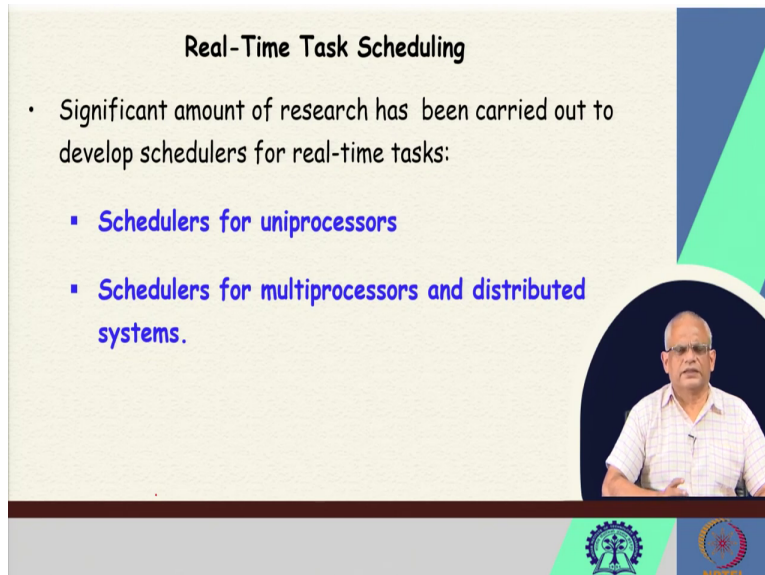


**Real Time Systems**  
**Professor Rajib Mall**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture 09**  
**Clock-Driven Schedulers**

Welcome to this lecture. In the last lecture, we had looked at some very basic concepts in real time task scheduling, we had said that the scheduler is a very important component in a real time operating system. The scheduler is the primary mechanism for meeting task deadlines. And today we will look at some basic types of schedulers and we will start with the simplest real time task scheduler. Now, let us proceed to that.

(Refer Slide Time: 0:59)



**Real-Time Task Scheduling**

- Significant amount of research has been carried out to develop schedulers for real-time tasks:
  - Schedulers for uniprocessors
  - Schedulers for multiprocessors and distributed systems.

The slide features a video inset of Professor Rajib Mall in the bottom right corner. At the bottom of the slide, there are two logos: the Indian Institute of Technology Kharagpur logo on the left and the NITRR logo on the right.

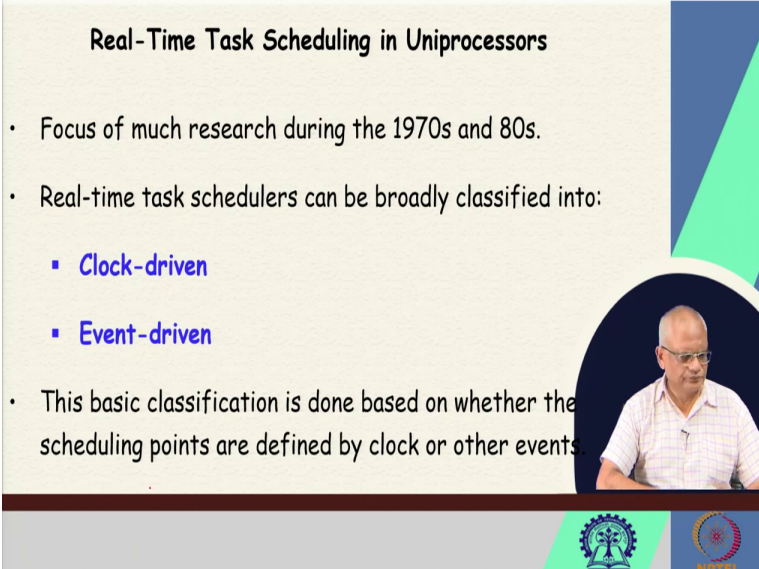
Real time task scheduling, a lot of developments have occurred in last 50 years or so, it was actually one of the most researched topic in the area of computer science to develop effective real time task schedulers. Initially, the research and

Even on a laptop or a desktop, we have multi core processors even in embedded applications. We have multi core processors, and also we have distributed systems. So, the schedulers of the, for the multiprocessors and distributed systems were developed, but the thing is that they were

developed based on the schedulers for uni processors. So, we need to thoroughly understand the schedulers for uniprocessors.

Because still many embedded processors and real time systems work on uniprocessors and also the schedulers for multiprocessors and distributed systems they are based on the schedulers for uniprocessors. Many of them actually finally, on different processors on a multiprocessor distributed system. It is only a uniprocessor task scheduler that is running, but that is for later. Let us now understand tasks scheduling for uniprocessors well.

