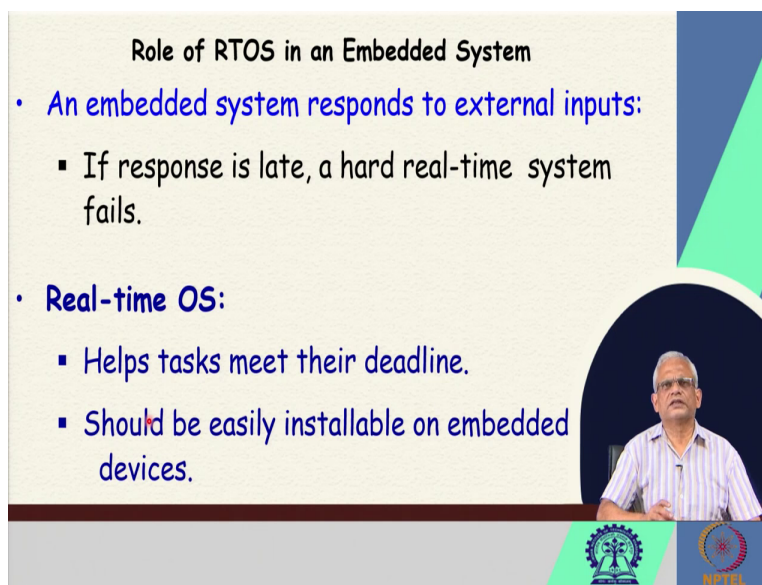


Real Time Systems
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 05
Characteristics of a real-time embedded system

Welcome to this lecture on Real Time Systems. So, far we had looked at few basic topics. Now, let us continue from what we had discussed.

(Refer Slide Time: 00:23)



Role of RTOS in an Embedded System

- **An embedded system responds to external inputs:**
 - If response is late, a hard real-time system fails.
- **Real-time OS:**
 - Helps tasks meet their deadline.
 - Should be easily installable on embedded devices.

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 (IIT) Kharagpur logo on the left and the Real-time Systems logo on the right.

We had remarked that an operating system in a real time system called as the real time operating system, it plays a crucial role in the operation of the system. Every embedded system, non-trivial embedded system would have a real time operating system in it. Now, let us see why do we need an operating system in an embedded system?

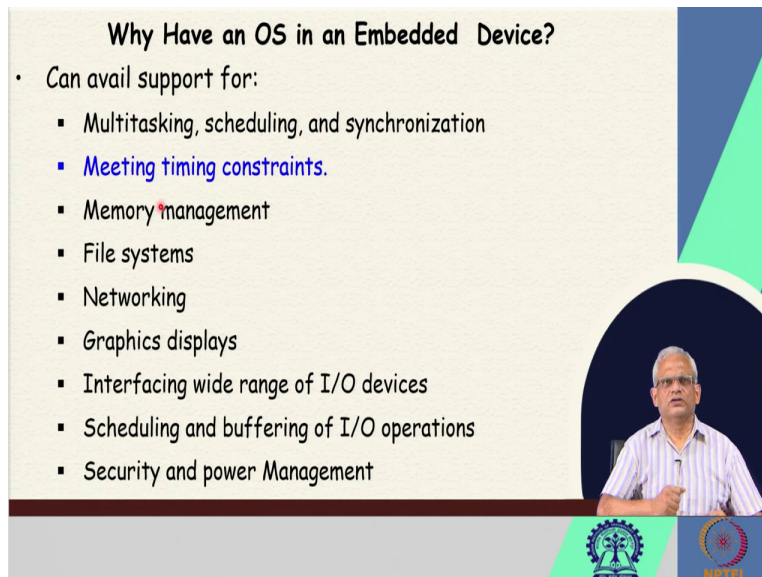
The most important reason for having an operating system in embedded system is that every embedded system responds to various types of inputs that are given by sensors and so on. And these responds must be attended to end results produced in time. And if the response is late, if it is the hard real time system, then we will say that the system has failed.

We will define what is a hard real time system? How is it different from firm and soft real time system? And we will see that in hard real time system any single task failing logically that is producing incorrect result or producing late results in failure of the system. And in this context, the role of the real time operating system is that once the designer specifies time constraints with tasks the real time operating system it helps the task to meet its deadline.

