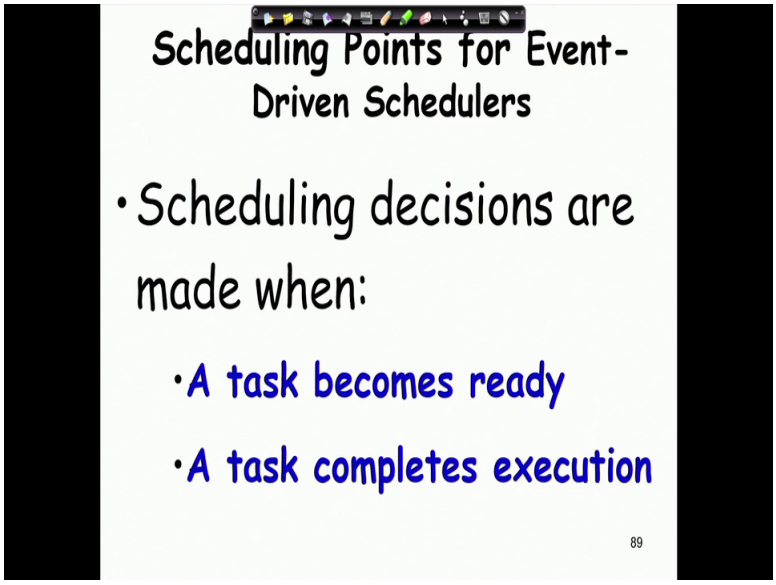
Real Time Operating System
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 07
Event – Driven Schedulers

Welcome to this lecture. So far, we had looked at the clock driven schedulers which are efficient used in very simple applications. Many small embedded applications use schedulers like cyclic schedulers table driven schedulers, but more sophisticated applications use event driven schedulers. We had seen a very simple event driven scheduler the foreground background scheduler in the last lecture. Now let us continue from that point onwards.

(Refer Slide Time: 00:57)

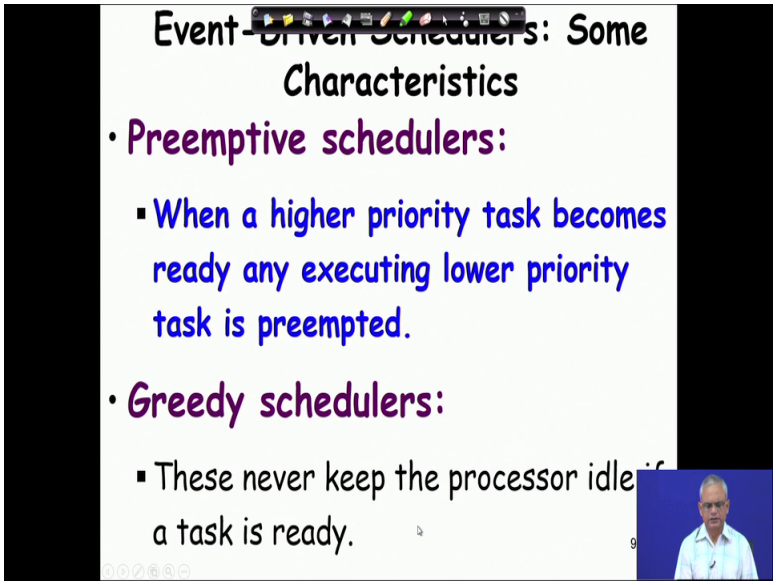


We had discussed that the scheduling points for event driven schedulers are when the task arrives or a task completes because these two events of concern either when a task instance becomes ready or a task instance completes execution then the scheduler wakes up the code for the scheduler start running to decide which task to run next. So, that is a very basic principle of this event driven schedulers.

Now let us look further; there are many types of event driven schedulers and these are the one which are extensively used in non trivial applications these are called as the preemptive schedulers because for a higher priority task these schedulers pre-empt, the

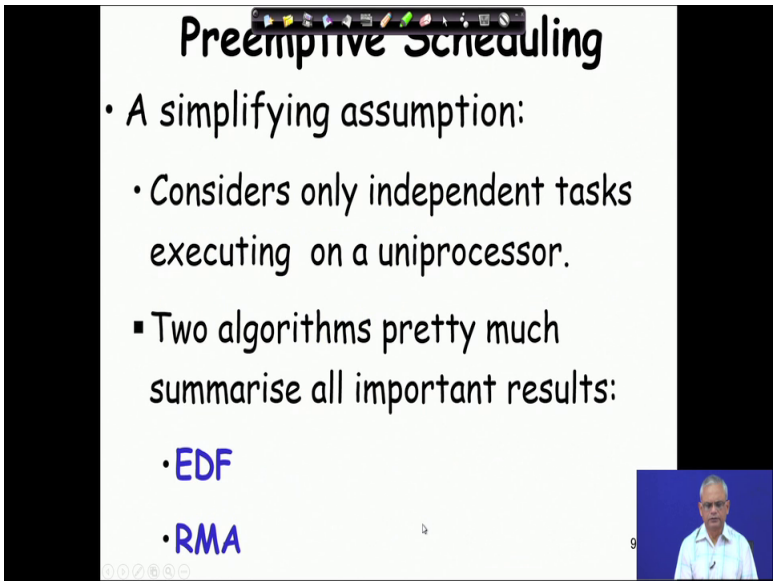
lower priority task the cyclic schedulers are non preemptive in the sense that once a frame is assigned to a task it will continue to run in that frame whereas, here depending on which task instance gets ready the executing task instance may get preempted. So, the operating system is much more sophisticated here it should be able to preempt a task, these are also called as Greedy schedulers.

