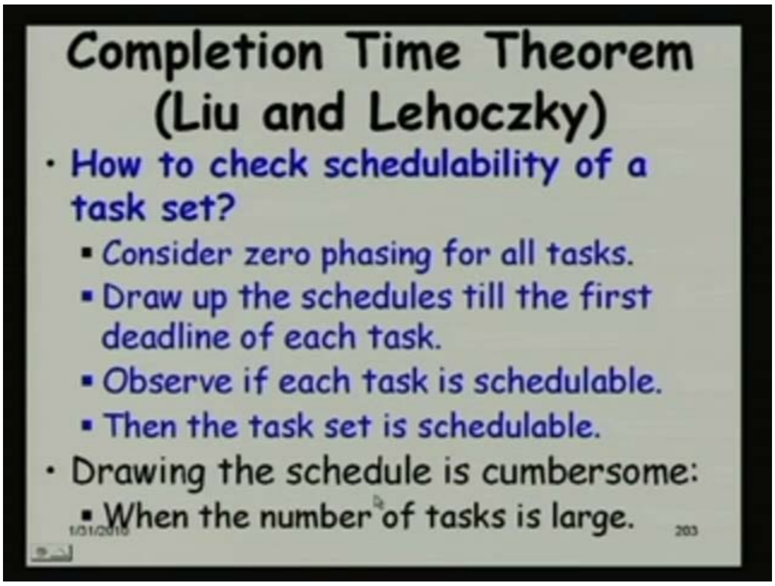
(Refer Slide Time: 09:13)



So, this was proposed in 1989 by liu and lehoczky and the theorem is very simple. The theorem says that looks at the deadlines of every task and assumes that the tasks are 0 phasing. So, even if some phasing information is given for the tasks assume that all tasks are 0 phasing and then check the first deadline of every task.

If every task meets its first deadline under 0 phasing then the task set will be schedulable. So, consider 0 phasing and check whether each task will meet its first deadline, do not have to look any further as long as it meets the first deadline, the task set will be schedulable, but what is the reasoning behind this theorem.

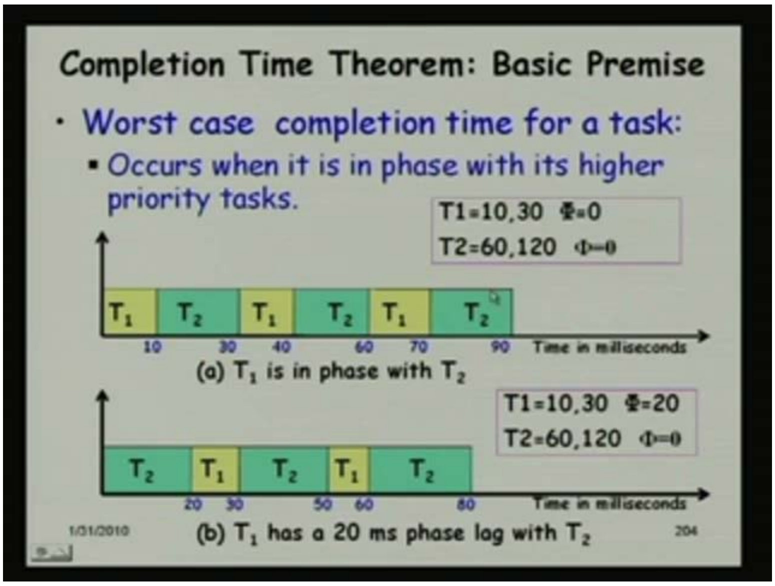
(Refer Slide Time: 10:10)



We will just look at the reasoning behind this theorem, sort while from now, but let us see using this theorem, how do I check the schedulability of a task set. So, first is all the phase information we need to make 0 and then for each 1 we will have to draw the schedule till the first deadline, and then check if the task set is schedulable and if all the task is schedulable I mean meet their first deadline, the task set will be schedulable.

But, the problem with this approach is that if you have let us say 10 tasks and each one has you know very different periods, drawing the first schedule for all of them is difficult. We should have some simpler test we will we will look at that, but for 2 3 tasks definitely you can drop the schedule and compute.

(Refer Slide Time: 11:10)



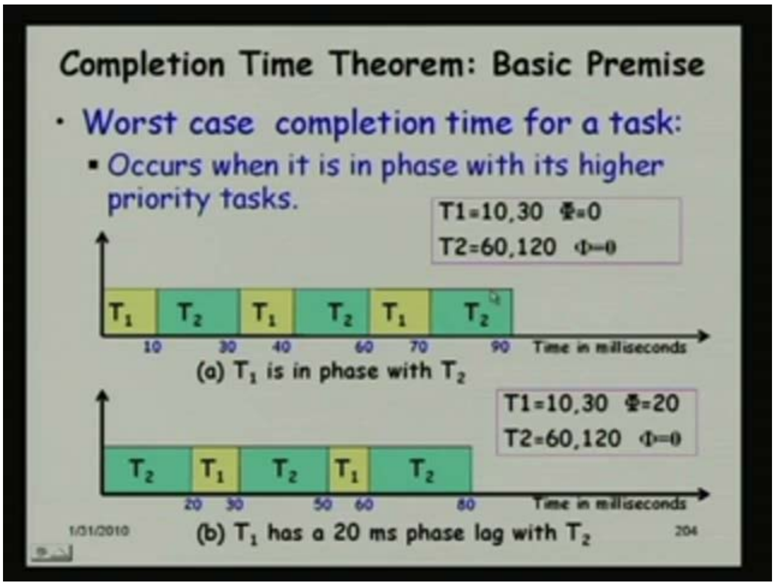
So, the basic premise or the main idea behind this theorem is that for any set of tasks, the worst case schedulability or the most likeliness of the missing the schedule occurs when the task set is in phase with each other. So, if a lower priority task is out of phase with a higher priority task then, it is likely to get more secure time, but if it is in phase with a higher priority task then, chances are that it will not meet its deadline. So, if we can check for the worst case situation that is all tasks are in phase then, we can say that, they will meet under all phasing.

Now, let us just to explain this idea that when the tasks are in phase the worst case occurs let us just take one example. We will take 2 tasks T1 has execution of time 10 and period of 30 deadline of 30 and let us say phase of 0. And another task T2 execution time of 60 and period of 100, 20 millisecond and phase of 0 now, which will have the highest higher priority between T1 and T2 yes T1 will have higher priority, because it has lower period or higher frequency.

So, let us try to draw the time by which T2 will complete that is called as the response time of T 2. So, T1 is in phase with T 2, because phi is equal to 0 we will also look at the same set of tasks when the phase differs, and for both of these cases, we will try to find out what is the completion time of T2 and if the deadline is aggressively set then, this is the situation we will see that the deadline is likely to be missed.

So, the first one let us draw the schedule see here, T1 is the higher priority runs for 10 and then, T1 continues to run until, T1 arises again and T1 arises at what time that is 30 right at 30 T1 arises, and T2 will be preempt. It is not it and T1 continues until, 10 milliseconds that is 40 and again as soon as the C P U becomes, available T2 runs and if we draw the schedule like this we will see that T2 will complete at 90 millisecond right it will have the 60 millisecond of execution at this point 90. So, that is the response time for T 2.

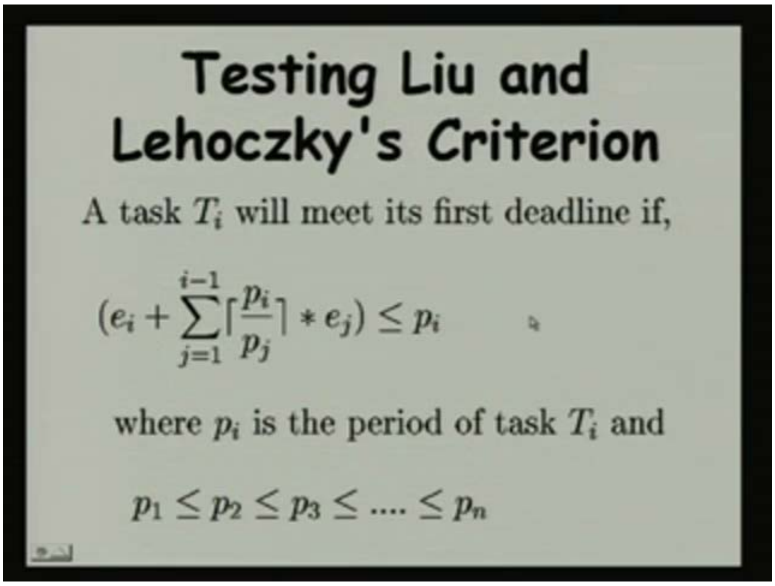
(Refer Slide Time: 11:10)



But, let us assume that T2 is out of phase with T1 that is T1 is arising after 20 millisecond the first instance of T1 arises at after 20 millisecond. So, first it will run and as soon as T1 arises it will run and. So, on and we will see that the response time is 80.

So, just because they are out of phase T2 could complete little here, we can also try to check whether what if T1 has 0 phasing, and T2 has phasing of 20 or 30. In each case we will see that, when they are in phase the response time is the longest. So, if we can show that, in the worst case that is when the phase is 0 every task meets its first deadline then, the task set will be schedulable.