Another important concern for the real time operating systems is that as far as embedded devices are concerned, the hardware is specific to the device. Do not use a general purpose of the self device like a desktop or a laptop, special hardware is manufactured, and therefore, the real time operating system should be easily implemented in various types of hardware.

We will see how is that done? How can operating system be easily installable across various types of hardware? And that is one of the important requirements for the real time operating system other than helping the tasks to meet the deadline. There are various other roles or purposes of the real time operating system. We will look at them in this course.

(Refer Slide Time: 03:28)



Why Have an OS in an Embedded Device?

- Can avail support for:
 - Multitasking, scheduling, and synchronization
 - Meeting timing constraints.
 - Memory management
 - File systems
 - Networking
 - Graphics displays
 - Interfacing wide range of I/O devices
 - Scheduling and buffering of I/O operations
 - Security and power Management

The slide features a list of OS services. A small inset video shows a man in a striped shirt speaking. At the bottom, there are two logos: one for a university and another for WPIU.

Now, let us identify some of the important reasons why we need a real time operating system in a embedded device. The first thing is that the system must respond to various types of events which are produced by the environment of the system. Maybe there are various types of sensors,

maybe a dozen sensors or maybe two dozen sub-sensors and each sensor produces some signals and these are input to the system.

And for each of these events the system typically creates one task to handle that event. And naturally any non-trivial embedded system is multitasking, should be able to have multiple tasks, should be able to schedule among these tasks and the tasks may need to communicate and therefore synchronization among the tasks.

As we had already identified, helping the tasks meet their time constraints. That is one of the very important roles of operating system in embedded device. Memory management, different tasks may have different data that they deal with some are private, some are shared, the program code itself and there may be multiple tasks and the code for these tasks. So, all these are stored in memory. Memory management is important issue for these operating systems.

The file systems there may be permanent data stored. And these are stored in files or databases. And that is also to be supported by the operating system. Networking, nowadays most of the embedded devices are Internet enabled. And therefore, supporting networking and providing basic network facilities to the real time tasks is a important role of the operating system.

Displays, the users or the operators of these systems they would need graphic display, various parameters and interaction with the system. Various types of I/O devices are typically attached to an embedded device, for example, a camera, maybe a microphone, maybe a heater, various types of actuators, sensors and so on. And the operating system should support interfacing various types of I/O devices.

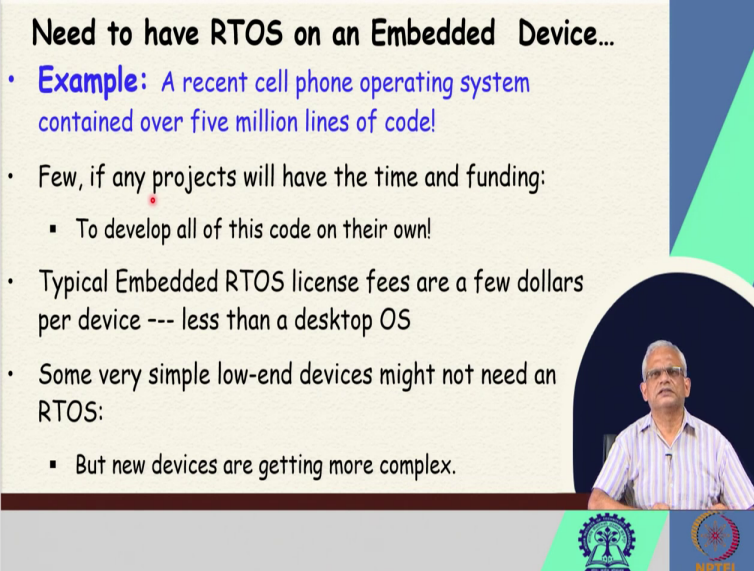
And then scheduling and buffering the I/O operations. Security, and nowadays because these are Internet-enabled, security, providing security to the system so that malicious users should not be able to hack the system and power management. These are some of the important features of a real time operating system. As part of this course, we will look at the basic features of the real

time operating system. And we will see all these issues. But let me just mention that possibly the most crucial and most investigated issue is the real time task scheduling.