(Refer Slide Time: 02:51)



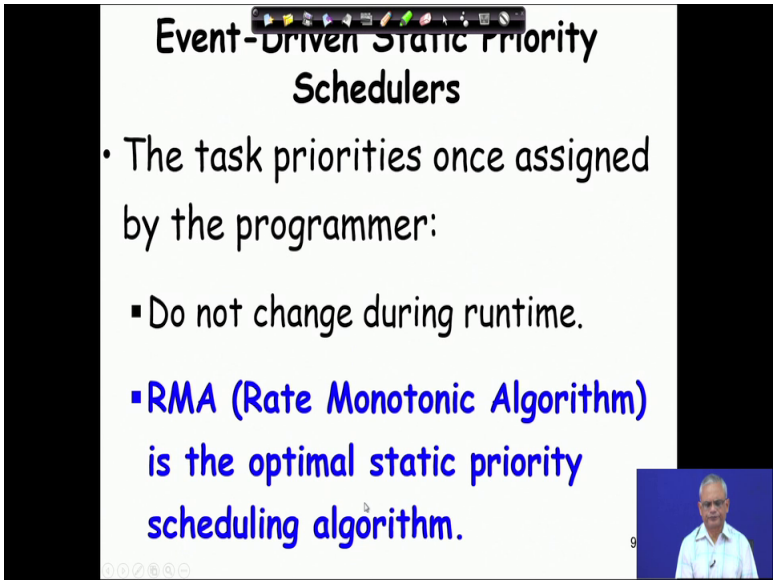
These are greedy schedulers because whenever a task completes it takes up the task that are waiting, it never tries to leave any time this processor unutilised. So, the processor is made never idol, if there is a task that is ready that is why these are called as a greedy class of schedulers.

(Refer Slide Time: 03:27)



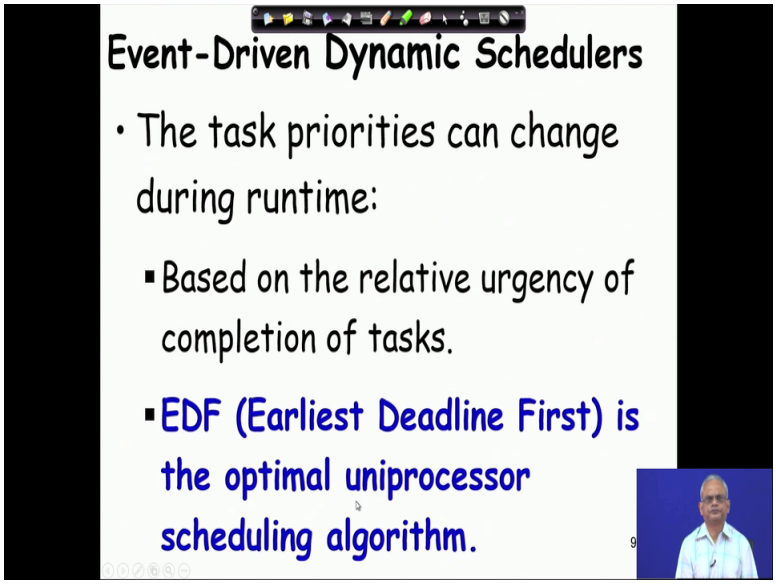
As I was saying that a large number of event driven schedulers are available, but then if we look at all those we find that those are basic variations of two main types of schedulers one goes by the name earliest deadline first and the other is rate monotonic analysis EDF; earliest deadline first and rate monotonic analysis. These are the two major types of schedulers and all others hundreds of other event driven schedulers are minor variations of these, if we understood these 2 schedulers, I think we know a good deal about event driven schedulers. So, let us look at these 2 types of schedulers carefully.

(Refer Slide Time: 04:26)



The rate monotonic algorithm is example of a static priority scheduler a static priority scheduler is a one where the designer assigns priority to tasks during his design and once the task priority they are assigned by the programmer. These do not change during runtime and in this class of schedulers the static priority schedulers the rate monotonic algorithm is the optimal static priority scheduling algorithm we will see that there are various of this algorithm, but rate monotonic is a optimal algorithm. It can run tasks; tasks set which are cannot be run by the other algorithms we will see the detail results as we proceed.