(Refer Slide Time: 15:21)



As we are saying that, when number of tasks is more it becomes difficult to draw the schedule. It takes long time and erroneous we can have a simple formula here, by which you can check each task whether it meets its first deadline. So, the formula is very intuitive see, if we want to check whether a task T i will meet its first deadline. Now, let us assume that T1, T2 , T 3, Tn are arranged in order of their period, that is T1 is highest priority and T n is lowest priority right.

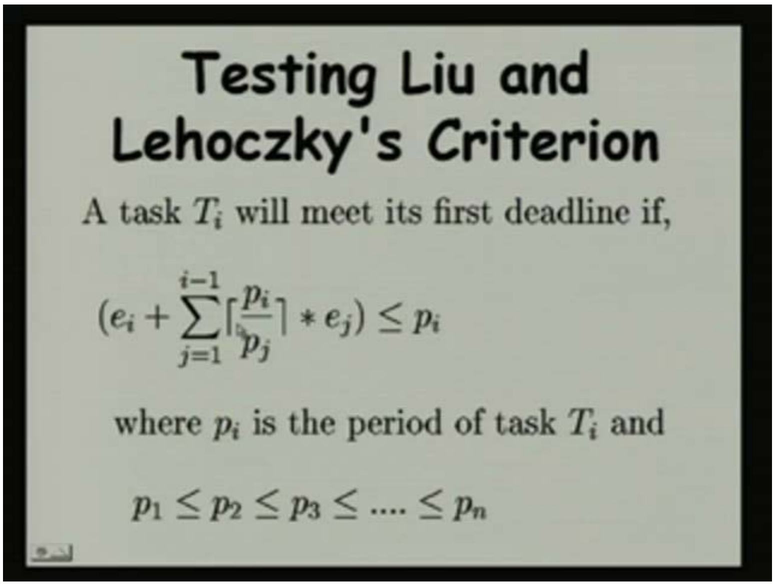
So, as long as there is a higher priority task the rate monotonic algorithm will not allow a lower priority task to run. So, using that simple thing that as long as there is a higher priority task the lower priority will not be allowed to run. So, the response time for the task T i will be e i plus sigma p i by p j ceiling into e j that is to be summed for all j equal to 1 to i minus 1.

So, we are checking for a task p i here, now we have to check whenever a higher priority task is there it should run for e j every higher priority task T j will run for e j and that time e i cannot do anything it has to wait right. So, during the 0 to p i how many instances of p j will occur. So, let us say p i is 100 now, p 3 let us say, let us say p i is p 4, p 4 is 100 and p 3 is let us say 30. So, how many instances of 30 will occur before 100. What do you think?

Four four.

Four because at 0, 30, 60 and 90. So, four instances will occur.

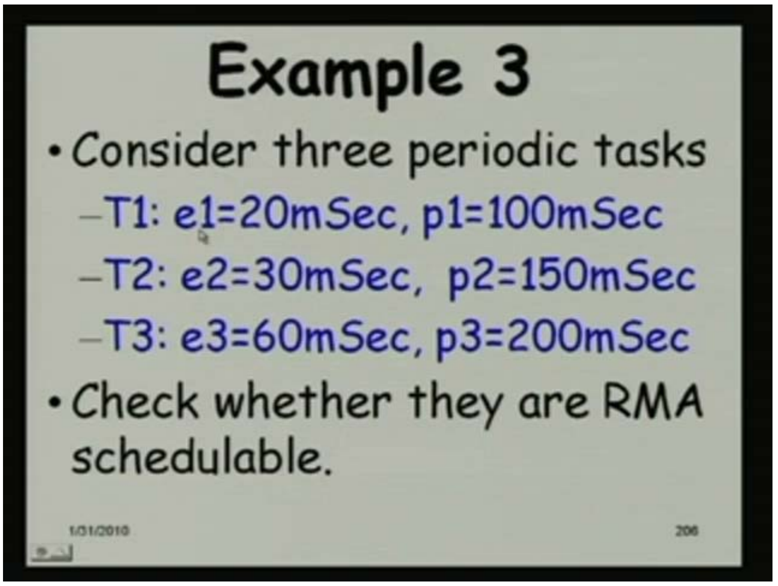
(Refer Slide Time: 15:21)



So, that is why we have made it taken a ceiling here, hundred divided by thirty ceiling. So, that will give us 4 and each time, it runs it will take e j amount of time and if as long as for every higher priority task the C P U is yielded, and e i is taken into account if that is less than p i then, the task T i will complete before p i does that appear, everybody is with this ok.

So, this is the basic formula we will use we will use this formula for every task, to find out whether it will be meet its first deadline

(Refer Slide Time: 18:29)



Let us take an example. So, 3 tasks here, 20 millisecond and 100 millisecond that is T 1, T2 runtime is 30 millisecond and period is 100, 50 T3 is 60 and 2 100 now, check whether they are RMA schedulable please try out.

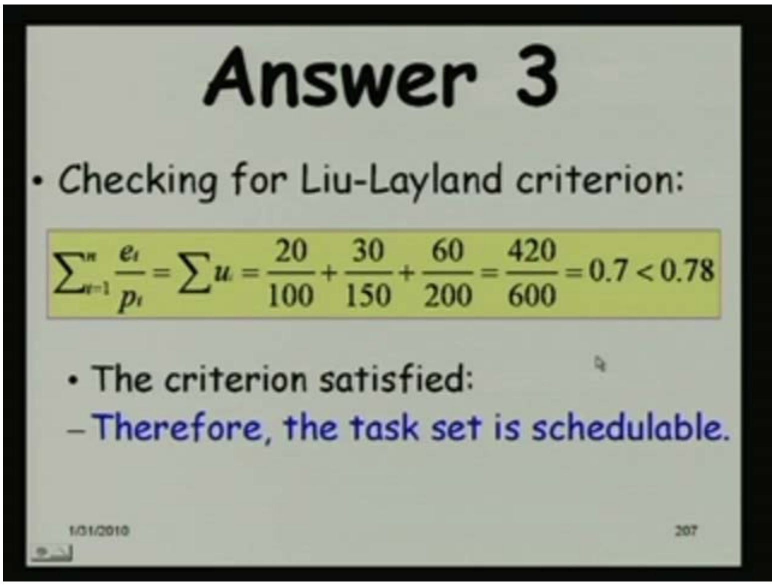
So, use the completion time theorem to check whether these 3 tasks will be schedulable. You you can first try the liu layland right check if it is schedulable then you do not have to use completion time theorem otherwise, you have to check for every task using the completion time theorem check if they are schedulable.

So, liu layland will have to find the utilization and then, check if it is n into 2 to the power 1 by n less than, i mean the utilization is less than n into 2 to the power 1 by n minus 1 and for 3 tasks, it is 3 into 2 to the power 1 by 3 minus 1 which is 0.778. 0.778

Schedulable.

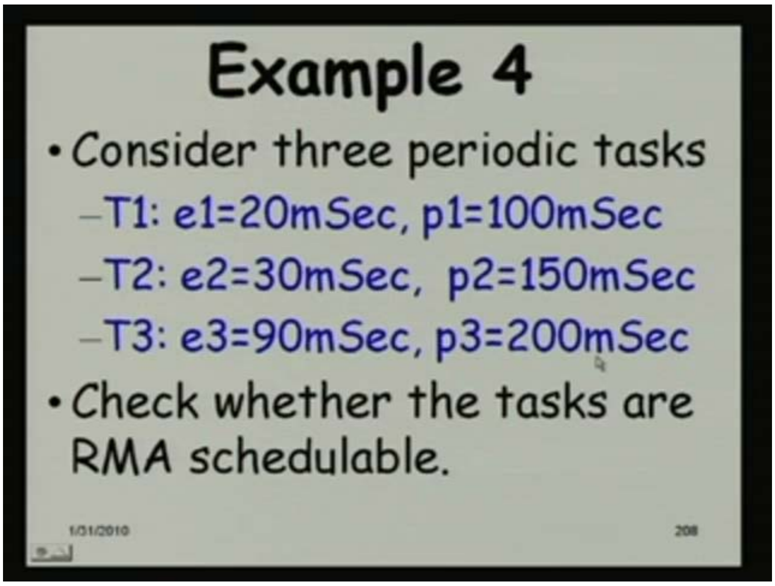
ok.