We will build up with the very basic notions of real time task scheduling. We will call them as clock driven schedulers. Those are the real time task schedulers used in very simple systems. And one thing is that the simple systems are very large in number. Many places we find these simple systems which are using clock driven schedulers.

And of course, the event-driven schedulers that is a very large topic we will discuss, we will take several classes to discuss the nitty-gritty, schedulability, various modifications, improvisations, and so on to the schedulers and task synchronization for the real time tasks. So, that is one of the largest topic in this course, that is real time task scheduling, resource sharing, synchronization. And we will look at real time databases, real time communications and so on.

(Refer Slide Time: 08:46)



Need to have RTOS on an Embedded Device...

- **Example:** A recent cell phone operating system contained over five million lines of code!
- Few, if any projects will have the time and funding:
 - To develop all of this code on their own!
- Typical Embedded RTOS license fees are a few dollars per device --- less than a desktop OS
- Some very simple low-end devices might not need an RTOS:
 - But new devices are getting more complex.

The slide features a video inset of a man in a striped shirt speaking. At the bottom, there are logos for a university and NPTEL.

But a question naturally arises is that is it necessary to have a real time operating system in embedded device? The embedded devices are large in number and very inexpensive. Can we afford to have a real time operating system? Do we need all the features there? Cannot we just write a small workable system during the program development? And have those features developed along with the application? The specific features, for example, networking or

whatever is needed. Cannot we just write that as part of the program development? Why have full operating systems? Would not make the system expensive?

Let us just address that question. Let us just examine a cell phone. If we, it has, it comes with operating system. Maybe it is Android or something. But if we look upon examine that operating system, we will see that the size of the operating system is huge, 5 million lines of code. And if somebody argues that why not develop some of those basic features along with the application program that we are developing so that we save costs on the operating system.

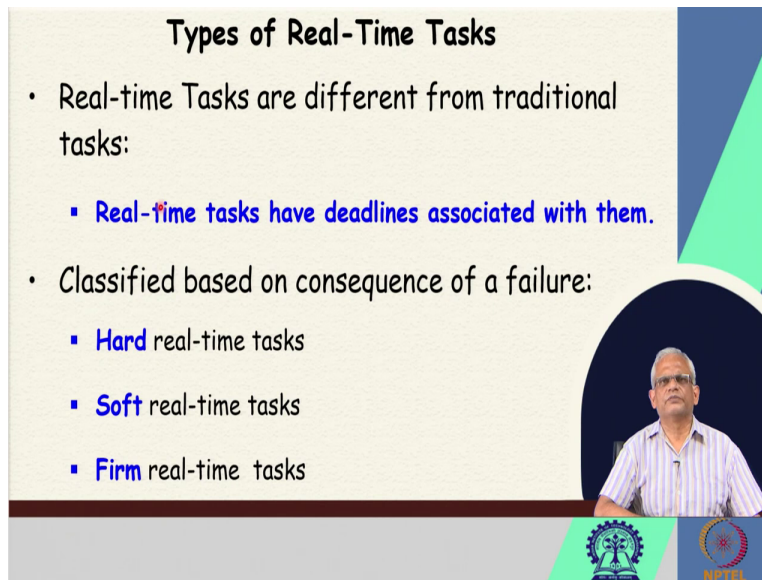
Then just look at this, that to develop all this code on their own, you will be over budget over time. Just look at the size of the code. The application program maybe several thousands of lines of code, maybe 10,000 or 20,000 lines of code. And here we have 5 million lines of code, much more than the application program. And whatever the application, whatever the license fee for this operating system.

Per device the license is a few dollars and in rupees terms, it may be few hundreds of rupees. And if we do not have that in embedded device, the system development may never complete, it will be full of bugs, because you are developing such a large system in a limited time and these operating systems are well tested across thousands of applications. But again, there are some very low-end devices, maybe using a 8 bit processor and doing very limited tasks like sampling some temperature, and based on the temperature reading, switching something on or off. Maybe the program code itself is a few hundred lines.

