Real Time Operating System

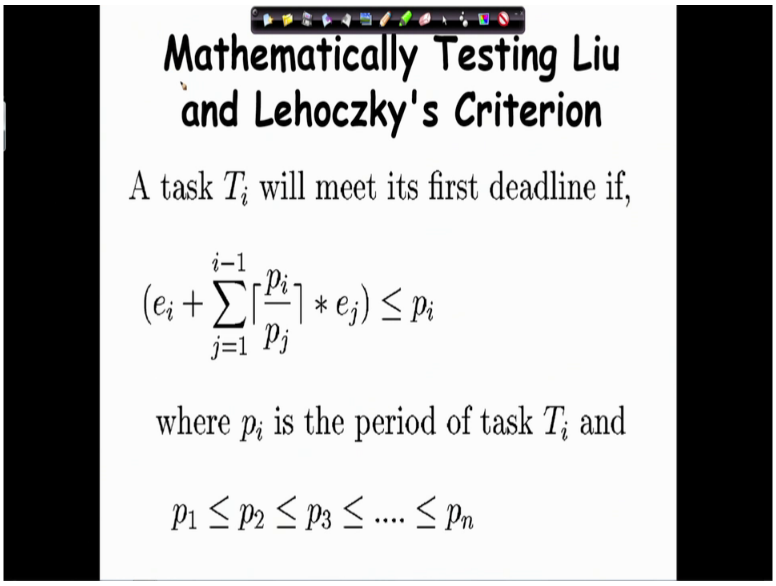
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 10
Rate Monotonic Analysis

Welcome to this lecture, if you recollect over the last few lectures we were looking at some real time tasks scheduling algorithms. Initially we looked at simple event simple clock driven schedulers and later we have been looking at event driven schedulers. Initially we looked at the EDF and then we were looking at the rate monotonic scheduler and in the rate monotonic scheduler we saw that the scheduling algorithm is really simple. It is a preemptive scheduler and we need to assign priorities to tasks based on their frequency of occurrence or their period.

And once we do that you can do some mathematical analysis to say that if tasks set will meet it is deadline and we looked at 3 important results, the first is that utilization must be less than 1 for a tasks set to be schedulable, but that was a necessary condition not a sufficient one, but then we looked at the sufficient condition the Liu Layland condition n into 2 to the power 1 by n minus 1. We looked at the expression and saw that the utilization should be less than for a tasks set to be schedulable, but then you said that that is a pessimistic condition because even if a tasks set fails the Liu Layland condition, still it may be feasibly scheduled and then we looked at the Liu and Lehoczkys theorem and we are trying to do some examples. So, let us start from that point.

(Refer Slide Time: 02:25)



Mathematically Testing Liu and Lehoczky's Criterion

A task T_i will meet its first deadline if,

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

where p_i is the period of task T_i and

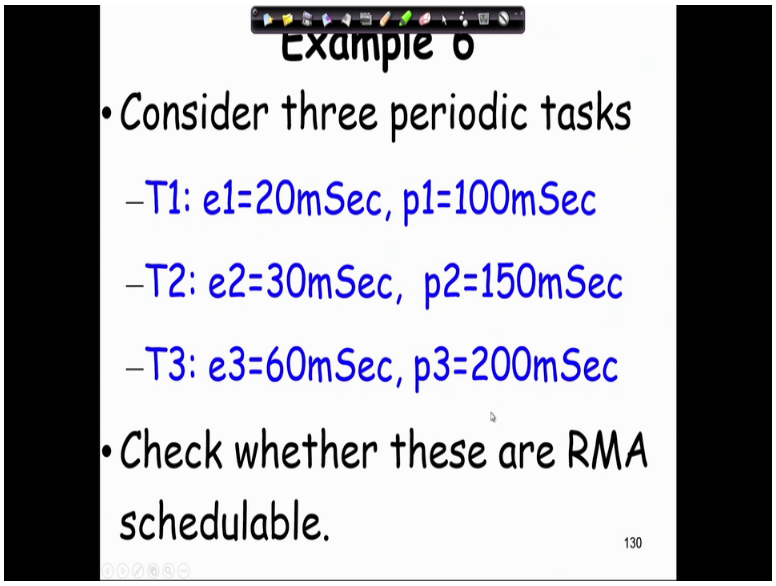
$$p_1 \leq p_2 \leq p_3 \leq .... \leq p_n$$

One was that we said that the main idea behind the Liu and Lehoczkys criterion is that in the worst case situation which occurs where all tasks arrive together, if the tasks meet their first deadline then the tasks will continue to meet their deadline under all phasings, if you recollect the exact wording of the theorem that if a tasks set meets it is first deadline in the 0 phasing condition, then the tasks set will meet it is deadline in all possible phasing's.