(Refer Slide Time: 20:00)



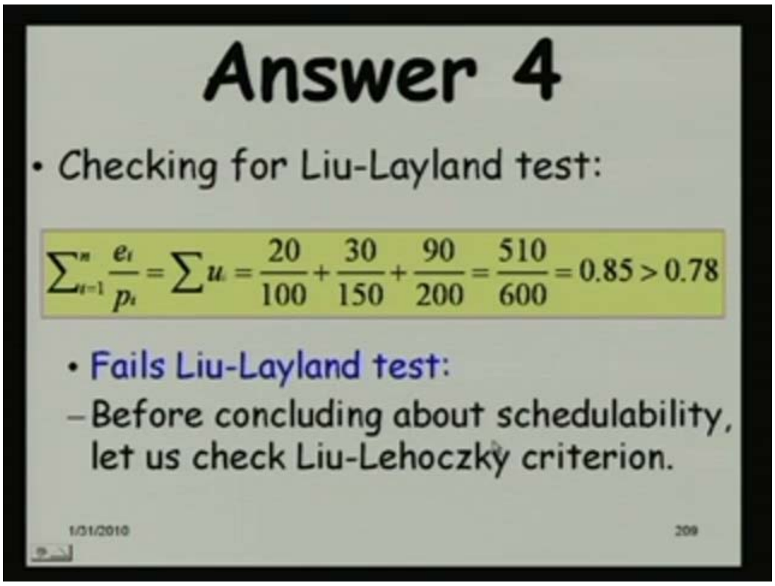
So, some of you found out that its schedulable let us, check the answer yeah it is schedulable, because if we check the utilization for the 3 tasks. 20 by 100, 30 by 150 and 60 by 200 is 0.7 and by liu layland criterion itself, we can say that, the task set is schedulable we need to look no further.

(Refer Slide Time: 20:27)



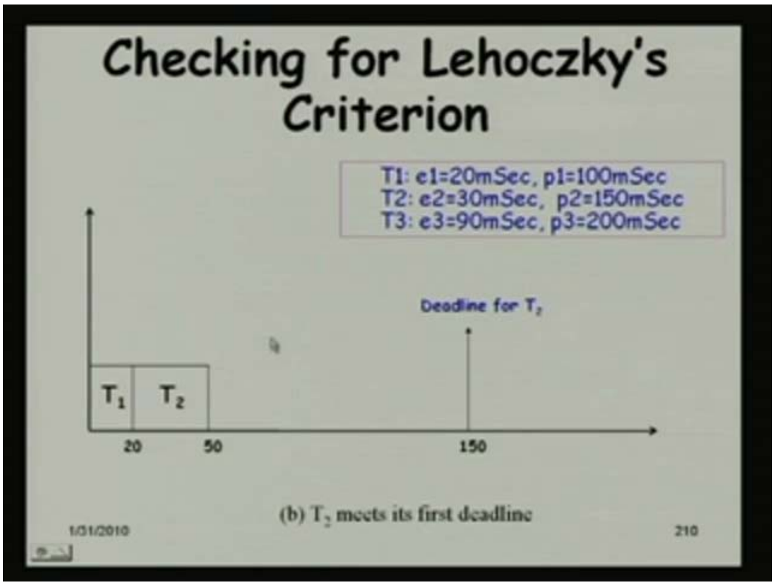
Now, let us try a different task set let us say 20, 100, 30, 1, 50 and 90, 200 for the 3 tasks. Check whether liu layland criterion is satisfied find the 3 task utilizations and sum them up not satisfied right are is everybody getting liu layland criterion is not satisfied, but please try the completion time theorem for every task please check whether the first deadline, will be met meeting. Right yeah let us see that,

(Refer Slide Time: 21:10)



So, if we check for liu layland test the utilization is working out to be 0.85. Which is greater than 0.78 and it fails the liu layland test, but before concluding about the schedulability we will have to check the third criterion that is the liu lehoczky's criterion.

(Refer Slide Time: 21:36)



So, these are the 3 tasks 20 100, 30 100,50 and 90 200. First let us, try the first task T 1. So, let us try drawing the schedule and see what happens and then, we will use the formula.

See, the first task is the highest priority, because it has the shortest period, and whenever T1 is there none of the other tasks will be able to run and under 0 phasing T1 will run and complete by 20 right. So, the for T1 the liu lehoczky's criterion is satisfied.

What about T 2. So, T2 is the second highest priority it will start to run as soon as T1 completes right see here, T1 has completed at 20 and T2 will start running for 30 millisecond. So, T2 the first instance of T2 will also complete before the deadline. So, even for this value lehoczky’s criterion is satisfied.

That is not 60.

30 sorry what is 60 oh the problem that we had given was 60 is it?.

20.

T 2 greater than it should be 60.

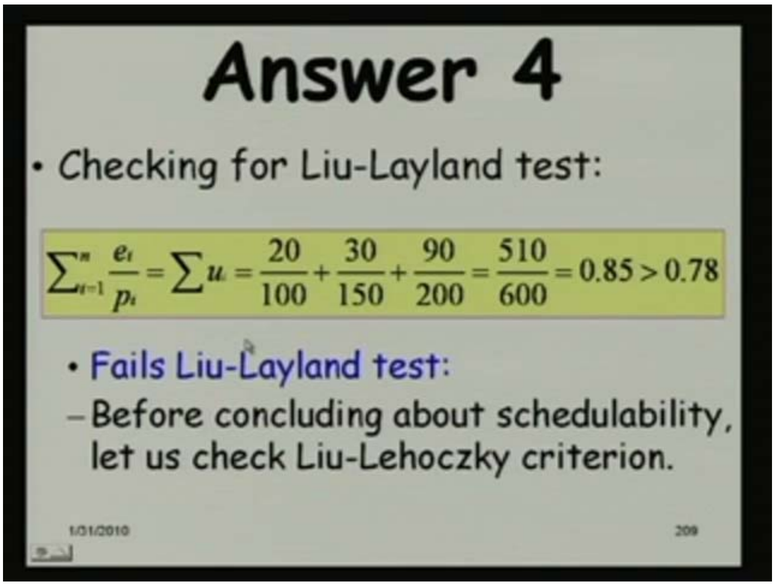
100 it is not 100,50 is it.

(( )).

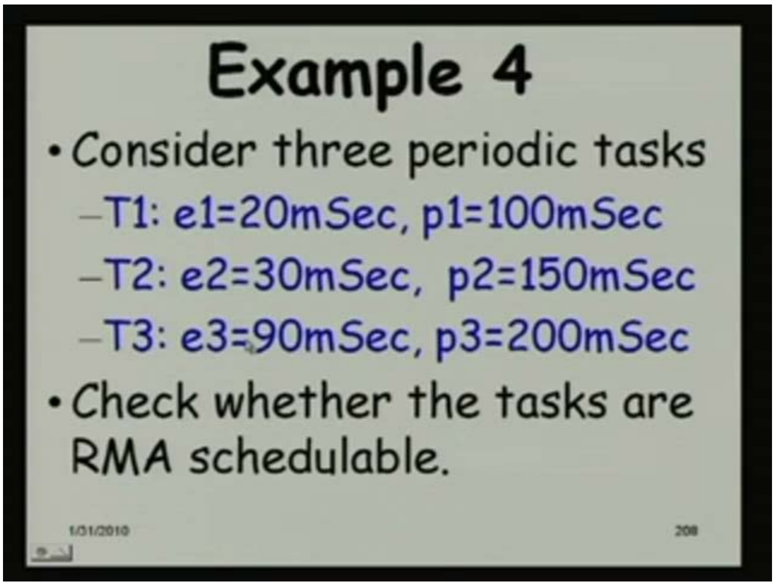
ok.

So, we need to just if it is 60 that was given. So, we have to check whether this deadline is satisfied, let me just verify about your saying.

(Refer Slide Time: 23:18)



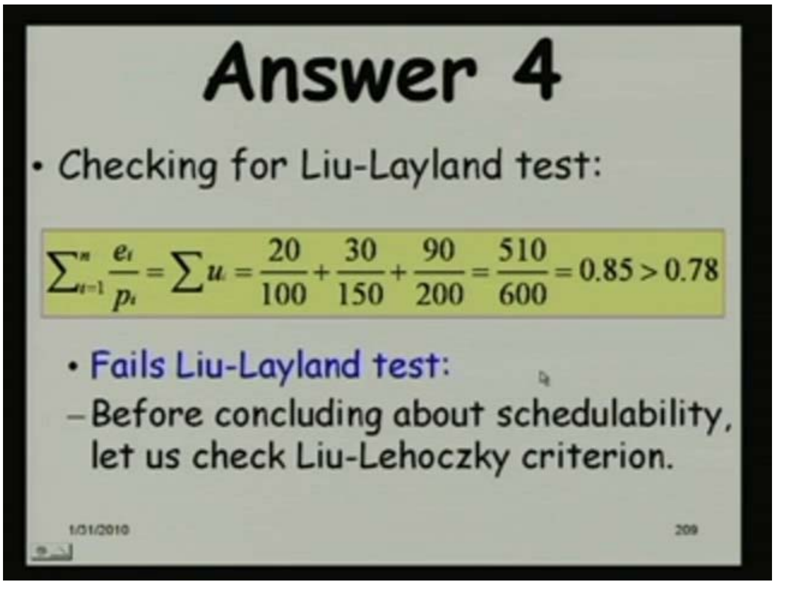
(Refer Slide Time: 23:21)



(( ))

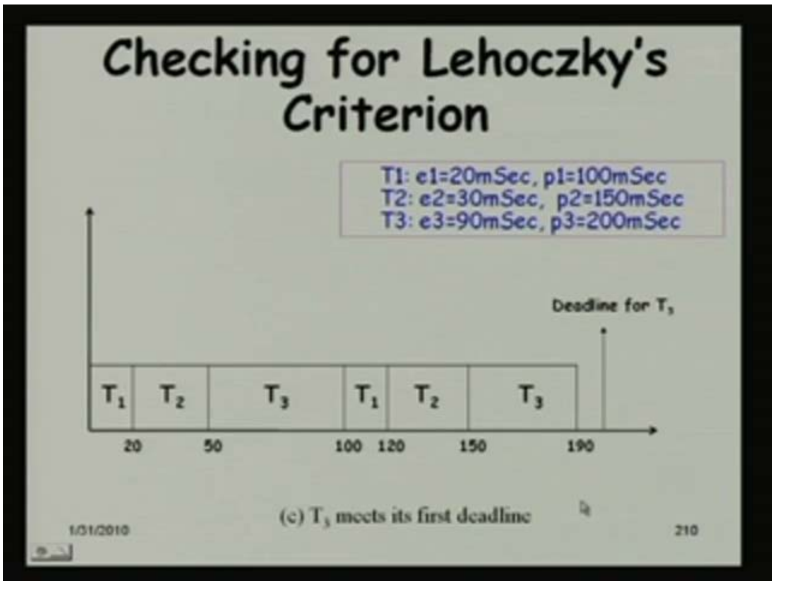
No, 30 this is know see, 20 and 100, 30 and 150, 90 and 200. That is alright.