We do not need a 5-million-line operating system for that, because the embedded device might have a very small memory, a very simple processor. And running this large operating system will be counterproductive. And there we might just have a simple operating system. We call this a monitor developed as part of the application program. During program development, we will discuss about the monitor approach, where we develop a simple scheduler, clock driven scheduler. We just sample some data from sensors, run the code as required and in time and the operating system code maybe few hundreds of lines.

So, very simple low-end devices may not need a full-fledged operating system. But a vast majority of the embedded devices do need a licensed or open-source operating system to be installed. And slowly the number of embedded devices with complex functionalities is increasing very rapidly.

(Refer Slide Time: 13:31)



Types of Real-Time Tasks

- Real-time Tasks are different from traditional tasks:
 - Real-time tasks have deadlines associated with them.
- Classified based on consequence of a failure:
 - Hard real-time tasks
 - Soft real-time tasks
 - Firm real-time tasks

The slide features a video inset of a man in a striped shirt speaking. At the bottom, there are logos for a university and a research center.

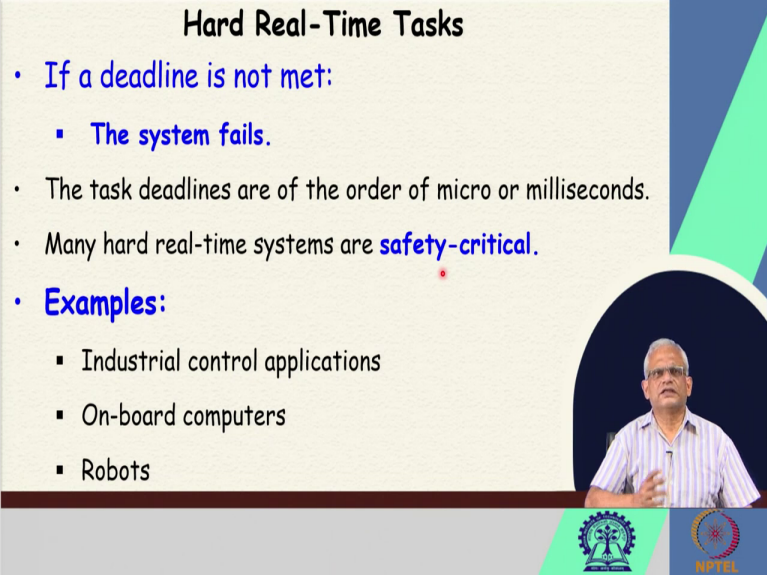
Now, let us examine a very fundamental issue that is the types of real time tasks. We said that in response to a system input, the system input is typically either a sensor input or a user input a task is created. In response to a system input, sorry, sensor input or a user input a task is created and these tasks are of various types.

Now, let us understand what are the different types of tasks that may be there in a real time system? One thing is that, if we say that it is a real time task, what we mean is that, they have some time constraints associated with them. Typically, it is a deadline, deadline constraint that the task must complete before some time.

And these tasks, one way to classify these tasks is based on the consequence of a task failing that is producing a incorrect result or exceeding the deadline. There are two ways in which a task can fail; one is produces a wrong result; the second is that, it produces the result later or after the deadline has passed. Now, depending on these types of failure we have three types of task, hard

real time tasks, soft real time tasks and firm real time tasks. Now, let us look at these three categories of tasks.

(Refer Slide Time: 15:37)



Hard Real-Time Tasks

- If a deadline is not met:
 - The system fails.
- The task deadlines are of the order of micro or milliseconds.
- Many hard real-time systems are **safety-critical**.
- **Examples:**
 - Industrial control applications
 - On-board computers
 - Robots

The slide features a video inset of a man in a striped shirt speaking. At the bottom, there are logos for a university and a center named 'SFTL'.

Now, first, let us look at the hard real time tasks. We say a task to be hard real time if a deadline is not met, then the system fails. The task is not meeting its deadline or producing an incorrect result. And in that case, the system itself fails. Remember that the system may have many tasks, but just one hard real time task failed because it missed its deadline or maybe there was an incorrect result produced and the system fails. So, this is the characteristic of a hard real time task. And here, the task deadlines are of the order of few micro or milliseconds.