(Refer Slide Time: 3:08)



**Real-Time Task Scheduling in Uniprocessors**

- Focus of much research during the 1970s and 80s.
- Real-time task schedulers can be broadly classified into:
  - **Clock-driven**
  - **Event-driven**
- This basic classification is done based on whether the scheduling points are defined by clock or other events.

The slide features a video inset of a man in a white shirt and glasses speaking. At the bottom, there are logos for IIT Bombay and IIT Madras.

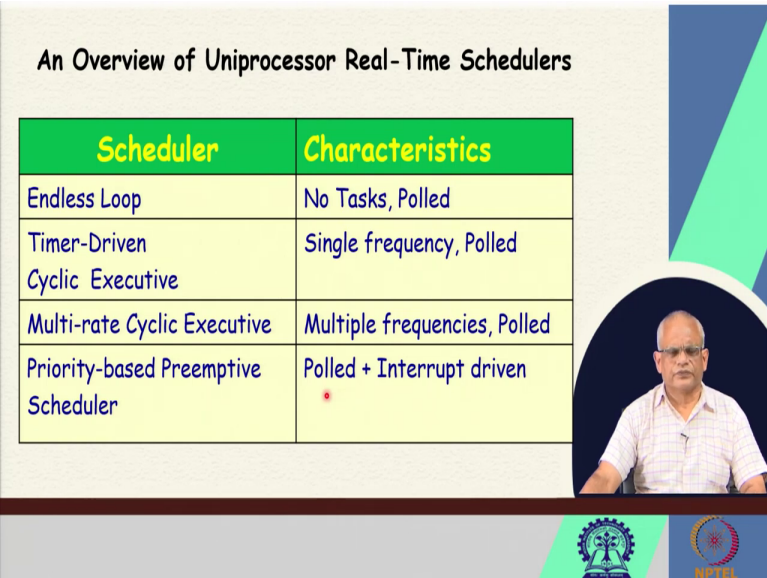
As I was saying that a lot of development occurred in real time task scheduling on uniprocessors in 1970s and 80s. Some are landmark results, we will discuss some of those results, very fundamental results, we will discuss and we will apply that in task scheduling. The real time task schedulers on a uniprocessor can broadly be classified into clock driven and event driven, this classification is done based on how the scheduling points are defined.

Remember, that the scheduling point is a very fundamental thing in real time task scheduling. The scheduling point is a point of time, a point in time at which the scheduler becomes active the scheduler does not run continuously. At certain points in time, the scheduler gets activated, and it

tries to find out which task to run next, and it may take out a task and assign it to the processor and run it.

In a clock driven scheduler, the scheduling points are defined by clock interrupts and in event driven scheduler, the scheduling points are defined by events other than clock events. So, the basic classification of Task Scheduler into clock driven and event driven, it is based on how the scheduling point is defined in a clock driven scheduler, the scheduling point is defined based on clock interrupts. And in event driven scheduler, the scheduling points are defined by other events, we will see what events exactly.

(Refer Slide Time: 5:50)



An Overview of Uniprocessor Real-Time Schedulers

Scheduler	Characteristics
Endless Loop	No Tasks, Polled
Timer-Driven Cyclic Executive	Single frequency, Polled
Multi-rate Cyclic Executive	Multiple frequencies, Polled
Priority-based Preemptive Scheduler	Polled + Interrupt driven

Now, let us look at an overview of the different task scheduler. Possibly the simplest scheduler, which maybe it does not really qualify to be called as a scheduler. But then, in very, very simple primitive system, we just have a endless loop. While true loop, and inside that we pull the different events and process. Very simple system. But then, we in a slightly more sophisticated embedded systems, we have, we call it a cyclic executing.

Let us try to understand what is the difference between a operating system and an executive, an executive is basically the scheduler that is the operating system. The operating system does not offer too many facilities other than task scheduler. So, the scheduler itself is the operating system


and we call it as the executive, the cyclic executive based on a timer, where we have a single frequency of operation, that is the different events occur based on a common frequency, multiple of a common frequency.

So, at every, we set a timer and we just check at which points the different events occur, and then run them that is the simple timer driven cyclic executing. The third one, the third one here is the multi rate cyclic executive. So, here we have multiple frequencies at which tasks occur, a small variation of the timer driven cyclic executive, your tasks can have multiple frequencies. And here again, based on the timer interrupt, the specific event is pulled and actions are taken. We will look at these two.