(Refer Slide Time: 23:31)



So, the utilization is 0.85 fails the Liu-Layland test now,

(Refer Slide Time: 23:40)

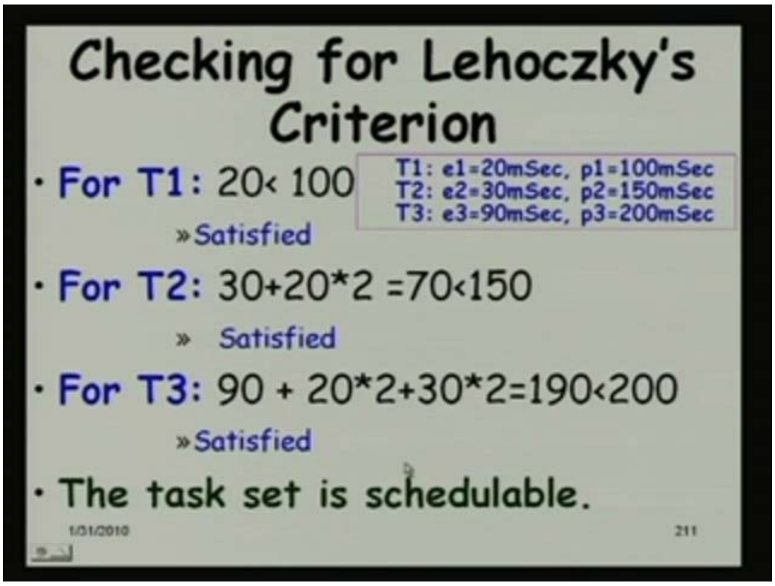


the three tasks are T1, 20 and 100; T2 is 30 and 150; T3 is 90 and 200. So, the first task will complete by 20 meet its first deadline that is at 100 the second task starts running from 20 and its run time is 30. So, completes by 50 and its deadline is 150 meets it well, before that is not it.

Now, the third task will run for 50, here 50 millisecond and by 100, the second instance of T1 will arise and then T1 will run and also the second instance of T2 will arise. So, that will also run and after that only T3 will again run and it will complete by 190. So, that is within 200 millisecond.

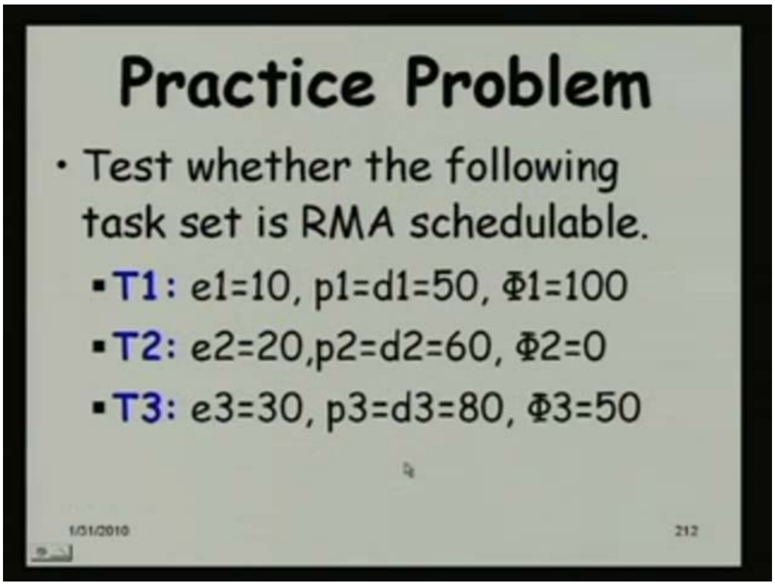
So, all the three tasks meet their first deadline and the task set is schedulable by drawing of the schedule.

(Refer Slide Time: 24:48)



We can also use the formula that we had given the first task no higher priority tasks exist. So, 20 is less than 100 satisfied the Liu-Lehoczky's criterion satisfied for T1 for T2, there is only one higher priority task that is T1 each time takes 20 millisecond and in 150 millisecond, it will arise 150 by 100 ceiling which is 2 times. So, 2 into 20 plus 30 is 70 is less than 150 satisfied for T2 also and for T3 there are 2 higher priority task T1 will arise 2 times T2 will also arise 2 times by the ceiling right. 200 by 100 ceiling is 2 and 200 by 150 ceiling is also 2. So, 90 plus 20 into 2 plus 30 into 2 is 190 is less than 200. So, for this also satisfied. So, the task set is schedulable.

(Refer Slide Time: 25:59)



Can you quickly try, this problem whether three tasks whose execution time is 10 and the period and deadline are both 50 millisecond and the phase is 100 millisecond and task T2 execution time. 20 millisecond and the period is equal to deadline is 60 millisecond and the phase is 0 and T3 is run time 30 millisecond period and deadline is 80 millisecond and phase is 50 millisecond?

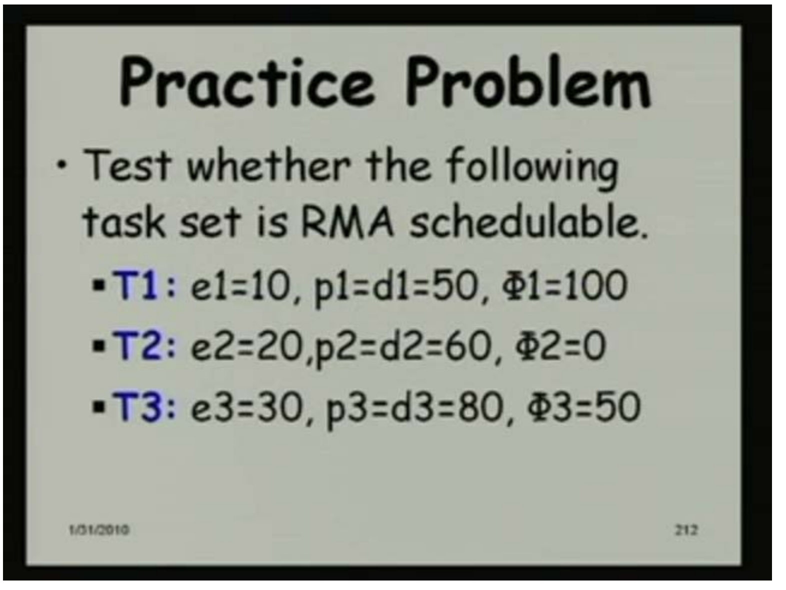
Will, it run satisfactorily under RMA scheduling? So, please workout because unless we workout the problem. We would not really understand, the intricacy we will go proceed further from here and check the shot coming of the completion time theorem anybody has.

No.

No the answer is negative anyone anyone else let us look at the solution please meanwhile try.

So,

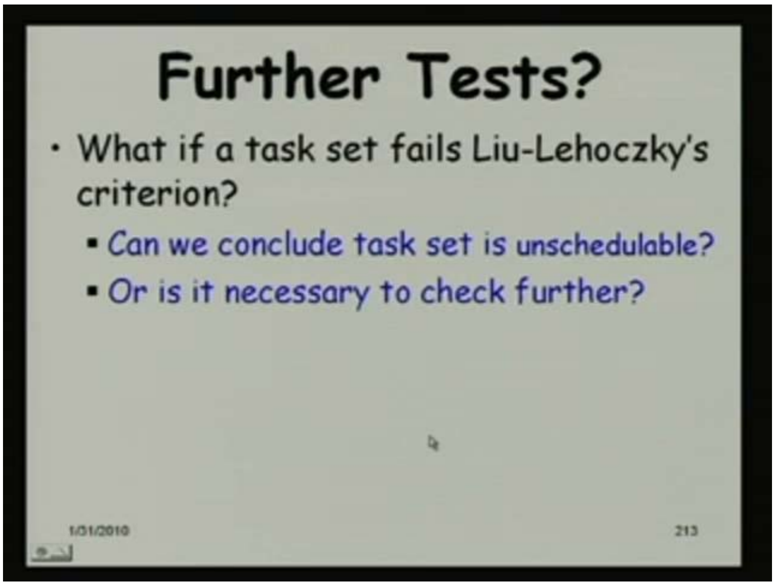
(Refer Slide Time: 25:59)



I have not got the solution here, but it is not too hard to try, because we have to consider 0 phasing ,if it runs under o phasing then the set of tasks will be schedulable for all phasing's. So, the first one will meet its deadline this has the highest priority. It will complete by 10 whereas whereas, the deadline is 50 thus what about the second one second one it needs 20 millisecond runtime and by this time two times e one will arise isn't it 60 by 50 ceiling is 2. So, 2 into 10 plus 20 sorry 2 into ten plus 20 is fo40. So, 40 is less than 60. So, even T2 meets its deadline.