(Refer Slide Time: 05:38)

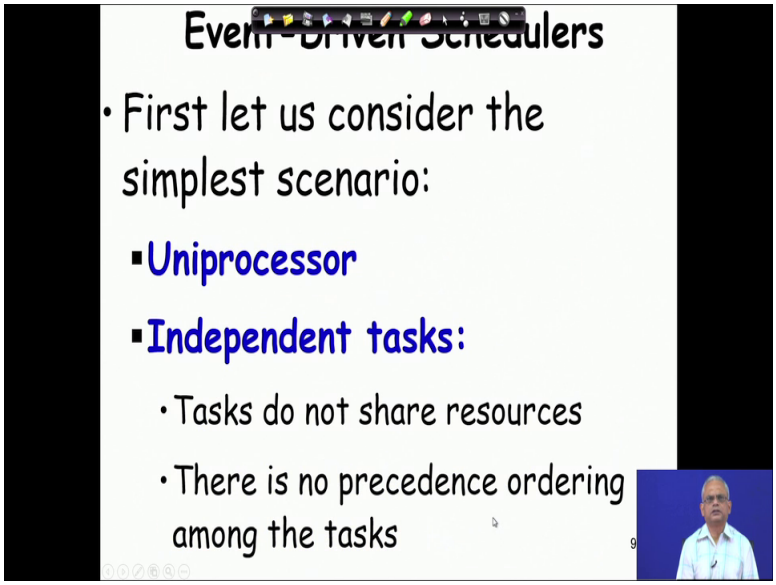


The other category of event driven scheduler are the dynamic scheduler or the dynamic priority scheduler here the priority is not held constant, but depending on the situation the priority of a task may be changed by the scheduler one reason why the priority of task may be changed by the scheduler is because of the urgency of the task completion. So, one task may be waiting for long time and it is about to miss its deadline then the scheduler will increase its deadline sorry increase its priority. So, that it will be able to meet its deadline. So, these are the dynamic scheduler where the scheduler keeps on changing the priority of the tasks and of all the dynamic priority schedulers the EDF or the earliest deadline first scheduler is the optimal uniprocessor scheduling algorithm.

So, what it means is that given a tasks set if the EDF using EDF a schedule cannot be worked out the EDF cannot schedule it then no other uniprocessor scheduling algorithm

can run it whether it is dynamic priority or static priority it is the optimal uniprocessor scheduling algorithm including both static priority and dynamic priority scheduling algorithm EDF is the optimal uniprocessor scheduling algorithm, if a task set cannot be run using EDF, it cannot be run using any other schedulers. Now let us look at further into the results.

(Refer Slide Time: 07:50)

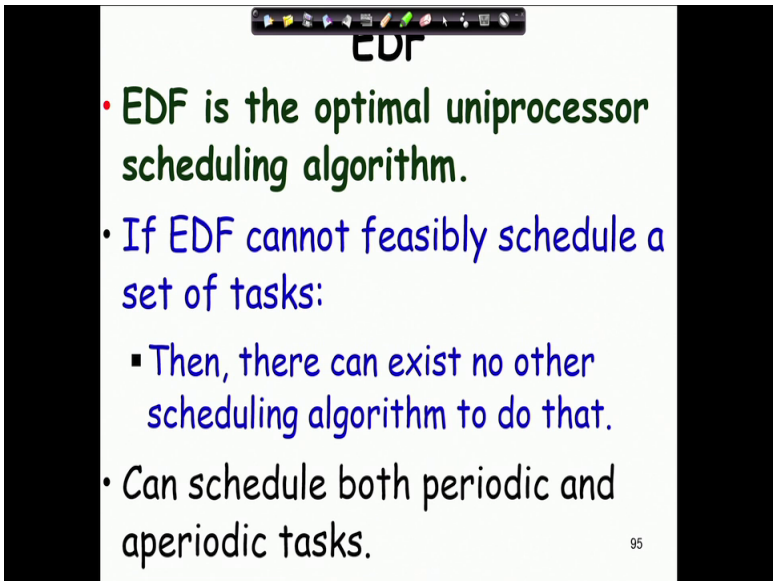


To start with a our discussion, we will consider the simplest scenario, the simplest scenario is that we have a uniprocessor and the tasks set is independent, we say the task set is independent is that the tasks do not share resources that is result set by one task is not used by the other tasks and also there is no precedence ordering among tasks like one task always need to run after that the other task and run, etcetera, those situations do not exists.

So, this is the simplest situation uniprocessor with independent tasks, but of course, may not be very practical because many applications you might have multiprocessor being used or we might have tasks which share results with each other or there may be precedence ordering, but for our understanding, let us start with a simplest one and as we proceed, we will relax these constraints that tasks share resources and what needs to be done to the scheduler precedence ordering what needs to be done to the schedulers and multi processor what kind of changes are needed.

