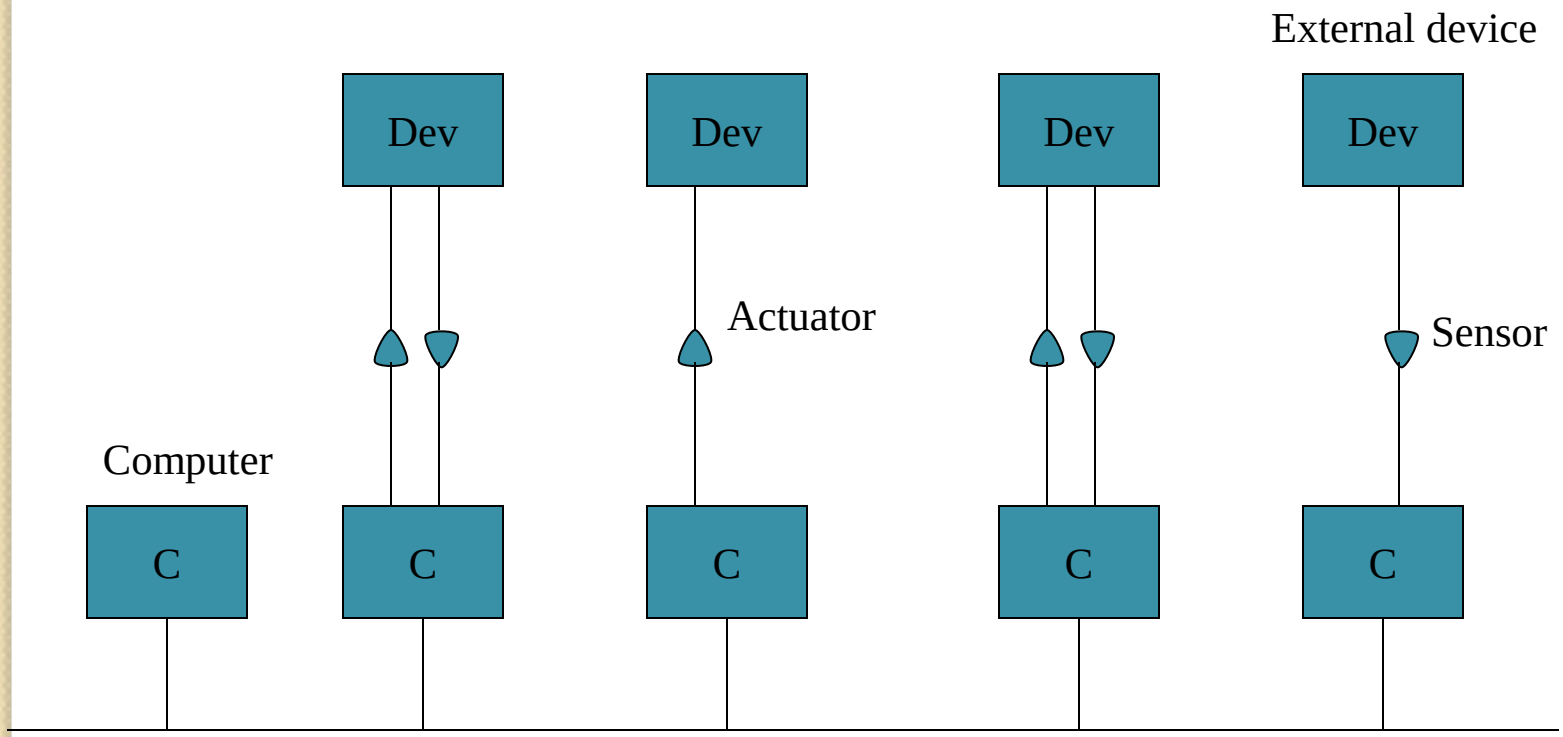
LECTURE 20
REAL TIME DISTRIBUTED SYSTEM

REAL-TIME DISTRIBUTED SYSTEMS

- **What is a real-time system?**
- **Real-time programs** interact with the external world in a way that involves time. When a stimulus appears, the system must respond to it in a certain way and before a certain deadline. E.g. automated factories, telephone switches, robots, automatic stock trading system.

Distributed real-time systems structure



Stimulus

- An external device generates a stimulus for the computer, which must perform certain actions before a deadline.

1. Periodic: a stimulus occurring regularly every T seconds, such as a computer in a TV set or VCR getting a new frame every 1/60 of a second.
2. Aperiodic: stimulus that are recurrent, but not regular, as in the arrival of an aircraft in an air traffic controller's air space.
3. Sporadic: stimulus that are unexpected, such as a device overheating.