Now, what about T3. So, T3 runtime is 30 and period is 80 and in 80, 2 instances of T1 and 2 instances of T2 , will occur right. So, 2 into 10 is 22 into 20 is 40. So, 40 plus 20 is 60 sixty plus 30 is 90, which is more than 80. So, for the third one the lie lehoczky's criterion is not satisfied and according to lie lehoczky this is not schedulable.

(Refer Slide Time: 29:03)



But what do? We do if the task set fails lie lehoczky's criterion.

Sir,

Yes please.

Sir, earlier in the last to last example.

Yes.

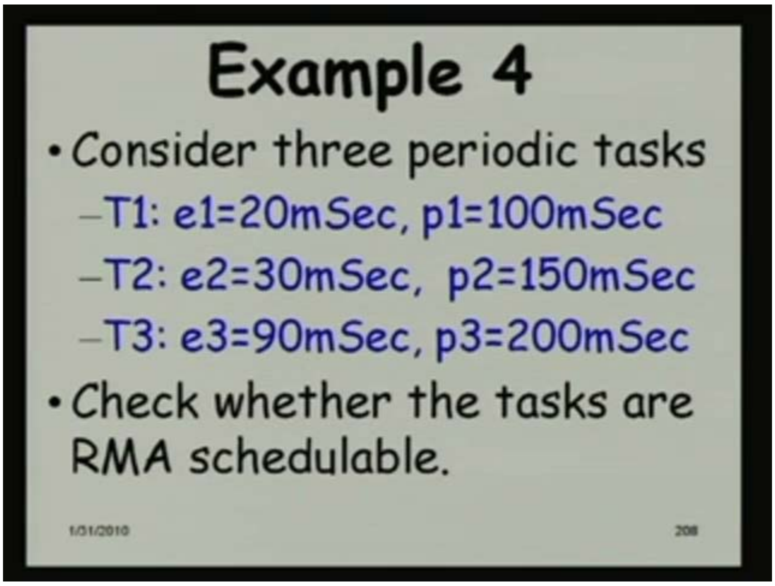
Sir.

Last to last let me just get there

last to last example

Yes

(Refer Slide Time: 29:19)



This is the one right.

Yes sir.

Yeah.

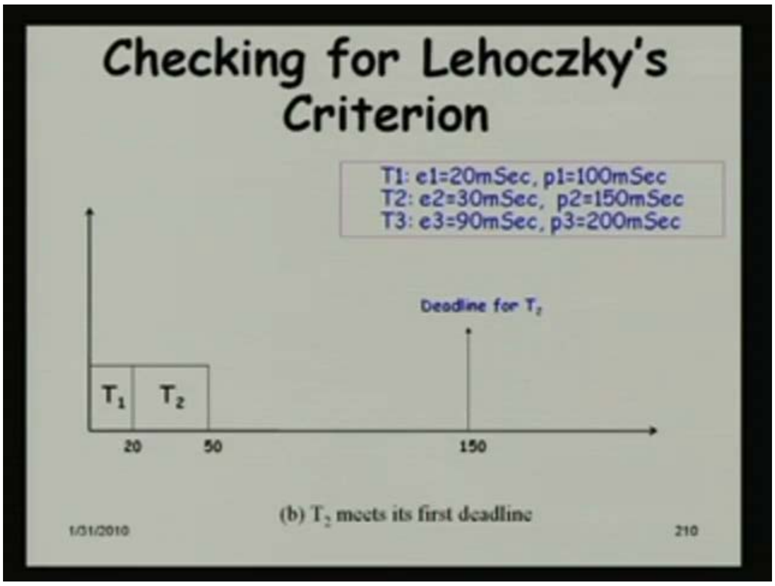
Sir, when we draw the when we drew the schedule.

Yes.

Ah we get the second task completing at 50.

Yes.

(Refer Slide Time: 29:31)



This one right.

Yes sir.

Yeah second task.

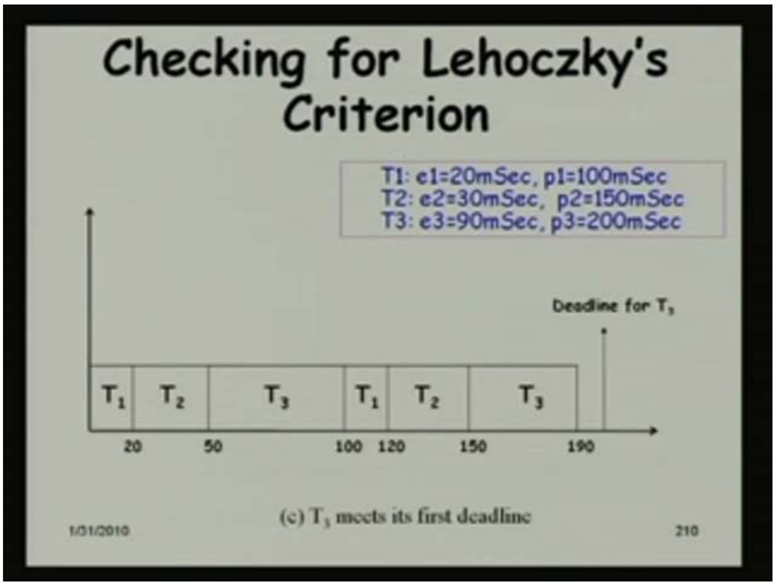
But, when we put in the formula we get a value 70.

Let's see.

We get the value 70.

I think the drawing must, be incorrect there.

(Refer Slide Time: 29:49)



Sir, drawing is correct.

Correct ok.

But, what we are considering is that we are we are ah means, we are trying to what we are what we are trying is that we are trying to know how many times the task will.

Yes.

Run there in the whole period.

Exactly, how many times T1 will occur.

In the whole whole period.

150 yes.

But in during the.

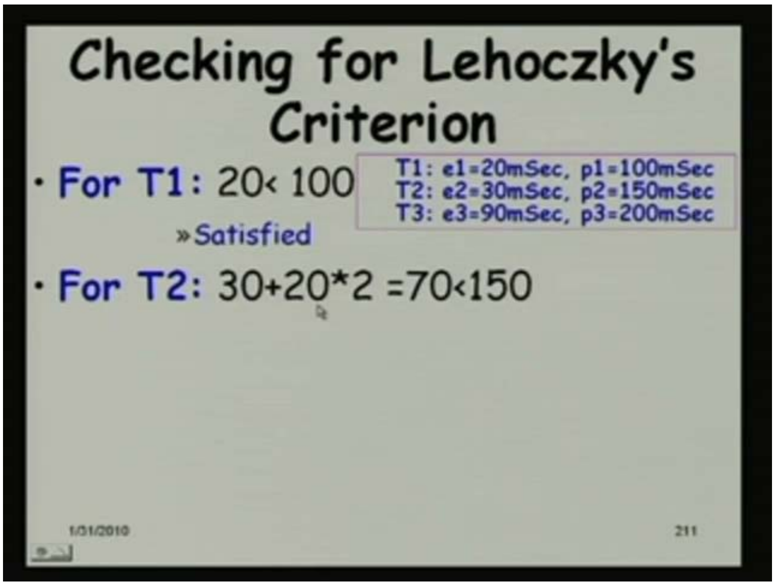
Before, the execution yes exactly.

So, this formula as he was he pointed out sorry, what is your name.

Ashish.

Ashish as ashish pointed out just see, here in the formula.

(Refer Slide Time: 30:32)



What, we are implicitly assuming that, this 20 the task set two by the time. It will occurs the second instance sorry the T2 would not have completed that is what is important, what we are licitly, we are assuming and that is the worst case actually.

So, by hand drawing, we are getting a more accurate picture compare to this this approximation. What he says is correct, actually because here even though the second task occurs only at hundred milliseconds by this time. It has already completed T2 , but we are taking T1s execution time also into account.

So, what will be it is impact, is it possible? That by using the hand schedule, we can conclude that something is schedulable and by using this. You can conclude that something is not schedulable, is it possible? yes you think. So, it will be; we will conclude by using the formula the expression.

I think, we are waiting for the phase phase things we the the formula.

No phase, we have not phase is not there, but let me just please answer this question that even though you said that the schedules or the completion times per a task is different as concluded by the formula is compare to the hand drawing of the schedule, is it likely that? We will get a different result by hand drawing compare to applying the formula is it possible.