And we are trying to check graphically whether tasks set will meet it is first deadline, but that was bit cumbersome and we then looked at a mathematical expression, which will tell if a tasks set meet it is deadline. So, the mathematical expression is given as for task T i for a task T i the first deadline will be met if e i plus sigma j equal to 1 to i minus 1 ceiling of p i by p j into e j is less than the task period the intuitive idea behind this expression, we had seen that whenever there is a higher priority task the task cannot run it is higher priority tasks will run and the time for which the higher priority tasks will run on 0 phasing that is the task arises in phase with it is higher priority task is given as ceiling of p i by p j into e j.

For example if the task period is 20 and you have a high priority task whose period is 7? So, then 20 by 7 ceiling will be 3. So, 3 times at most p j can occur within p I, which is 20 because they have occurred once at 0 once at 7 p j will occur and again at 14 p j will occur and therefore, we can generalise that for any higher priority task p i by p j ceiling into e j because each time p j the tasks j occurs it runs for e j time ceiling p i by p j into e j, that is the time that will the higher priority task T j will run and that will have to be considered for all higher priority tasks that is j equal to 1 to i minus 1. So, for every task T i we need to check if this condition holds if this condition is for every task then we can say that the tasks set meet it is deadline .

(Refer Slide Time: 06:23)



Example 6

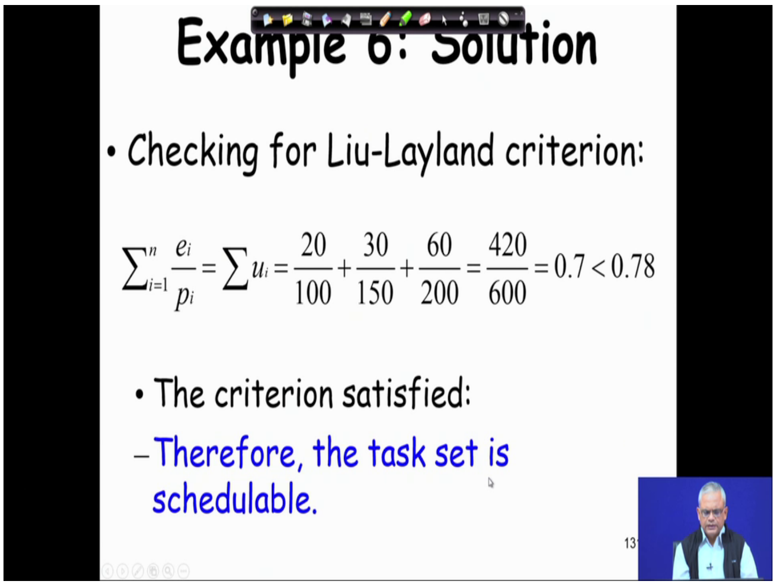
- Consider three periodic tasks
  - T1: e1=20mSec, p1=100mSec
  - T2: e2=30mSec, p2=150mSec
  - T3: e3=60mSec, p3=200mSec
- Check whether these are RMA schedulable.

130

So, now let us look at some examples based on the Liu Layland and Liu lehoczkys expressions, the first example we consider here is 3 tasks T 1 T 2 and T 3 and for T 1 the execution time is 20 millisecond and the period is 100 e 2 is 30 millisecond period is 150 and e 360 millisecond and p 3 is 2 100 millisecond. By the rate monotonic analysis we have to give the highest priority to T 1 because it is period is the shortest next high highest priority to T 2 and T 3 is the lowest priority.

Now, let us check whether with this priority assignment of the rate monotonic scheduling the task set will meet it is deadline.

(Refer Slide Time: 07:29)



Example 6: Solution

- Checking for Liu-Layland criterion:

$$\sum_{i=1}^{n}\frac{e_i}{p_i}=\sum u_i=\frac{20}{100}+\frac{30}{150}+\frac{60}{200}=\frac{420}{600}=0.7<0.78$$

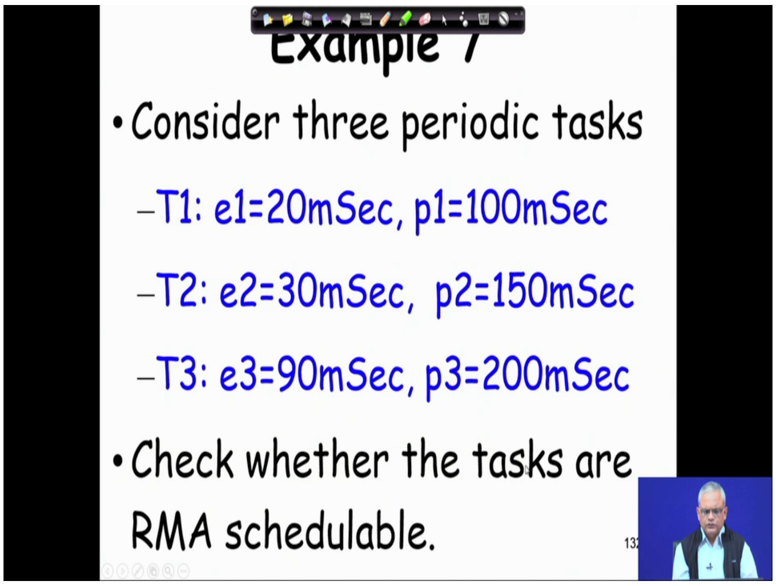
- The criterion satisfied:
  - Therefore, the task set is schedulable.