Rarely, the deadline is a second, mostly talking of the task completion times of the order of a few micro or milliseconds. And many of these hard real time tasks are associated with systems which are safety critical. We will define this term as we proceed today in this lecture. What is a safety critical system? In very layman term, a safety critical system is one which causes severe damage, injury or loss of life, if there is any failure. So, let me just repeat that. A safety critical system is one where any failure of the system causes severe damage, injury or even loss of life.

Let us look at some examples of hard real time tasks in systems. Industrial control applications, they have hard real time tasks, onboard computers, and fly-by-wear aircraft, we are just

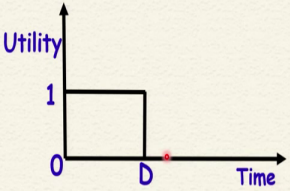
discussing about a drone which carries out some operations, safety critical operations, maybe surveillance of enemy areas or maybe a fly-by-wear aircraft or an automated, an autopilot system. And there we have onboard computers and many of the tasks in these systems are hard real time tasks.

A robot is another example where we have had real time task. One is, for example, the task handling the movement of the robot. So, if the movement is not done in time, the robot may collide, for example. So, these are some of the systems where we have hard real time tasks.

(Refer Slide Time: 19:04)

Firm Real-Time Tasks

- If a deadline is missed occasionally, the system does not fail:
 - The results produced by a firm real-time task after its deadline are rejected.



The slide includes logos for IIT Bombay and NPTEL at the bottom.

Now, let us look at the firm real time tasks. Here a firm real time task is one where if there is any failure of the task, that the task does not meet its deadline or the logical incorrectness, then even if the task fails, the system does not fail. It is only the tasks that fail. And the results that are produced by the firm real time tasks which are produced late or incorrect, these are just rejected but the system does not fail.

We can represent the task utility in this graph. If the result is produced before a time D , D is the deadline, and then the utility of the result is 100 percent. But if the result is produced after the deadline, the utility of the task is 0. That means we just discard the result. It is not useful. So, that is the characteristic of a firm real time task.

(Refer Slide Time: 20:18)

Firm Real-Time System: Examples

- A video conferencing application
- A telemetry application
- Satellite-based surveillance applications



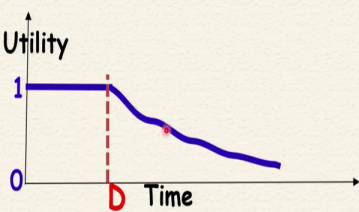
Some examples of firm real time tasks is a video conferencing applications. Here the frames are received. And if there is a delay in receiving frame or a delay in processing the frame, then these are simply discarded. If we still show a delayed frame, then it will appear as a glitch in the video. The users can notice that there is something wrong. But since there are many frames per minute are being played, if few of the frames or missed, the user may not notice them. So, few frames which are processed late can be simply dropped.

A telemetric application, data is received maybe every few milliseconds or a second. And if there is a delay in some data, because these data are getting processed in real time, there is a delay in processing or receiving up some data these may be simply dropped without really hampering the functioning of the system. A satellite-based surveillance application. So, here, signals are received by the satellite sent and these are processed for surveillance. And a few of the signals are delayed or missed. These can be dropped without really hampering the surveillance application as long as these are not too many, just few of them can be simply ignored.

(Refer Slide Time: 22:08)

Soft Real-Time Tasks

- If a deadline is missed, the system does not fail:
 - Only the performance of the system is said to have degraded.
 - The utility of results decrease rapidly with time after the deadline.



The graph illustrates the utility of results over time for soft real-time tasks. The vertical axis is labeled 'Utility' and has a scale from 0 to 1. The horizontal axis is labeled 'Time'. A blue line represents the utility curve, which remains constant at a value of 1 until a deadline 'D' is reached, indicated by a vertical red dashed line. After the deadline, the utility drops sharply and then more gradually, approaching 0. A small inset video of a man is visible on the right side of the slide.


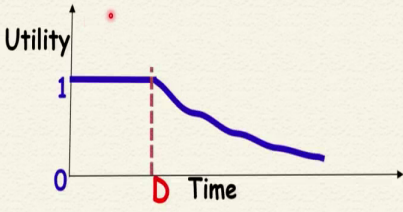
Now, let us look at the third category of the task. We had looked at the hard real time, firm real time; now let us look at the soft real time tasks. In the soft real time task, if a deadline is missed, then the system does not fail. And also, unlike the firm real time system, the late results are not discarded. These are used, but the utility of the result decreases with time.