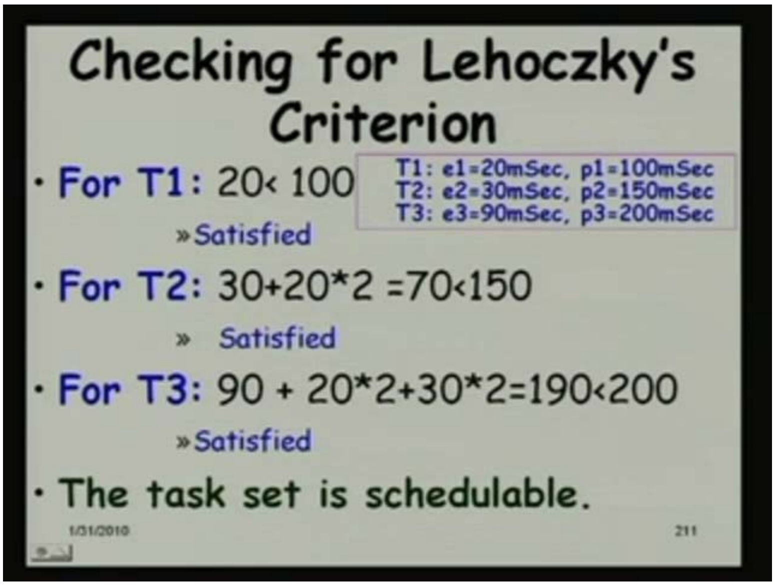
Even, if it does not satisfy the formula it maybe schedulable.

Is it? So, you think So, what about others. It will be schedulable in that case; I just ask you to just construct, one example where it fails the formula, but by hand drawing it is schedulable.

The formula detects the worst case scenario.

Worst case, but see see to convince yourself just give me, one example where the by formula it says not schedulable and the hand drawing. It is schedulable just construct one for one for me I think you will be disappointed not right now you try later you will be disappointed, because see, here this we are considering the entire period here and as long as it occurs within that period right within 150 we should consider it in the that is in the worst case this 20 will be considered anyway.

(Refer Slide Time: 33:01)



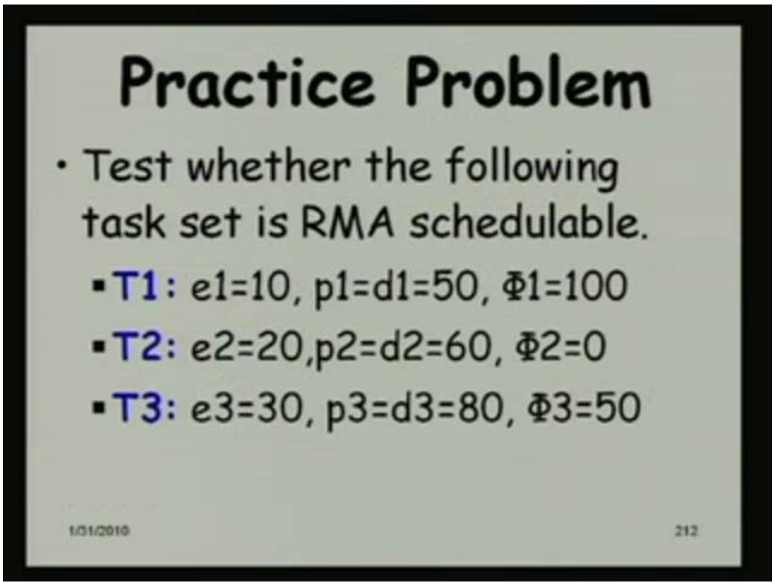
So, you will not be able to construct an example, where it will fail this formula and will pass the schedule. I mean hand drawn schedule please try that and if you can construct an example, please give it to me.

Sir, if the period was 4o and 60 instead of 150.

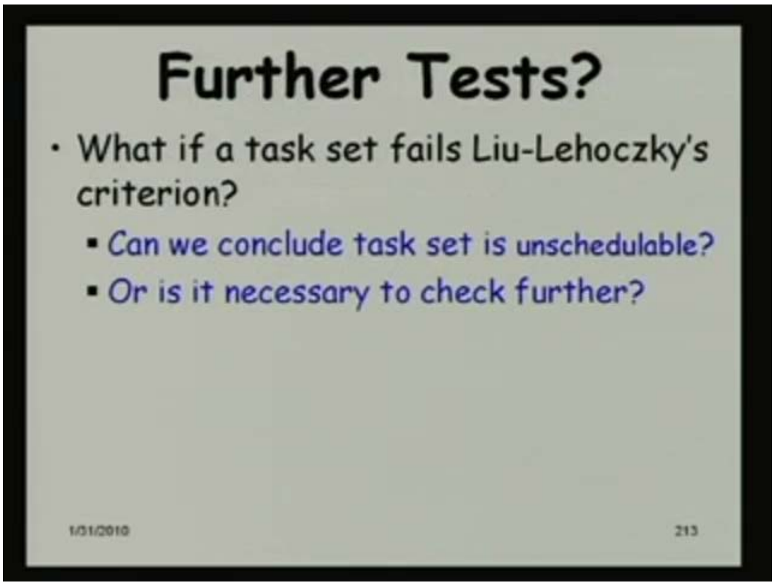
So, please try that please, try that convince yourself and we will we will check that.

So, let us precede next class or whenever you find. You please construct example and give it to me.

(Refer Slide Time: 33:52)



(Refer Slide Time: 33:54)



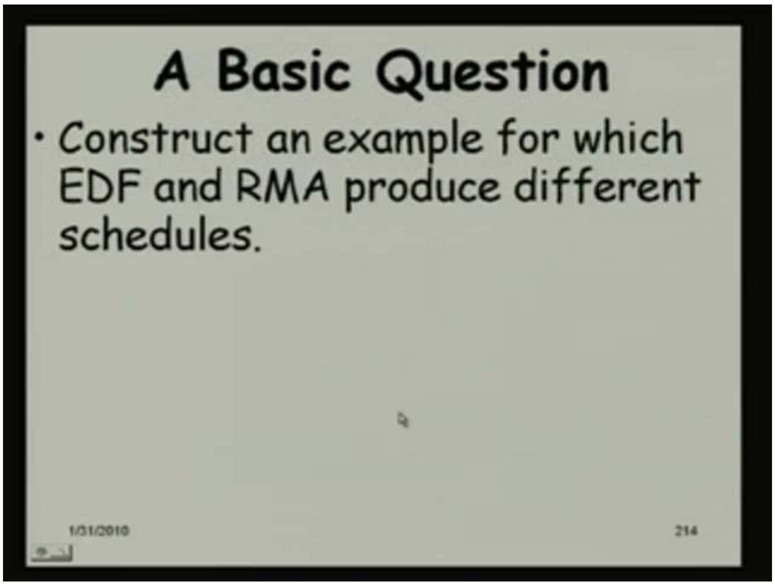
Let us proceed, further I mean his observation was nevertheless reported, because it is by completion time theorem. We are not really saying that task response time seventy is not the task response time we are saying that whether it will complete by 150 or whatever.

If it is response time; yes there is a problem here. You are not computing the response time; we are checking whether it will meet its first deadline and that is not period of 40 and 60 in that.

So, you please try that you draw. The schedule and give it to me we will we will display that ok.

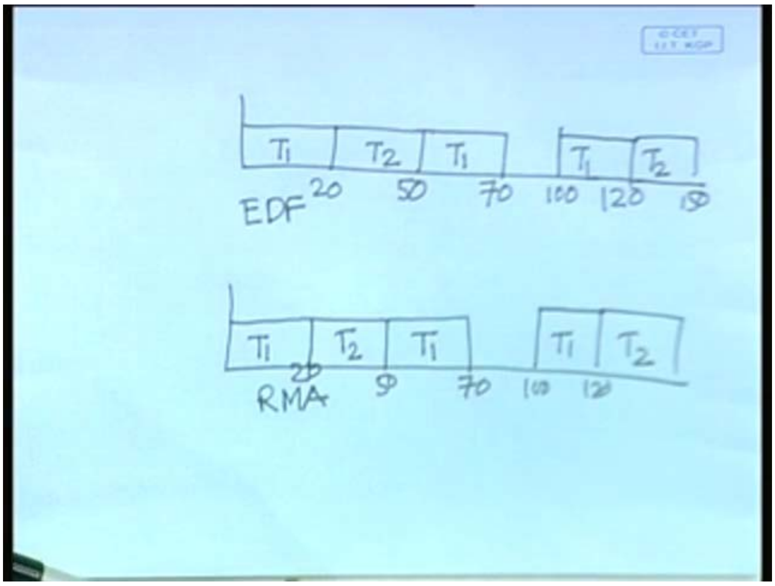
So, now what if a set of tasks fails lie lehoczky's criterion can, we conclusively say that task set is unschedulable or is there necessity for further check actually lie lehoczky's checks. The worst case scenario that is the 0 phasing case, but if the phases are non 0 then, it is possible that the lie lehoczky's result is negative, but we have the task set schedulable because of the phase between the tasks.

(Refer Slide Time: 35:18)



Now, very basic question let us try to understand see for most of the task set, you try to draw the schedule using EDF and RMA

(Refer Slide Time: 36:06)



You will find that they have the same schedule; you just tryout take any set of tasks. Let us say, T1 is 20 and 50, T2 is 30 and 100. Check the EDF schedule and RMA schedule is it different? Or is it the same schedule? both RMA and EDF are giving. So, check 20 and 50 and 30 and 100 are 2 tasks.