13

The first thing we check is Liu Layland criterion we compute the utilization due to various task and for the first task it is 20 by 100, 30 by 150 for the second 60 by 200 for the third task and we get 0.7 and that is the total utilization due to the tasks. And we need to check whether the Liu Layland bound is satisfied worth these and the Liu Layland bound for 3 tasks is 3 into 3 to the power 1 by 3 minus 1. So, that is n into 2 to the power 1 by n minus 1.

So, if you simplify you will get the Liu Layland bound to be 0.78 and the utilization of the tasks set is less than 0.78 and therefore, straight away we can say that the tasks set will meet it is deadline the Liu Layland criterion is satisfied and the tasks set is schedulable that we can infer.

(Refer Slide Time: 08:44)



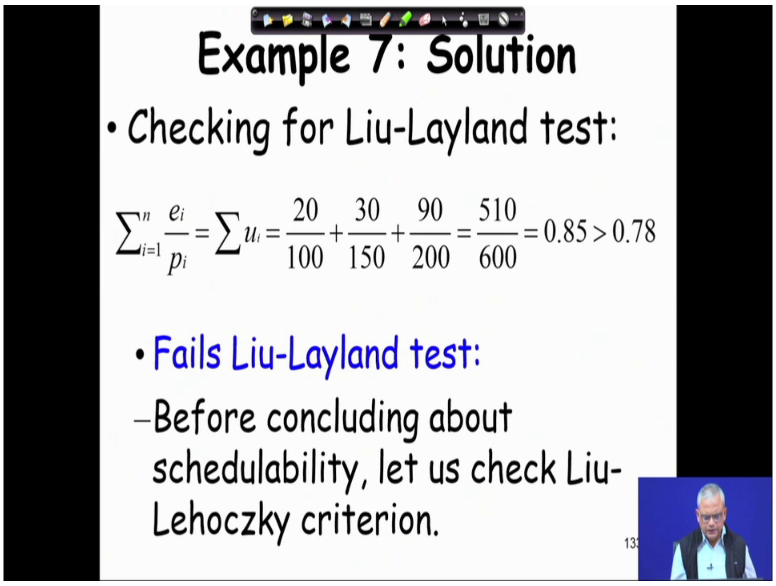
Example 7

- Consider three periodic tasks
  - T1: e1=20mSec, p1=100mSec
  - T2: e2=30mSec, p2=150mSec
  - T3: e3=90mSec, p3=200mSec
- Check whether the tasks are RMA schedulable.

13

Now, let us look at a different tasks set where the first is 20 millisecond execution, 100 millisecond is the period, second task thirty millisecond is the execution time 150 millisecond is the period and the third task is ninety millisecond is the execution time and 2 and ok.

(Refer Slide Time: 09:23)



Example 7: Solution

- Checking for Liu-Layland test:

$$\sum_{i=1}^{n} \frac{e_i}{p_i} = \sum u_i = \frac{20}{100} + \frac{30}{150} + \frac{90}{200} = \frac{510}{600} = 0.85 > 0.78$$

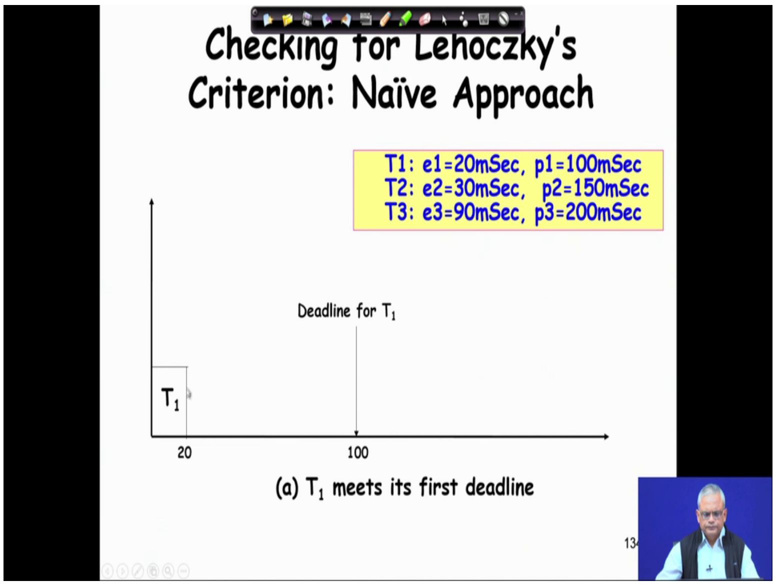
- Fails Liu-Layland test:
  - Before concluding about schedulability, let us check Liu-Lehoczky criterion.