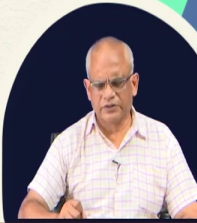
And in a more sophisticated system, we have priority based preemptive scheduling. So, these are the event based, the middle two are clock driven. The fourth one is a event driven scheduler. And the first one of course, even clock is not there just a simple program. So, in a priority based preemptive scheduler, we also poll and also there are specific interrupts, which needs to be serviced. We will look at these in more detail.

(Refer Slide Time: 9:22)

**Clock-Driven Scheduling: Basics**

- Decision regarding which job to run next is made only at clock interrupt instants: 
- Scheduling points determined by a timer
- Timer can be one-shot or periodic
- Which task to be run when and for how long is stored in a table.

Task	Time
1	5
3	7
2	10
7	5
6	2



First, we will look at the clock driven Scheduler. In the clock driven scheduler, there is a clock which give interrupts. So, the clock interrupts are obtained in the rising edge of the clock pulse

and these are the scheduling points. The rising edge of the clock pulses are typically used as the scheduling points. But one thing is that even though we have shown it to be periodic timer, timers can be of two type.

One type of timer is called a one-shot timer. In a one-shot timer we set the timer and it just gives one timeout signal, one timer interrupt, but in a periodic timer, we set the timer and it keeps on giving interrupts at regular intervals. So, let me just repeat that in a one-shot timer if we set the timer for some time, it gives one interrupt when the time elapses, the given time elapses. On the other hand in a periodic timer, we set the timer and once we set it every fixed interval that we have set it keeps on giving interrupts.

And in the clock driven scheduler, we have a table which contains the tasks and when exactly it is to be run that we call as the scheduled table something like this. So, here as you can see that the first column here is the task, task number. And the second column is the time for which it will run. So, for task one, it will run for 5 units. Let us say we have set a timer whose time is 1 unit, so, it will run for 5. And then the next one is 7, 10, 5, 2 etc.

So, fundamental to all clock driven scheduler is a table like this called the scheduled table. And once the scheduler becomes active at the scheduling point, the control is transferred on the interrupt to the scheduler, the scheduler starts running. And it consults the scheduling table and it dispatches the task, the next task on that table we will just see details of this. But that is the main idea here. And the timer interrupt can come at periodic intervals or maybe at different intervals depending on whether we are using a one shot timer or a periodic timer.

(Refer Slide Time: 13:35)

**Assumptions**

- No actual processes exist:
  - Each minor cycle is just a sequence of procedure calls.
- The procedures share a common address space and share data.
  - This data need not be protected (via a semaphore, for example) because "processes" are not preempted.
- All "process" periods are multiples of the cycle time.

The slide features a yellow title box for 'Assumptions', a circular inset photo of a man in a light-colored shirt, and logos for institutions at the bottom. The background is a light beige color with a blue and green geometric design on the right side.

Now, let us look at some assumptions behind the clock driven scheduler. The clock driven scheduler as I was mentioning, they are very simple. Here we do not even have tasks. Its just a program with some functions, and so on procedures. But there are no tasks even though we refer them as tasks, they are not really tasks for our convenience maybe we refer to them as tasks.

And here in each cycle, if it is a, on each interrupt, clock interrupt, we run few sequence or procedures and here the program is simple and all the procedures share common address space and also share data just like a C program with a global data and function calls and here the data is not protected we do not use semaphore etc. because here every task is assumed to run up to its completion every time slot is, every clock interrupt, we run the code that should be run to completion, we do not leave the data structure and interrupt the tasks and so on since interrupting a task is not there here a task runs to completion, task means a set of code.

So, there is no concept of a semaphore here, we do not use semaphore, the code runs, the intended code runs in the assigned time slot and we do not need semaphores and all the processes here that is the code segments or a few procedure calls, they are assigned some multiples of the basic clock interrupts.

(Refer Slide Time: 16:01)

**Assumptions**

- Clock-driven schedulers used in deterministic systems
- Tasks are assumed to be periodic:
  - Also the parameters of all periodic tasks are assumed to be known a priori
- Aperiodic jobs may exist
- There are no sporadic jobs
  - Recall: sporadic jobs have hard deadlines, aperiodic jobs do not

The slide features a yellow header with the word 'Assumptions' in black. The main content is a list of assumptions in blue text. A small circular inset photo of a man in a white shirt is on the right. At the bottom, there are two logos: one of a tree and another of a gear.