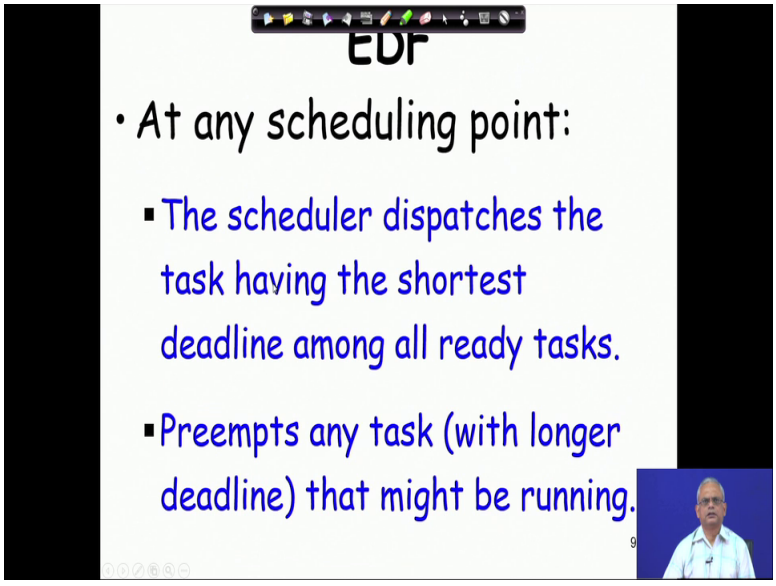
So, as we proceed we will look at those situations, but we start with very simple.

(Refer Slide Time: 09:26)



First let us look at the EDF. So, if EDF cannot run a set of tasks it not necessary to look for another scheduler which can run this tasks because there will exists no scheduler which can run it, EDF is a important scheduler it is optimal first of all any complex tasks set with challenging deadlines, EDF can find a feasible schedule for that not only that it can schedule both periodic and aperiodic tasks at any scheduling point the scheduler dispatches the shortest deadline ready task.

(Refer Slide Time: 10:14)

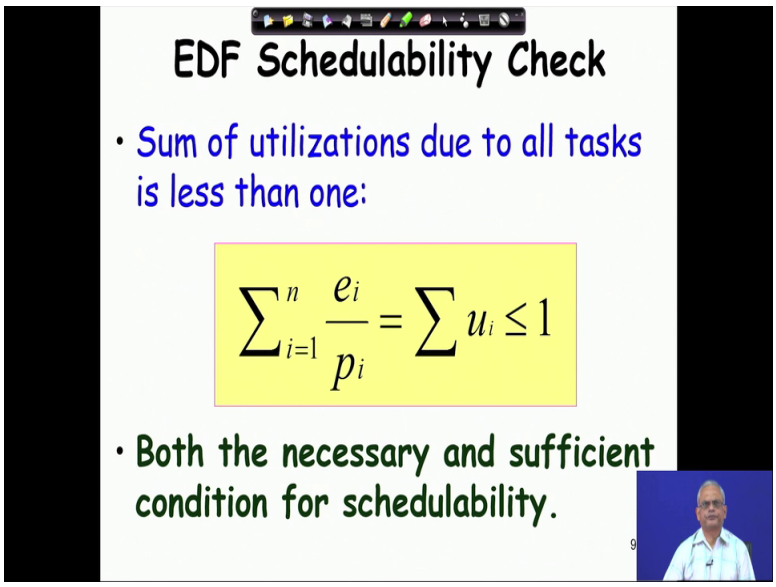


So, this is the basic of the EDF whenever the scheduler starts running it looks at all the tasks that are waiting finds out the deadline of the tasks and find out finds out which are the shortest deadline and then it starts running that. So, this is the crux of the algorithm and the name of the algorithm comes from here earliest deadline first. So, every scheduling point it looks at the tasks which has the earliest deadline and it takes up first to schedule.

This I hope this point is clear why it is called as earliest deadline first scheduler because every scheduling point the scheduler starts running and then it checks all the tasks that are ready finds out the task that has the earliest deadline and then starts to run it and when it runs, it may preempt any other task that may be running you may ask that that at a scheduling point will there be a task running or let me ask you this question that when there is a scheduling point, is it possible that some task is running at a scheduling point.

The answer is yes there are two types of scheduling points one is on task completion of course, on task completion at that task scheduling know the task will be running, but the other type of scheduling point arises due to task arrival and when a periodic task instance arrives it is very likely that some other task might be running at that point and then the scheduler will check the relative deadline of the 2 tasks and then if the one that has arrived has a shorter deadline then it will preempt the running task.