13

So, let us check the utilization. So, 20 millisecond out of period is hundred second task is 30 millisecond period is a execution time 150 and the third task is 90 millisecond and the period is 200.

So, the utilization is 510 by 600 which is 0.85 and the Liu Layland bound for 3 task is 0.78 and this exceeds the Liu Layland bound and therefore, it fails the Liu Layland test. So, according to Liu Layland it may not be schedulable, but then we know that Liu Layland is a pessimistic result we need to check the completion time theorem of the Liu and lehoczky to say that whether it is schedulable or not.

(Refer Slide Time: 10:30)



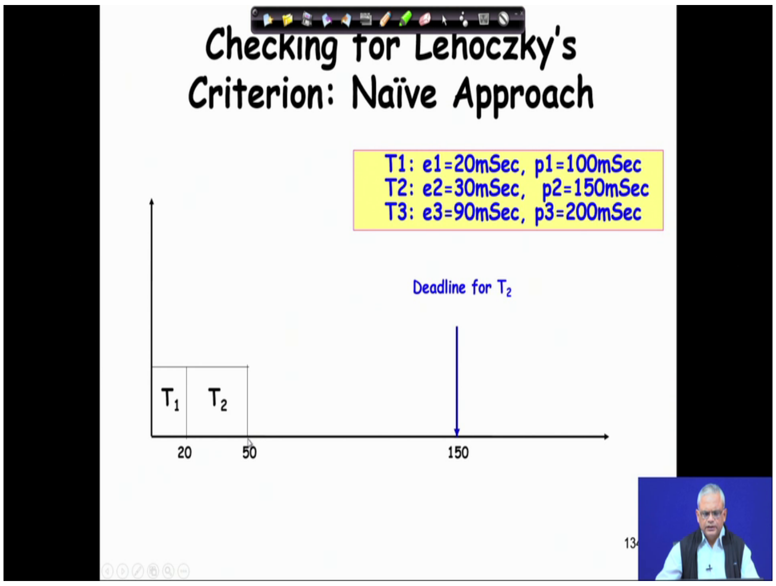
Checking for Lehoczky's Criterion: Naïve Approach

T1: e1=20mSec, p1=100mSec
T2: e2=30mSec, p2=150mSec
T3: e3=90mSec, p3=200mSec

(a) T_1 meets its first deadline

The first thing is let us look at the naive approach where we graphically plot whether the tasks all the tasks meet their first deadline under 0 phasing conditions and if all the tasks meet their first deadline under 0 phasing then the tasks set is schedulable. So, let us start with task T 1, T 1 is the highest priority task and even if it occurs with T 2 T 3 T 1 will run and it complete by 20 whereas, the deadline is 100 therefore, T 1 meets it is first deadline. Now let us try for T 2 T 2 takes 30 millisecond and 150 millisecond is the period. So, the first instance of T 2 must complete before, 150 millisecond and since T 1 T 2 T 3 T 1 T 2 occurs together first T 1 will run.

(Refer Slide Time: 11:49)



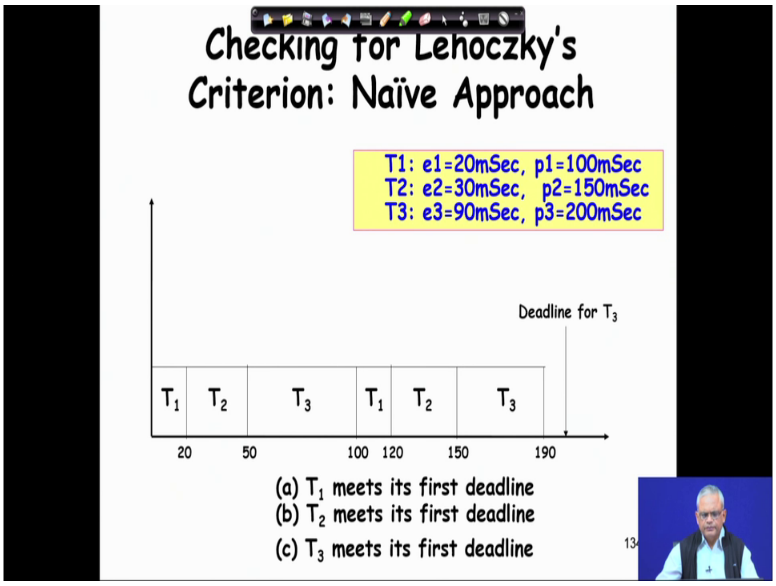
Checking for Lehoczky's Criterion: Naïve Approach