So, let me just write here say T1 execution time is 20 and the period deadline is 50 and T2 execution time is 30 and period and deadline are 100. So, the schedule according to EDF

T one square

It will start with T1 that has the earliest deadline. So, let me draw for EDF it will start with T1, because that has the earliest deadline and it will run after 20 and 20 T1 has completed and T2, will run for 30 and again. T1 will run and again T2 will run and. So, on

Now, if you tryout RMA same thing T1 runs, it has the higher priority and after that T2 runs after that T1 runs, T2 runs. So, both EDF and RMA are producing identical schedules for this task set. You will find that for many task set, they are scheduling identical schedules. Can you construct one task set for which they produce different schedule? that will help us to understand the behavior of.

Out of the T2 we have there will be a gap for 100, 200.

After hundred T2 will appeared.

T 2 ok.

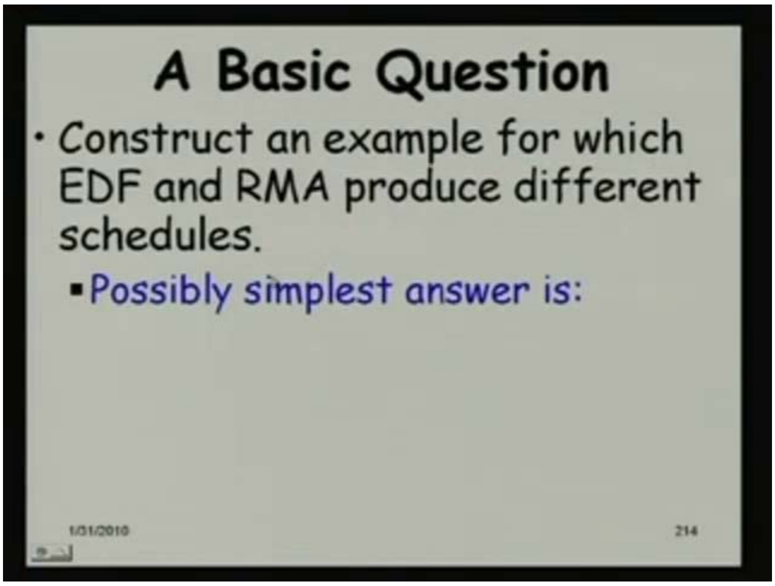
See for both of them let us draw again neatly yeah what he says is correct. So, T 1 runs for 20 then T 2 runs for 50 sorry 30 and by that time. T 1 does not T 1 exist by that time. Is not it. So, T 1 as arisen by that time runs up to 70 and by that time T 2 does not exist arises only at 100 and by 100 again T 1 exists right.

So, T 1 will run and then at 120, T 2 will run is not it for 120, 150. So, this is the EDF. Now, let us try the RMA. So, T1 runs up to 20 then T 2 runs up to 50 and by that time T 1 again arises runs up to, 70 and after 70 the run out task instance is to run and again at 100 we have T 1 arising and that has higher priority. So, T 1 will run to120 and after that it will run. So, identical schedule both EDF and RMA.

Can you just construct one example, where EDF and RMA produce different schedules that will give, us some understanding into the scheduling policies of EDF and RMA and how they are different. So, just construct one example of a task scenario, where T 1 and T2 sorry where RMA and ED will have different schedules.

Please think about the basic principle between EDF RMA their schedulability. That is the scheduling constraint and that should give you, some hint any one thinks of some example not too difficult. Let us see the answer see the simplest answer is that.

(Refer Slide Time: 41:07)



When something is on schedulable by RMA, but is schedulable by EDF definitely that schedule will be different. Is not it?

Because by this schedule RMA is not able to schedule it. So, EDF must be using a different schedule is not it that is one hint. I mean this is the simplest answer to construct a task set, which is have different schedule under RMA and EDF another example.

Sir, we have said that RMA is an optimal static uniprocessor.

But, EDF in the dynamic.

Yeah, exactly see something not schedulable by RMA, will can be schedulable by EDF. Because EDF is the dynamic priority algorithm, but if you anything is not schedulable under EDF RMA cannot schedule that.

So, this is one example simple example where they produce different schedules, but suppose we say that see give us an example, where the task set is schedulable under both RMA and EDF, but the schedules are different in that case we have to consider the task set in which the task is preempted before completion.

So, possibly the example that we are just saying the three tasks where the task T3 was getting preempted right before, it could complete another task is to arise and preempt it and that is the situation. Where these two will produce different schedule please think about it why they will produce.

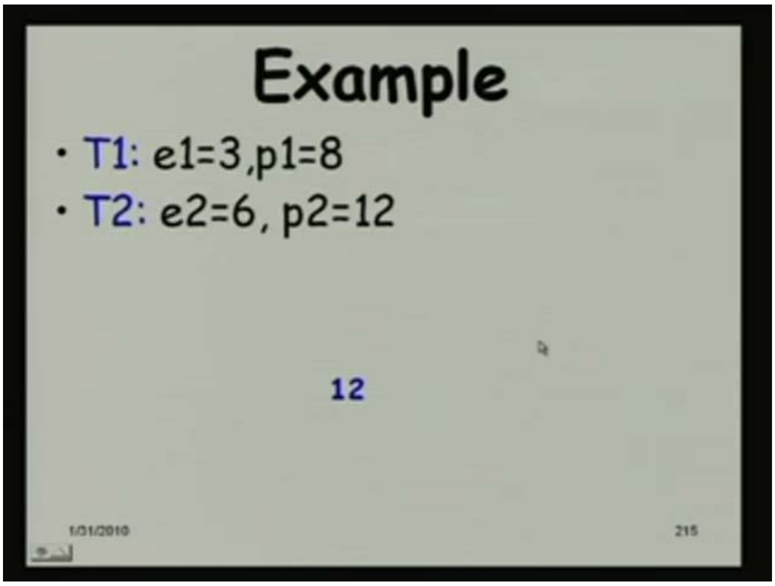
(( ))

Sorry.

EDF is also preempt able, but what I am trying saying is that in RMA scheduling if the tasks one task is getting preempted in EDF. It may not be preempted because, it will already having the deadline very near. So, just because it had a higher priority does not static priority does not mean, that it will preempt it in EDF in EDF. It will check the deadline each time. So, the preemption condition for both these algorithm is different. So, if you consider the preemption case task preemption case you will find. That they construct different schedules ok.

Let's proceed further. So, please tryout some examples, for your understanding.

(Refer Slide Time: 43:35)



This is one example, where they produce different schedule see task e one sorry task T1 the execution time is three millisecond and period is 8 millisecond, T2 is 6 and 12. So, T1 has higher priority runs up to 3 and then T2 runs up to it should be nine is not it.

Yeah, it should be nine. So, it runs up to 9 and then

Why should 8 and (( ))

8 is it.

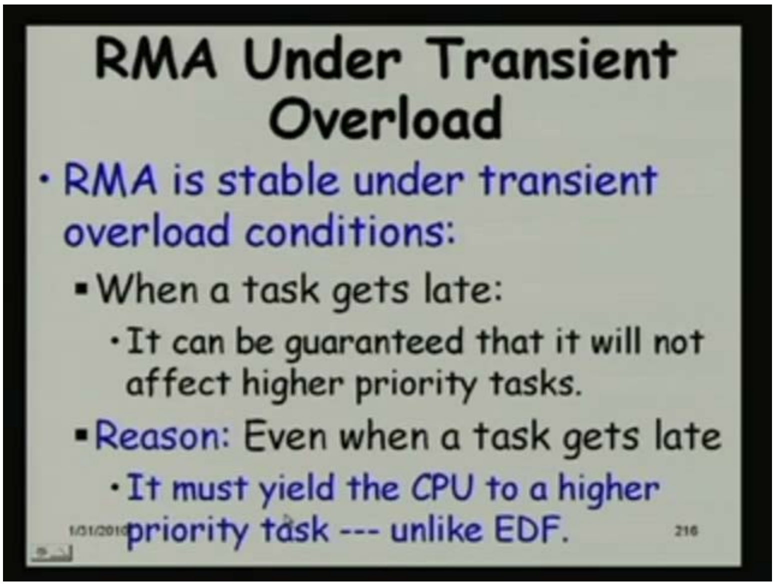
(( ))

ok T2 sorry no it runs for 3 and then [FL] right it runs up to 8 and gets preempted here that is what, we are saying it is getting preempted here because, of the static priority.

Just check here that at this point T2s deadline is at twelve right T2 ’s deadline is at 12 and T1 ’s deadline is at 16 right. So, T1 has a further deadline then T2 and T1 should not preempt T2 according to EDF. So, we check the EDF, that is what happens here.

So, T2 will complete and then only T1 will run right on the other hand here just because ,T1 had a higher priority. It is preempting as soon as it has arisen at 8 and it will take this 3 to complete different schedules of RMA and EDF looks ok let us proceed.

(Refer Slide Time: 45:25)



We had said, that EDF behaved very poorly under transient overload, but let us see the RMA behavior under transient overload, we will see that RMA is stable under transient overload. When does transient overload occur can anybody answer this question? why why does a transient overload occur. There are many events.