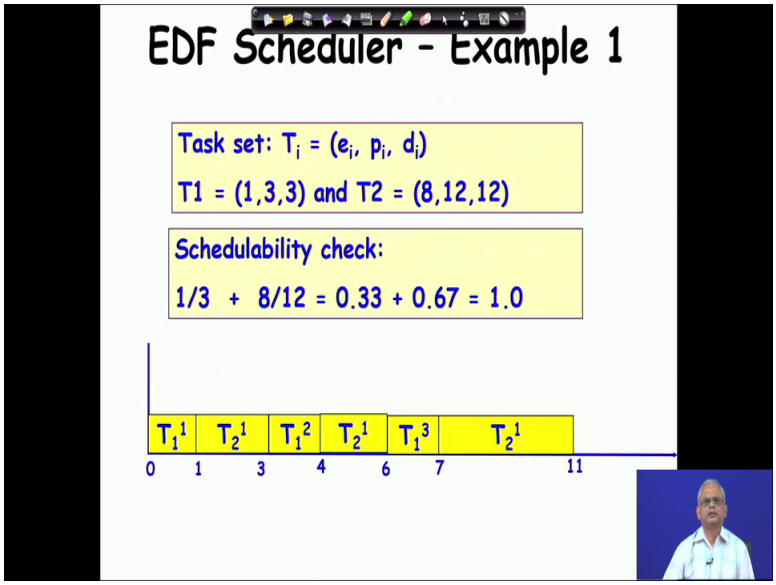
(Refer Slide Time: 12:57)



An important issue that we need to check understand well and also be able to check is the schedulability; the schedulability of a task set means that whether EDF can run it, the schedulability expression for EDF is very simple. It is sigma e i by p I, if there are n tasks 1 to n, then sigma i equal to 1 to n e i by p i e i by p i as we had earlier also mentioned, it is basically the utilization of that task it takes e i execution in over a p i duration and therefore, the utilization due to the task e i sorry, t i is e i by p i we represent that using e y.

E y is the utilization due to the task t i when given by e i by p i and for all tasks, you can sum the utilization due to various tasks as long as the utilization is less than equal to 1, we say that the tasks set is schedulable or it can satisfactorily run on a earliest deadline first scheduler this expression is both the necessary and sufficient condition for schedulability for the EDF schedulers.

(Refer Slide Time: 14:43)



Let us look at an example, we have 2 tasks their execution time period and deadline are given as one unit the period is 3 and the deadline is 3 for task 2 the execution time is 8 the period and deadline are 12. So, will task set be feasibly run using a EDF scheduler the answer is obtained by using the expression sigma e y is less than equal to 1 the utilization due to task one is one by 3. So, every 3 units it needs one unit of execution the utilization due to task 2 is 8 by 12 which is point six seven and sigma e y is equal to 1

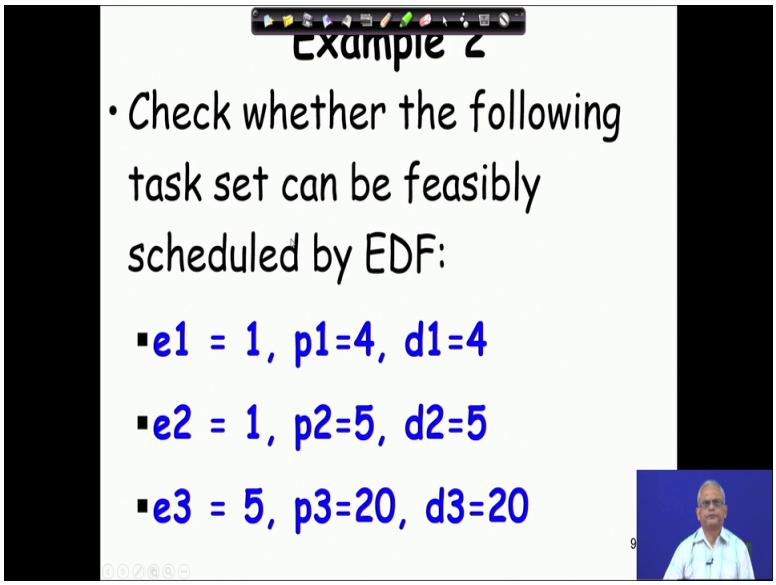
which is less than equal to 1 and therefore, this task set is feasibly schedulable using the EDF scheduler let us see how will the task set run or how will the EDF scheduler run.

Tis task set let us assume that both the tasks T 1, T 2 the task instance start arriving at time zero. So, time zero both tasks arrive and the scheduler will wake up because that is a scheduling point now which task will it take for scheduling it will take the task with a earliest deadline. So, between T 1 and T 2 which has the earliest deadline the one has deadline 3 and the other has deadline 12. So, it will take the task T 1 the first instance of task T 1 which has the earliest deadline it will take the up for execution.

It completes after one unit because execution time of T 1 is one and at that point defines a scheduling point and the scheduler will wake up and find that there only one ready task which is T 2 1 it will start executing T 2 1 the T 2 1 executes for 2 time units and at time 2 sorry time instance 3 the second instance of t ones arrives and then checks the deadline and finds that T 1 has lower deadline then T 2 and therefore, preempts T 2 and T 1 starts running the second instance of T 1 it completes after one time instance and then it resumes the T 2 1 the T 2 1 runs for 2 units because the total requirement for T 2 1 is 8 units at the time instance 6. The third instance of T 1 arrives and because it has a shorter deadline then the T 2 it again preempts T 2 T 1 starts running and then T 2 starts running.