T1: e1=20mSec, p1=100mSec
T2: e2=30mSec, p2=150mSec
T3: e3=90mSec, p3=200mSec

And has T 1 completes T 2 will be taken up and T 2 will run up to 50 millisecond and by that time it will complete, but it is deadline is 150. So, well within the deadline T 2 completes because the next instance of T 1 will occur only at 100, now let us look at the third task requires 90 millisecond and the deadline and the period both are 200 millisecond.

Now, let us again graphically plot. So, T 1 T 2 T 3 arrive together T 1 is the highest priority that runs for 20 and then T 2 runs for 30 and at this point T 3 can start to run.

(Refer Slide Time: 12:46)



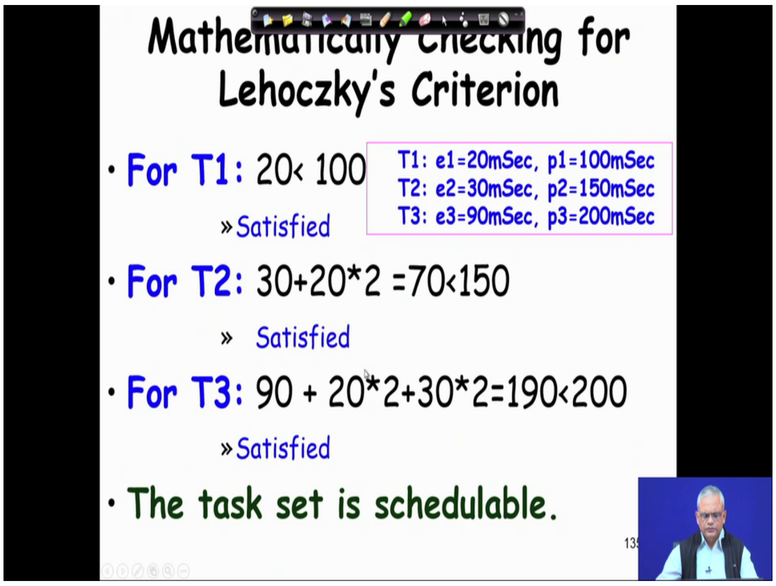
Checking for Lehoczky's Criterion: Naïve Approach

T1: e1=20mSec, p1=100mSec
T2: e2=30mSec, p2=150mSec
T3: e3=90mSec, p3=200mSec

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

So, T 3 can run up to 100 because at 100 T 1 would again the T 1 instance will arise. So, by that time T 3 would have completed 50 millisecond of execution and after T 1 completes again T 2 can execute and finally, T 3 executes and completes, but that is well within it is deadline. So, the tasks set meets it is deadline we can graphically plot and check both T 1 T 2 and T 3 all of them meet their first deadline now let us do it mathematically.

(Refer Slide Time: 13:43)



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: e1=20mSec, p1=100mSec
T2: e2=30mSec, p2=150mSec
T3: e3=90mSec, p3=200mSec

So, again we have the 3 tasks 20 100 30 150 90 and 200. So, first let us check the Liu Lehoczkys condition for the first task T 1 is the highest priority task and it will not be preempted by any other task and therefore, it will run for 20 and it will meet it is deadline because the dead line is 100. So, 20 is less than 100 and this ensures that T 1 would meet it is deadline.

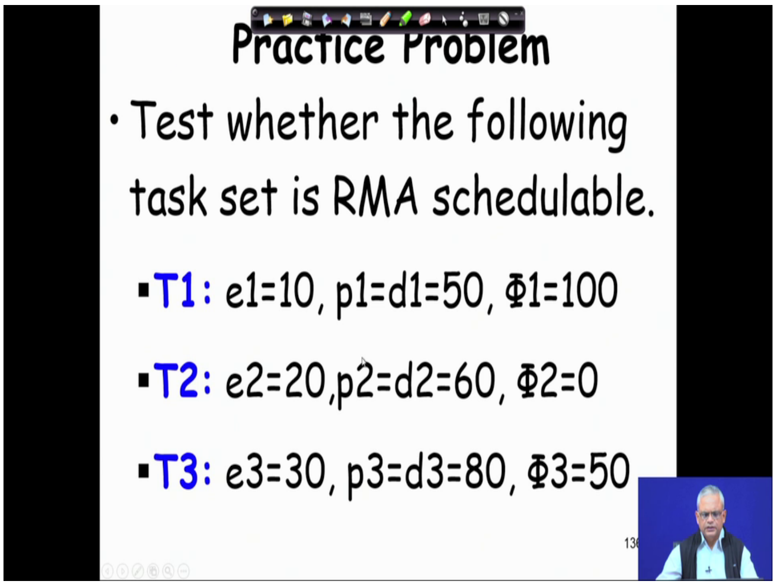
Now, let us check for T 2 T 2 would need 30 millisecond and T 1 is it is higher priority task there is only 1 higher priority task and that can arise 2 times, because 150 by 100 ceiling is 2 and each time that T 1 task runs it will take 20. So, we have written 20 into 2 and that gives us 70 and; that means, that T 2 complete execution by 70 millisecond whereas, the period is 150. So, even in this case also the Liu Lehoczkys criterion is satisfied. Now let us look at T 3 T 3 requires 90 millisecond execution time and it is period is 200 and the task T 1 the highest priority task can occur twice, because 200 by