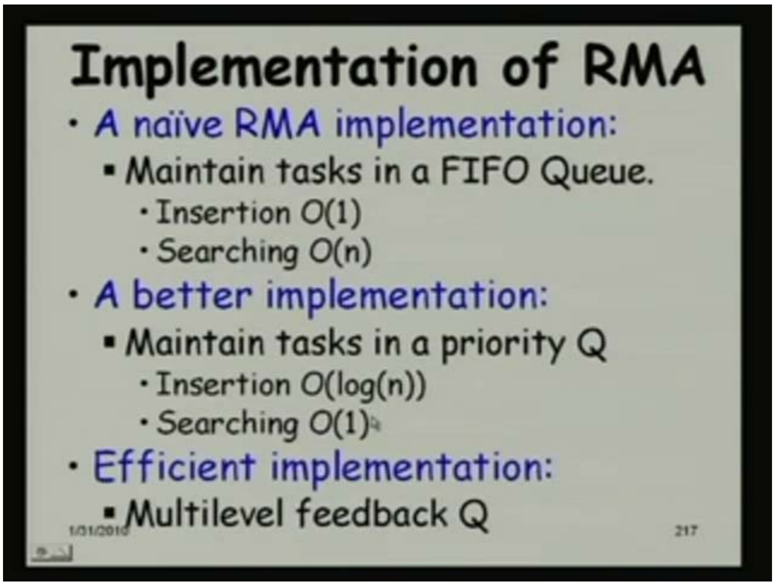
So, one situation is that one event is giving rise to many tasks, suddenly system will get overloaded like, we are saying that a fire alarm a fire situations was sensed and many event many task had to be started.

There can be other situations also like one task for some reason starts getting late. So, the system will get overloaded. So, let us see under the both the situations one is task is getting late due to whatever reason possibly that it took a branch, which was not used well and it started taking more time. In that branch have got into an infinite loop and many reasons actually.

So, when a task gets late according to EDF. Its priority will keep on increasing. Because, it is deadline is approaching, is not it? But here the priority static whether it gets late or not it will be preempted as soon as there is a higher priority task. So, it cannot block a higher priority task just because, it is getting late as long as there is a higher priority task unlike EDF. The CPU must be yielded by a lower priority task.

And the other situation also, where we have many tasks arising suddenly there also the highest priority task will never miss its deadline because, that is the basic of this scheduling algorithm.

(Refer Slide Time: 47:24)



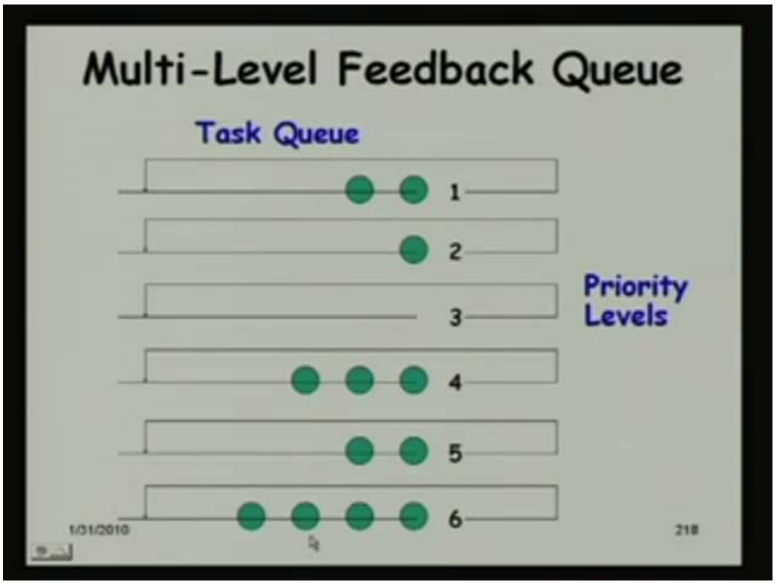
So, it is stable under transient overload.

Now, let us see, what are the possible implementations of RMA because most of the commercial operation system many, of them we will see that they just give you facilities to set priority statically and then, you implement the RMA.

The simplest implementation is a first in first out queue insertion is simple. I mean is efficient o of 1 whereas, searching you have to search all the tasks that are o n. So, considering the number of searches, you have, it is very inefficient a better implementation would be to maintain a priority queue and here insertion is log n and searching is o 1.

But a more efficient implementation is a multilevel feedback queue.

(Refer Slide Time: 48:34)



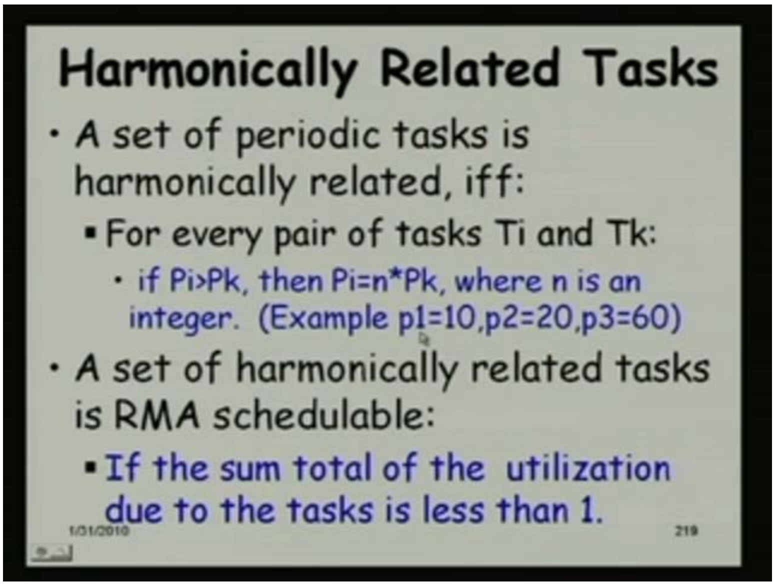
So, in a multilevel feedback queue, what we have is we arrange the tasks in different queues their priority order. So, 1 is the highest priority 6 is the lowest priority. Now, the when you insert a task, you just insert in the appropriate queue based on its priority and when you select the task, you start looking from the highest priority, if something is available here some task run it. If not look at the next one.

So, what are the insertion time and the searching time? Yes, consider the average case. So, that is an assignment for you; please compare the efficiency. For the simple implementation the priority queue and the multi level feedback queue; just do some analysis and so, which one will be better by how much.

You, can make any assumptions that, you feel like like how many, tasks you can consider a bound on the number of tasks and the bound on the number of priority levels and so on. So, please submit your answer on the analysis of the implementation. If there is a good answer, we will we will give some bonus marks for the answer.

So, that is analyze the time and space complexity for the multilevel feedback queue and compare with the other implementations. There is no time limit on when you want to submit the answer, but once somebody submits the answer will not consider the other answers unless, there is a significant difference in the thinking procedure etcetera.

(Refer Slide Time: 50:58)

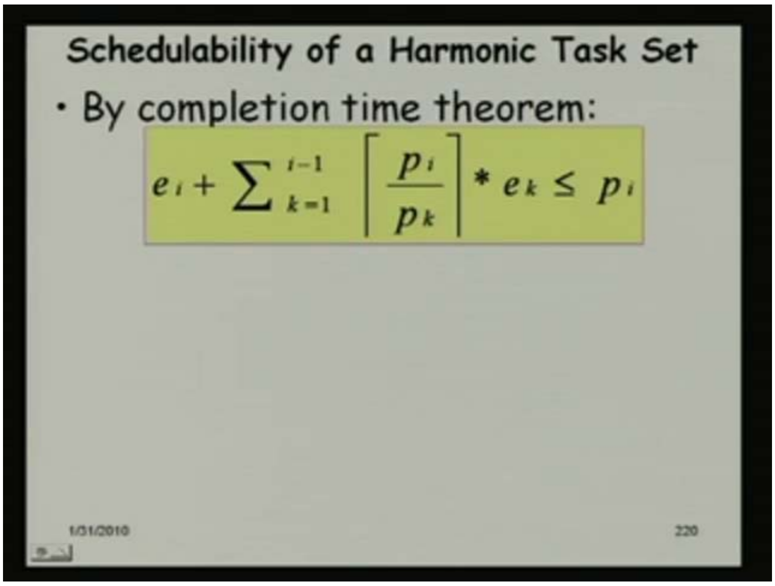


So, now let us check the case for harmonically related tasks. We call a set of tasks harmonically related. If whenever we take a pair of tasks T i and T k. If the period i is greater than p k then p i will be n times p k where n is a integer.

For example, for three tasks whose period is 10, 20 and 60 are harmonically related, because 20 is10 into 2, 60 is 3 into 20 and 6 into 10. So, every higher period is integral multiple of every other lower period then we will call it as a harmonically related set of tasks.

The interesting result is that, if a set of tasks, the periods are harmonically related then they are RMA schedulable if their utilization is less than one. So, as long as the liu layland criterion was known giving 0.77, 8 and so on, but here you see that for harmonical harmonical tasks, you are able to get almost hundred percent utilization. Please think, what is the reason ok.

(Refer Slide Time: 52:23)



We will show, it more formally in the next hour that why it is less than hundred percent is sufficient condition for harmonic set of tasks. So, we will stop here and continue in the next class.