So, this is the basic idea behind the EDF scheduler at every scheduling point it starts to run and then decides the task which has the shorter deadline and takes it up for execution.

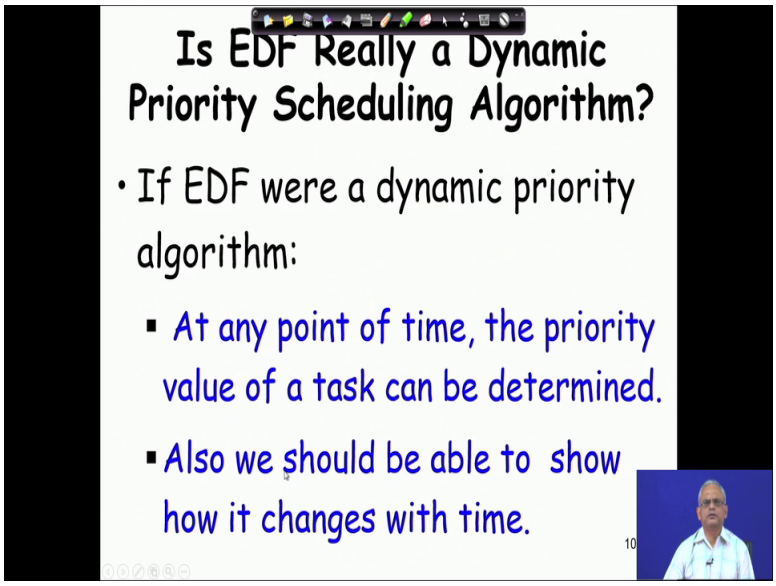
(Refer Slide Time: 19:23)



So, let us do one example we have a task set 3 tasks T 1 has execution time of one period one deadline sorry period four deadline 4 task 2 execution time one period 5, deadline 5, task 3, execution time 5, period 20 and deadline 20. This is normally the task set in a real time application and their period and deadline are same worst majority of the task are periodic and their period and deadline are the same and the execution time fraction of the period.

Now, we want to check whether this is EDF schedulable. So, you need to use the expression that we have sigma e y is less than equal to 1 the utilization due to T 1 is 1 by 4 utilization due to T 2 is 1 by 5 and utilization due to T 3 is 5 by 20 which is 1 by 4. So, you can sum that up.

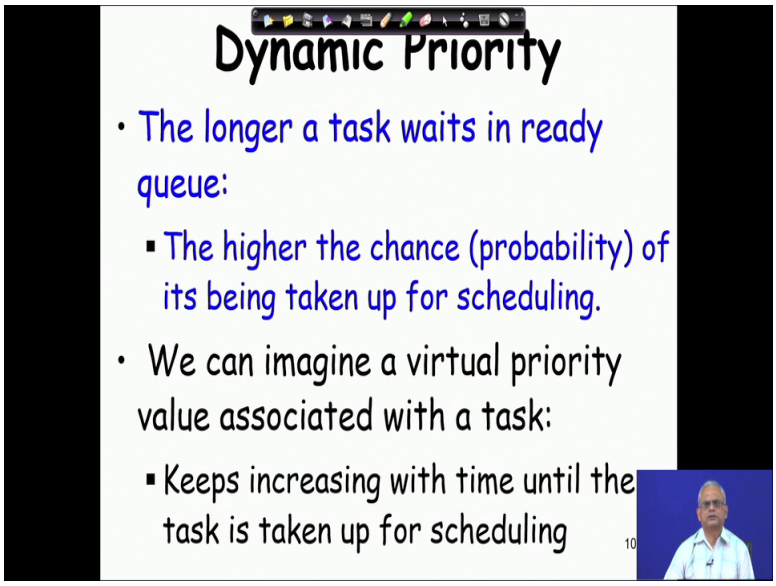
(Refer Slide Time: 20:50)



And check whether this is schedulable or not now let us try to answer a very fundamental question that we have said that there are 2 classes of algorithm event driven schedulers one is the static schedulers the other is dynamic priority schedulers and we said that EDF is a dynamic priority scheduler, but then if EDF is dynamic priority scheduler, then we should have a concept of a priority and we should be able to determine what is a priority of a task, we should be able to check whether the priority of a task that is running is greater than the arriving task or the arriving task has a higher priority, but so far if you look at the discussion that we have never mentioned anything with priority we just said that the scheduler examines the task having the earliest deadline and then schedules it.

So, what is the notion of priority here and why do we call it as a dynamic priority scheduling algorithm. So, not only you should be able to calculate the priority of a task, but also you should change the we should show that it changes with time then only we can call it as a dynamic priority algorithm, but we have never said anything about some priority how does priority change with time. So, how do we really call it as dynamic priority scheduling algorithm.