100 ceiling is equal to 2. So, 20 into 2 40 millisecond will be taken up by the highest priority task.

Now the second highest priority task is T 2 it will also occur 2 times because 200 by 150 ceiling is 2 and each time T 3 occurs it will execute for 30 millisecond. So, the total works out to be 190,which is same as got in the graphical sketching, but here these are simple expression we can easily do it quickly without any mistake whereas, the graphical 1 is prone to mistake and also takes lot of time.

So, from now onwards all our examples we will use the mathematical expression directly the graphical solution helps us understand how the Liu Lehoczkys criterion works the main concepts behind the Liu Lehoczkys criterion, but to really work out the solution we will use the mathematical expression. So, from the Liu Lehoczkys criterion we can see that the tasks set is schedulable.

(Refer Slide Time: 16:55)

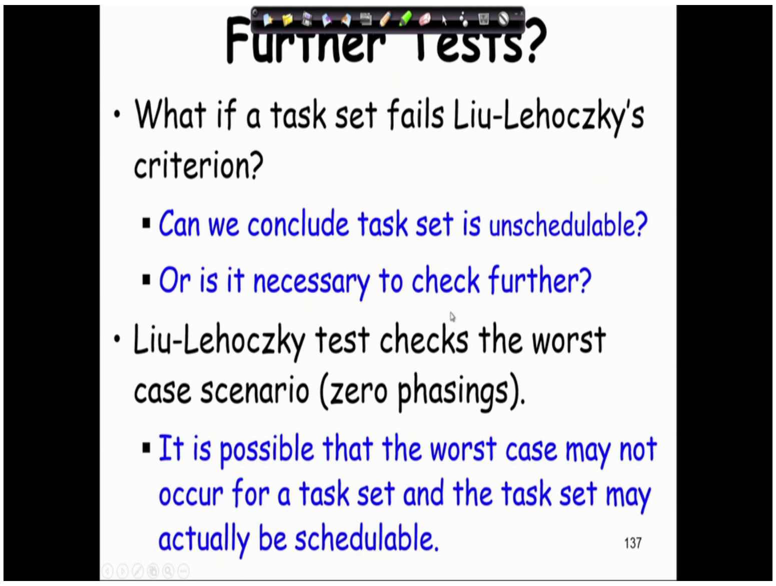


Practice Problem

- Test whether the following task set is RMA schedulable.
  - **T1:** e1=10, p1=d1=50, Φ1=100
  - **T2:** e2=20,p2=d2=60, Φ2=0
  - **T3:** e3=30, p3=d3=80, Φ3=50

Now, there is a practice problem please do it on your own we have 3 tasks T 1 the execution time is 10 the period and deadline both equal to 50 and the phase is 100 millisecond all are in milliseconds, the task T 2 execution time is 20 the period and deadline in 60 and the phase is 0. And task T 3 the execution time is 30 the period and deadline both are 80 and the phase is 50. So, please try this check the Liu Layland condition whether it is schedulable and if it is not schedulable according to Liu Layland then before concluding that the tasks set is not schedulable we need to try the Liu and Lehoczkys criterion.

(Refer Slide Time: 18:01)