And if the result is produced late, we say that the system is having a degraded performance. If we plot this in the form of a graph, we will see that the result of the task has a utility value of 100 percent until a deadline. And after the deadline it does not become 0 like a firm real time system. But then it is decreasing with time.

(Refer Slide Time: 23:25)

Soft Real-Time Tasks

- **Another definition:**
 - Probabilistic requirements on deadline.
 - For example, 99% of deadlines should be met.

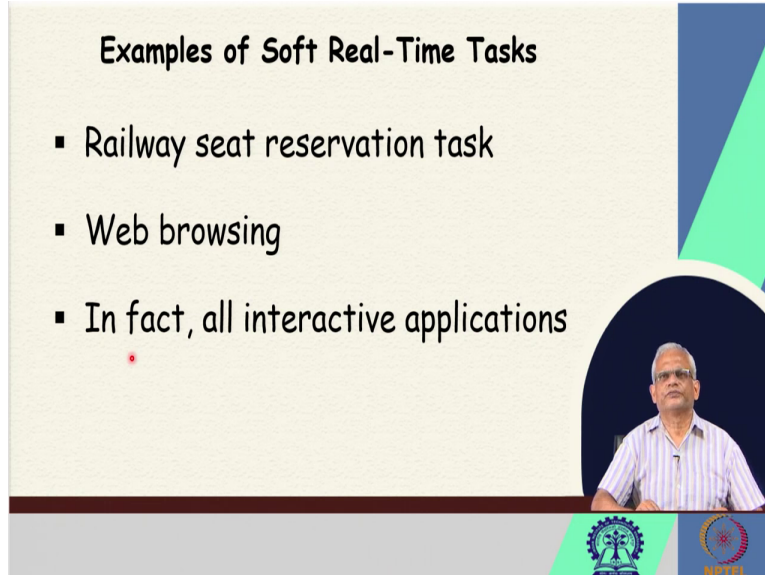


Just to take an example. Ok, before example, let us look at another definition of a soft real time task. Here we assume that there may be several misses of the deadline, but we give a requirement and the system in terms of probabilistic meeting of the deadline by the tasks. We may say that 90 percent of the time the result will be produced in time or before deadline or maybe 60 percent of the time the results will be produced within the stipulated time, or we may say that even in overloaded situations 60 percent of the results will be produced within time. Or another example is 99 percent of the deadline will be met.

(Refer Slide Time: 24:30)

Examples of Soft Real-Time Tasks

- Railway seat reservation task
- Web browsing
- In fact, all interactive applications



Now, let us look at some example. Let us look at a railway seat reservation task. So, here, you are trying to make a reservation on a railway, a seat reservation. Now, if you made a booking and you got the booking and printed the ticket, you said that the system performed well. Now, let us say you are just waiting. You just give the booking request and it is just taking time. 30 seconds passed. You waited for a minute.

And then you said you are thinking that is the system not responding, is there something wrong with the system. And if it produces the result after 2 minutes, you say that the system has a degraded performance. And maybe after 5 minutes, you might just close the application and say that no, it is not working now.

The typical human response time is of the order of few seconds, like 10 seconds or 20 seconds. So, if the result is produced in 10 or 20 seconds, you say that the system is working fine. But as it becomes late than 20 second, maybe 30 second, a minute and so on, you said that the system is having a degraded performance.

Similarly, web browser, once you clicked on a link and the page displayed within 10 seconds, you had do not, you say that it has appeared instantly, because human response time is of the order of 10 to 20 seconds. You do not notice any delay if it displays within 10 seconds or so, say

that the system is having good performance. If the page takes a minute to display, you say that it is having a degraded performance.

And for that matter, all interactive applications are actually soft real time tasks where a human being gives input, waits for a response from the system, those are all interactive tasks, and all interactive tasks can be classified as soft real time tasks. So, we just looked at some very basic definitions about the tasks and the types of tasks. We are at the end of this lecture. We will stop here and continue from here in the next class. Thank you.