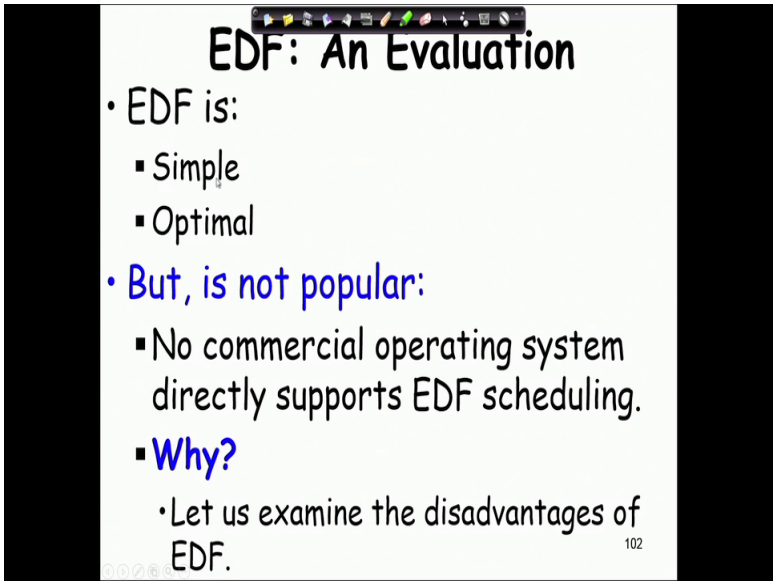
(Refer Slide Time: 22:39)



Let us see the answer the answer is that as the task waits because there are other high priority algorithms or having shorter deadlines their taken out, but as the task waits it chances up being taken up increases.

So, you can imagine a virtual priority value associated with a task it keeps increasing with time until the task is taken up for scheduling.

(Refer Slide Time: 23:19)

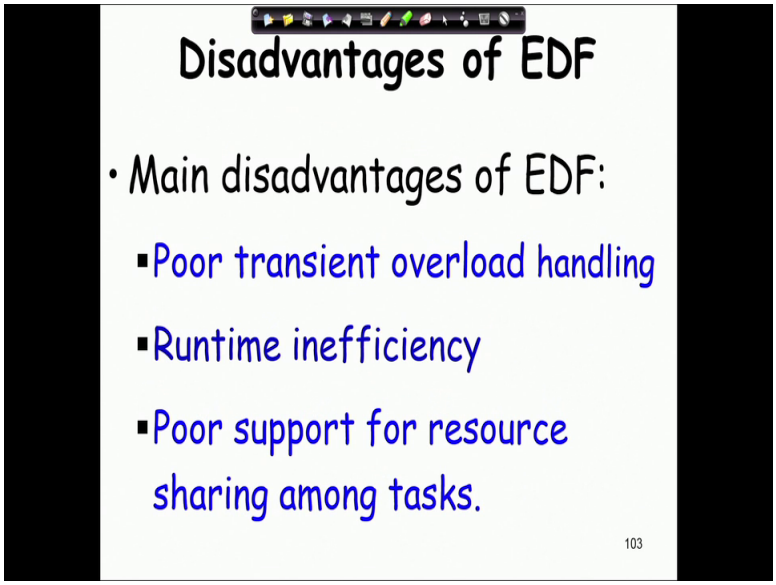


EDF; we saw that it is a optimal algorithm there can be no other algorithm which can schedule a set of tasks if EDF cannot; it is a very simple algorithm at every scheduling

point just check which has a shortest deadline and take that out for execution, but it is not popular its rarely used in applications. In fact, as we say look at the commercial operating system none of the operating system directly supports EDF you cannot select EDF scheduler if at all we have to our self-implement.

EDF scheduler using the support available by the operating system, but why if it is a such a good algorithm, its efficient simple why is it that none of the applications and very few applications use it, it must be something to do with the disadvantages of this EDF, it must have some very severe disadvantages that is why it is not used, the answer to this that why it is not used will become clear if we look at the disadvantages of the EDF scheduler.

(Refer Slide Time: 24:49)



The main disadvantages of the scheduler the first disadvantage goes by the name poor transient overload handling. So, what it means is that if due to some reasons a task takes more time to execute may be it took a longer path in a code maybe it was waiting for some condition which did not occur may be it went into an infinite loop in those cases of transient overload it is not known which task will miss the deadline even the most important task can miss a deadline because a very low priority task it just got delayed.