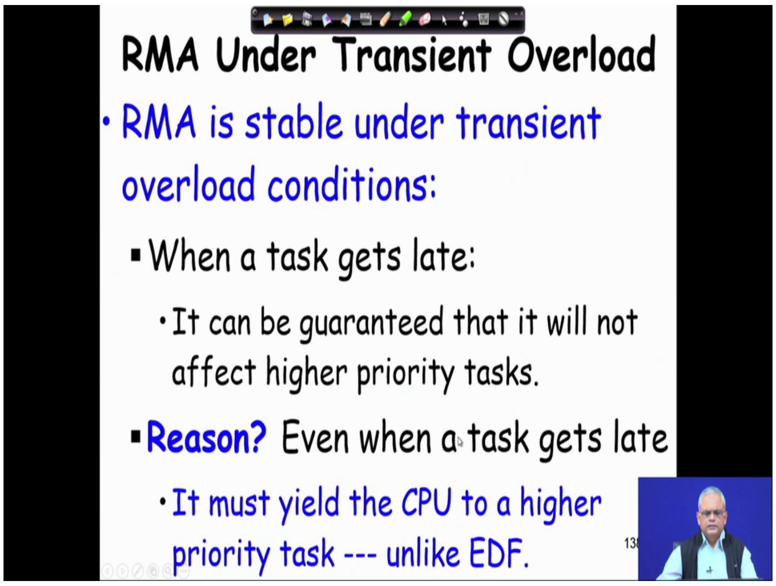
Further Tests?

- What if a task set fails Liu-Lehoczky's criterion?
  - Can we conclude task set is unschedulable?
  - Or is it necessary to check further?
- Liu-Lehoczky test checks the worst case scenario (zero phasings).
  - It is possible that the worst case may not occur for a task set and the task set may actually be schedulable.

137

But just asking this question that suppose a tasks set fails the Liu Laylands criterion and also Liu and Lehoczkys creiterion, then can we safely conclude that the task set is unschedulable or is there a chance that the tasks set is schedulable. The answer to this is that even when a tasks set fails Liu Lehoczkys criterion still it may be possible that it may meet it is deadline, the main reason behind this is that in Liu Lehoczkys criterion we check the worst case condition where a task arrives in phase with all it is higher priority tasks, but then for a tasks set it may not occur the 0 phasing of a tasks with all it is higher priority task may not occur and that is the reason, that it may fail the Liu Lehoczkys criterion, but then it may still meet it is task deadlines, but then these cases are very less where the tasks set fails Liu Lehoczkys test and still meet it is deadlines.

(Refer Slide Time: 19:28)



RMA Under Transient Overload

- RMA is stable under transient overload conditions:
  - When a task gets late:
    - It can be guaranteed that it will not affect higher priority tasks.
  - Reason? Even when a task gets late
    - It must yield the CPU to a higher priority task --- unlike EDF.

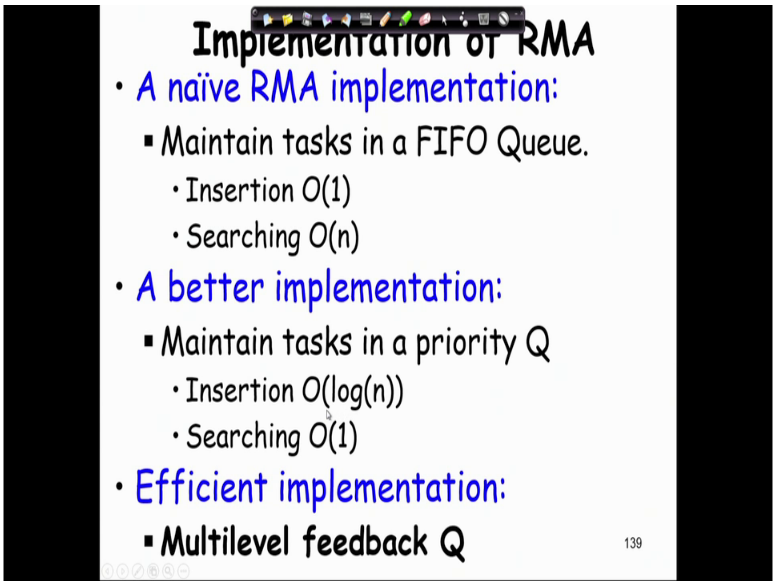
Now, let us look at some issues in rate monotonic scheduling, the first thing that we want to check is about a very severe thing that we talked about in the in the EDF scheduling is the transient overload. The transient overload if you remember we had said that sometimes very low priority task like task result logging or some health monitoring checking etcetera these are low priority tasks.

And if some of these tasks get delayed due to some reason than that may affect the highest priority most critical task. If a lowest priority cannot affect the higher priority tasks, then we say that system is stable under transient overload, but if the lowest priority task when it gets delayed can also delay the highest priority task then we say that the system is unstable poor transient overload handling capability which was true in case of EDF. Now let us examine for the rate monotonic scheduling.

So, here we make a statement here that the rate monotonic scheduling is stable under transient overload conditions the rate monotonic algorithm is stable under transient overload conditions, but then how do we argue how do we show that it is really stable can you think of something that will help us convince that the rate monotonic scheduler the lowest priority task cannot make the highest priority task to miss it is deadline. The answer is very simple, because here in the basic rate monotonic principle is that a higher priority task will be taken up for execution when it is ready even when a lower priority task is not completed.