And we call this as deterministic because we know the task characteristics, the clock driven schedulers are run in deterministic systems, very simple systems, where we know what are the tasks to be run, what code to be run, and when to run all that we know. So, these are used in deterministic systems, for more sophisticated systems, we need to use event driven scheduler and all the tasks are assumed to be periodic based on the clock interrupt we will service these tasks.

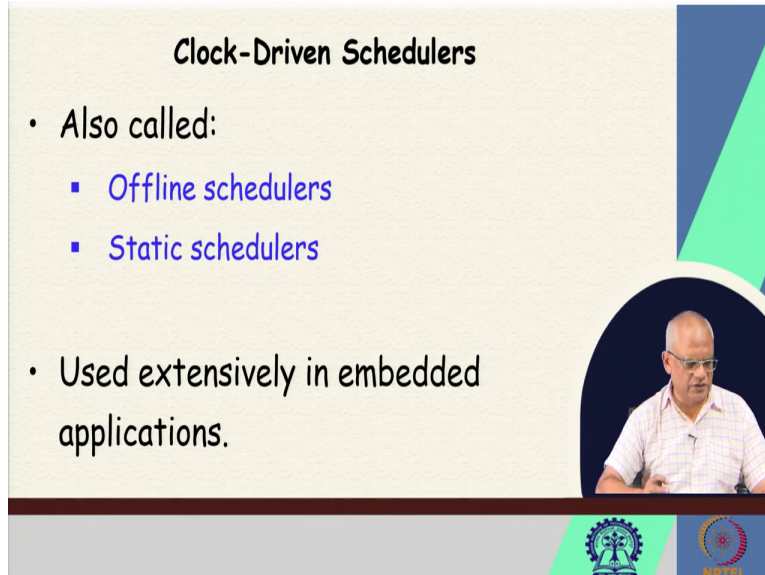
And also, the different parameters of the task are known a priori for example, what will be the execution time what processing exactly needs to be done, which code to be run and so on, for every task, this is known, the maximum execution time which code to run and so on is known. But then, we can accommodate few aperiodic jobs also but, in a clock, driven scheduler, it is very hard to accommodate sporadic jobs.

We had said that sporadic jobs are also they occur at random time instants. But they have hard deadlines. So, in a clock driven scheduler it becomes extremely difficult to accommodate sporadic jobs, we can in a event driven scheduler, we can accommodate sporadic jobs, and we will see how to do that a little later. But in clock driven scheduler, which are used in very simple systems, we have only few deterministic periodic tasks and if at all few aperiodic tasks, it is said that a periodic tasks is like a human query and so on those are examples of periodic jobs.

(Refer Slide Time: 18:17)

### Clock-Driven Schedulers

- Also called:
  - Offline schedulers
  - Static schedulers
- Used extensively in embedded applications.



Now, the clock driven schedulers are also known as offline schedulers or static schedulers. These are called offline because the designer manually prepares the schedule that which tasks to be run at what instant at what clock interrupt, which tasks to run, that all is decided during the design time. And therefore, it is offline scheduling, no scheduling decisions as such need to be taken during runtime accepting consulting the table and trying to run the task that is next.

These are also called static schedulers, because all the tasks are known beforehand that no new tasks can be accommodated or in other words, the system is static that is well known tasks, few of them and run at regular intervals. So, these are static schedulers. But in small embedded applications these are used extensively and remember that small embedded applications are vast in number.



Many of them are hidden. We do not see them. There are many safety critical systems where we have embedded applications running inside we do not see them, they just keep on running and many of them they do use the clock driven schedulers.

(Refer Slide Time: 20:15)



### Pros and Cons of Clock-Driven Schedulers

- **Pro:**
  - **Compact:** Require very little storage space
  - **Efficient:** Incur very little runtime overhead.
  - **Small code:** Can be proven to work correctly
- **Con:**
  - **Inflexible:** Very difficult to accommodate sporadic tasks.
- Used in low cost applications



Now, let us understand what are the advantages and disadvantages of the clock driven schedulers. So, we will be in a better position to know where to use this clock driven schedulers and where not to use the clock driven scheduler. One of the most important advantages of the clock driven scheduler is that these are extremely compact just about 100 line of code or less.

And that is the operating system because this is the executive, there is no other operating system support just 100 or so lines of code and require very little storage space and for very tiny embedded systems, these are ideal very small program needing very little storage and these are efficient. The complexity of the scheduler is very manageable just consulting a table identifying which starts to run and running that, very small time overhead for the cyclic executives and another main advantage is that in safety critical systems we are often called to establish that these are very high reliability.