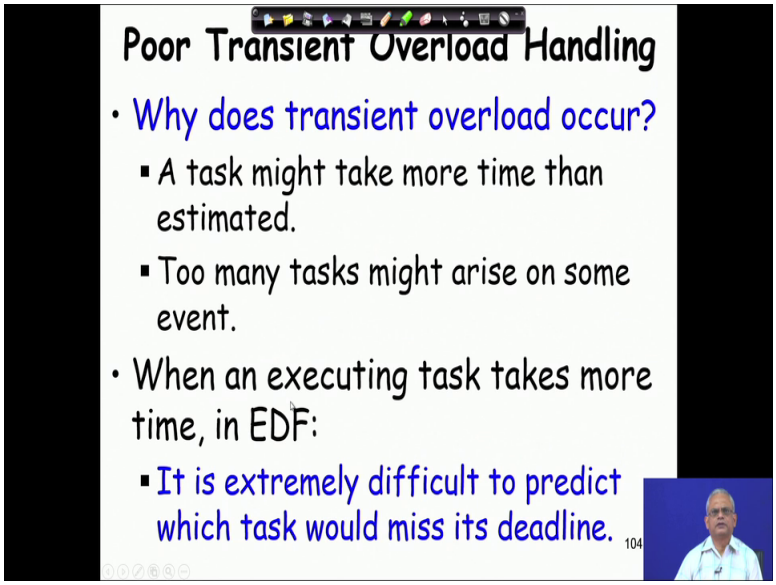
Implementation of the EDF is a problem because you see here all the tasks have to be kept if there are n tasks the n tasks are to be maintained and thus scheduler needs to compute the completion time for all the tasks to fight the earliest deadline. So, a runtime

inefficiency of the algorithm we will see some efficient implementations of the EDF scheduler, but still those are not good enough the other schedulers are much more efficient.

So, the first point is poor transient overloading second is runtime inefficiency, but the third one is possibly the most glaring short coming this may be the reason that why non trivial applications do not use EDF it goes by the name; poor support for resource sharing among tasks in a realistic application the tasks do share resources the result produced by one task used by other tasks we will see that the other type of schedulers for example, the rate monotonic can be extended to handle sharing resources, but EDF will see that resource sharing if we permit in EDF it really creates lot of problems that it cannot be used.

So, the third one is possibly the most important reason why EDF is not popular, but then the other 2 reasons also are bad runtime inefficiency, it is a bad; we can have more efficient schedulers transient overload handling even if there is a minor overload in the situation in the system then the system will break down.

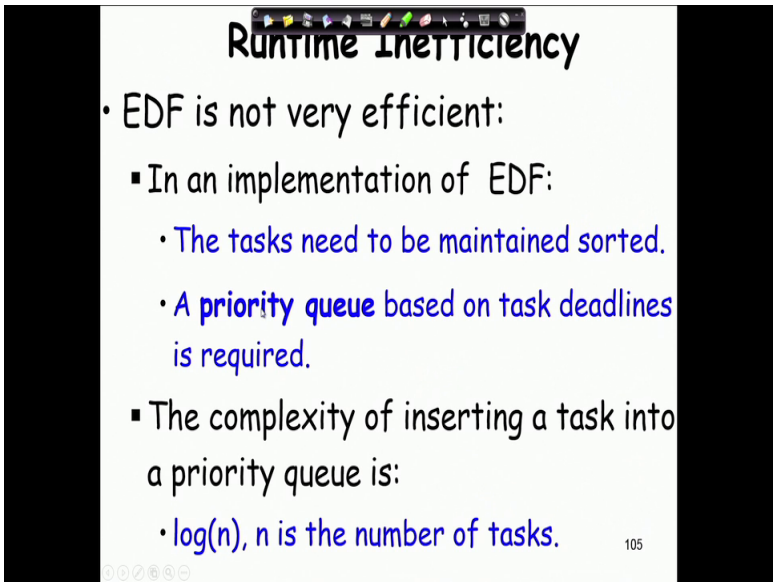
(Refer Slide Time: 27:58)



Let us understand how why transient overload occur in a system a transient overload occurs when a task takes more time than it is estimated or maybe too many tasks arise at time instant. So, when an executing task takes more time in EDF it becomes difficult to predict which task could miss its deadline? So, as I was saying that even the most

important task most critical task that may miss its deadline just because a very simple task like a logging task event logging task or a result logging task that got delayed maybe it could not write properly in the disk it was taking more time, but if we do not log events nothing happens or if we do not log nothing very severe happens log results we could have actually not want that situation that due to a event logger or a result logger the critical task missed its deadline.

(Refer Slide Time: 29:30)



But here in this scheduler that can happen that just because the logger task is taking more time the critical task missed its deadline runtime inefficiency we said that the scheduler each time needs to look at the jobs waiting to be executed a simple implementation may be a queue and then computes the deadline for each task how much time is remaining and one implementation may be based on the priority queue we will see the implementation based on the priority queue, but in this case also we will see that it is not really very efficient for this lecture we are running short of time we will just complete here saying that even in a priority queue when we want to insert a task into the priority queue we need log n time if n is the number of waiting task.

We said that the event driven schedulers are used in non trivial applications where we might have dozens of tasks and log n is not a very efficient situation of course, selecting the highest priority we can do in one time even using a priority queue our result is not very encouraging and if you just use a simple linked list or a simple queue to maintain

the task that are running that are ready then it will be o n which is still worse than this and even if we use a priority queue it is not really very encouraging.

So, let us see whether we can have a better implementation then a priority queue, but that we will look at in the next lecture.

Thank you very much.