So, it preempts the lower priority task there is no chance that a lower priority task will continue to run when a higher priority task arrives and that conclusively shows that rate monotonic scheduling is stable under transient overload, because whenever a higher priority task arrives even when a lower priority task has not completed that will be context switched and the higher priority task will run. So, the lower priority task must yield the CPU to the highest priority task to the higher priority task and this is contrary to what used to happen in EDF and therefore, this is a very positive feature of the rate monotonic scheduling and it is course over the EDF algorithm as far as the transient overload handling is concerned.

(Refer Slide Time: 23:06)



Implementation of RMA

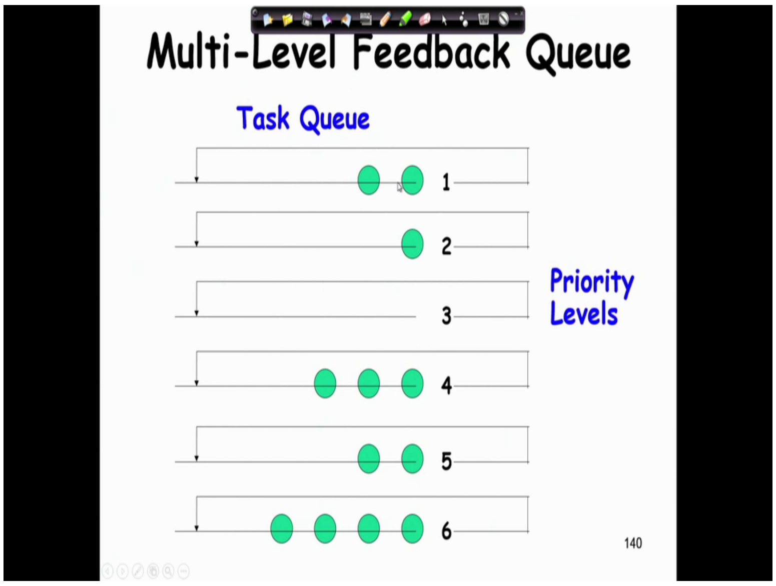
- A naïve RMA implementation:
  - Maintain tasks in a FIFO Queue.
    - Insertion O(1)
    - Searching O(n)
- A better implementation:
  - Maintain tasks in a priority Q
    - Insertion O(log(n))
    - Searching O(1)
- Efficient implementation:
  - **Multilevel feedback Q**

139

The next thing we want to discuss is that how is the rate monotonic scheduler implemented? A very simple implementation without much thought can be to keep all the tasks in a queue. So, in the queue as the tasks arrive we keep it at the queue and each time either a task arrival or completion event occurs, we need to check through the queue to see which task has a highest priority of the all the waiting tasks in the queue which task has the higher priority, but if we analyse this situation then insertion when a task arrives we keep it at the end of the queue and therefore, insertion time is one, but then on each scheduling point that is on a tasks arrival or completion we need to check the entire queue to find which has the highest priority among the waiting tasks and therefore, this is O n.

If there are n tasks waiting in the queue not a very efficient solution, but what about a priority queue a priority queue can be implemented based on a heap I think you know that a heap can be implemented on array and we use a array data structure to have a heap and in the heap the insertion is log n, if you look at the heap algorithm inserting an element into a heap is O log n, whereas the highest priority task can be obtained easily and that is O 1, but can see that this is a better solution than a linked list more efficient, but then still it is not good enough because at each scheduling point we need to if a task arrives we need to insert that task in the heap which is log n, can we do better yes we can use a multi-level feedback queue and this is 1 which is actually used in the operating system implementations of a rate monotonic scheduler.

(Refer Slide Time: 26:13)



The idea here is that we allow only a fixed set of priority a little later after a few classes we look at the po6 real time standard and will see that the number of priority levels that are available is typically 16 and in some cases we will see that it is 32. So, if we have a fixed number of priority levels then we have a queue here. So, priority 1 if it is assigned to a task each time that task arises we insert it in the queue and after it completes we keep it waiting and until again the task period is over we insert a phrase instance here and that we do for all task priority levels.

So, if you check here the insertion is O 1, but what about selecting the task on a scheduling point the selection is also O 1 because we need to check from the highest priority task and check if there is a task in any of these queues.

So, if there is any task in a queue starting from one the scheduler look through and in whichever queue it finds that there is a task waiting at the front of the queue it just takes it and that is O 1 and since there is a limited number of task levels and therefore, it just needs to look for up to certain limit and therefore, both insertion into the task queue and also getting a task from this both are O 1 a very efficient implementation and therefore, as we discuss the real time operating system implementation we will see that the rate monotonic scheduler is typically implemented with a multi-level queue and all real time operating systems.

They support real time tasks scheduling through the rate monotony scheduler by providing a fixed set of priorities at this point getting closed to the time we will stop here.

Thank you.