Now, how do we establish that the operating system is of higher reliability that whatever be the situation the event combinations or extraordinary situations, it should not get into some path in the code where it gets delayed or it gets into an infinite loop or it just keeps on waiting and so on. We need to prove that for safety critical systems and when the code is very small like 100 lines or so, simple code it becomes easy, we can even formally prove that the code will behave well under all situations.

So, getting a reliability certification for clock driven scheduler, the cyclic schedulers is much easier than event driven schedulers and large operating system where the code is so big that there are millions of paths in the code possible and there may be some paths we do not even know and if the code gets into those paths, under some situation never know whether it will hang, it will get into infinite loop and so on.

So, that way the clock driven schedulers are advantageous in use in safety critical applications we can even prove the operating system is well behaved. Now, let us look at the disadvantages of the clock driven Scheduler. One is the inflexibility, we cannot accommodate sporadic tasks here we had already said that that a task occurs at random instant and it has hard deadline becomes difficult to accommodate that in a clock driven scheduler.

And also, the tasks which may arise dynamically it becomes difficult to accommodate here these are static schedulers. But as you are saying that there are hundreds and thousands of low-cost applications where the clock driven schedulers are popular.

(Refer Slide Time: 24:28)

**Popular Clock-Driven Schedulers**

- **Round robin schedulers**
  - Not used in real-time applications
- **Table-driven Schedulers**
- **Cyclic Schedulers**

The slide features a light beige background with a blue and green geometric design on the right side. A circular inset shows a man in a white shirt speaking. At the bottom, there are logos for institutions, including one with a tree and another with a gear.

Now, let us look at the types of clock driven scheduler. The round robin scheduler also qualifies as a clock driven scheduler, because based on clock interrupt, we run the next task in a round robin manner. But then, in real time systems round robin schedulers are not normally used because it becomes difficult to meet the deadline subtasks, it just gives equal time slice to all tasks and a task which is nearing its deadline, it takes no special effort to meet the deadline. So, the round robin schedulers are not really used in real time applications even though it is a clock driven scheduler, and that is why we have included here but will not spend much time on the round robin schedulers.

The table-driven schedulers here, we have a basic table containing when which tasks to be run. And based on that, we run those tasks and that is the table-driven scheduler, extremely simple scheduler. The main thing here is about the table and every scheduling point, it just consults the table and runs the next task. And then the cyclic schedulers. These are slightly more sophisticated than the table-driven schedulers and they optimized several things, several of the shortcomings of the table-driven schedulers are overcome using cyclic schedulers.

The cyclic schedulers are very popular. Out of all the three, the cyclic schedulers are most popular, round robin schedulers almost not used in real time applications, table driven

schedulers, they have lots of difficulties, shortcomings, we will see that. The cyclic scheduler is the one which overcomes that, and we will see that these are very popular.

(Refer Slide Time: 26:51)



**Round Robin Scheduler**

- Periodically releases the CPU from long-running jobs based on timer interrupts:
  - So that short jobs can get a fair share of CPU time
- **Preemptive:** A process is forced to leave its running state and replaced by another running process
- **Time slice:** Interval between timer interrupts



The slide features a video inset of a man in a light-colored shirt speaking. At the bottom, there are logos for IIT Bombay and NITW.

The round robin scheduler, based on the clock interrupts, it just takes up the next task and run. And here, the long running tasks are interrupted. These are preemptive, whereas the other two that we discuss, we will discuss the table driven and the cyclic scheduler are not preemptive. And here we have time slice the interval between timer interrupt and every job is given a time slice.

(Refer Slide Time: 27:44)

## Round Robin Scheduler: Some Thoughts

- Time slice is a critical parameter:
  - If time slice is too long, scheduler degrades to FIFO
  - If time slice is too short, throughput suffers as context switching cost dominates



The time slice is a critical parameter for the round robin scheduler, because if it is too long, then it becomes a first in first out kind of scheduler. And if it is too short then the scheduler becomes very inefficient because the scheduler needs to be invoked again and again, the scheduler takes time and also the context switch time between different jobs that dominates the computation time and becomes extremely inefficient. We are not going to really discuss about round robin scheduler, maybe you would have also read it in your basic operating system course.

But since it is a clock driven scheduler, we just had a minute or so discussion we will not discuss this further. We will look at the table-driven scheduler and the cyclic scheduler. But then we are almost at the end of this lecture, we will stop here and we will continue in the next lecture. Thank you.