Two types of RTS

- Soft real-time systems: missing an occasional deadline is all right.
- Hard real-time systems: even a single missed deadline in a hard real-time system is unacceptable, as this might lead to loss of life or an environmental catastrophe.

Design issues

- **Clock Synchronization** - Keep the clocks in synchrony is a key issue.
- **Event-Triggered versus Time-Triggered Systems**
- **Predictability**
- **Fault Tolerance**
- **Language Support**

Event-triggered real-time system

- when a significant event in the outside world happens, it is detected by some sensor, which then causes the attached CPU to get an interrupt. Event-triggered systems are thus interrupt driven. Most real-time systems work this way.
- Disadvantage: they can fail under conditions of heavy load, that is, when many events are happening at once. This **event shower** may overwhelm the computing system and bring it down, potentially causing problems seriously.

Time-triggered real-time system

- in this kind of system, a clock interrupt occurs every T milliseconds. At each clock tick sensors are sampled and actuators are driven. No interrupts occur other than clock ticks.
- T must be chosen carefully. If it too small, too many clock interrupts. If it is too large, serious events may not be noticed until it is too late.

An example to show the difference between the two

- Consider an elevator controller in a 100-story building. Suppose that the elevator is sitting on the 60th floor. If someone pushes the call button on the first floor, and then someone else pushes the call button on the 100th floor. In an event-triggered system, the elevator will go down to first floor and then to 100th floor. But in a time-triggered system, if both calls fall within one sampling period, the controller will have to make a decision whether to go up or go down, for example, using the nearest-customer-first rule.

Cont..

- In summary, event-triggered designs give faster response at low load but more overhead and chance of failure at high load. Time-trigger designs have the opposite properties and are furthermore only suitable in a relatively static environment in which a great deal is known about system behavior in advance.

Predictability

- One of the most important properties of any real-time system is that its behavior be predictable. Ideally, it should be clear at design time that the system can meet all of its deadlines, even at peak load. It is known when event E is detected, the order of processes running and the worst-case behavior of these processes.

Fault Tolerance

- Many real-time systems control safety-critical devices in vehicles, hospitals, and power plants, so fault tolerance is frequently an issue.
- Primary-backup schemes are less popular because deadlines may be missed during cutover after the primary fails.
- In a safety-critical system, it is especially important that the system be able to handle the worst-case scenario. It is not enough to say that the probability of three components failing at once is so low that it can be ignored. Fault-tolerant real-time systems must be able to cope with the maximum number of faults and the maximum load at the same time.

Language Support

- In such a language, it should be easy to express the work as a collection of short tasks that can be scheduled independently.
- The language should be designed so that the maximum execution time of every task can be computed at compile time. This requirement means that the language cannot support general **while** loops and recursions.
- The language needs a way to deal with time itself.
- The language should have a way to express minimum and maximum delays.
- There should be a way to express what to do if an expected event does not occur within a certain interval.
- Because periodic events play an important role, it would be useful to have a statement of the form: every (25 msec){…} that causes the statements within the curly brackets to be executed every 25 msec.

Real-Time Communication

- Cannot use Ethernet because it is not predictable.
- Token ring LAN is predictable. Bounded by kn byte times. K is the machine number. N is a n-byte message .
- An alternative to a token ring is the TDMA (Time Division Multiple Access) protocol. Here traffic is organized in fixed-size frames, each of which contains n slots. Each slot is assigned to one processor, which may use it to transmit a packet when its time comes. In this way collisions are avoided, the delay is bounded, and each processor gets a guaranteed fraction of the bandwidth.

Real-Time Scheduling

- Hard real time versus soft real time
- Preemptive versus nonpreemptive scheduling
- Dynamic versus static
- Centralized versus decentralized

Dynamic Scheduling

- **1. Rate monotonic algorithm:**
- It works like this: in advance, each task is assigned a priority equal to its execution frequency. For example, a task runs every 20 msec is assigned priority 50 and a task run every 100 msec is assigned priority 10. At run time, the scheduler always selects the highest priority task to run, preempting the current task if need be.

Cont..

- **2.Earliest deadline first algorithm:**
- Whenever an event is detected, the scheduler adds it to the list of waiting tasks. This list is always keep sorted by deadline, closest deadline first.

Cont..

- **3.Least laxity algorithm:**
- this algorithm first computes for each task the amount of time it has to spare, called the laxity. For a task that must finish in 200 msec but has another 150 msec to run, the laxity is 50 msec. This algorithm chooses the task with the least laxity, that is, the one with the least breathing room.

Static Scheduling

- The goal is to find an assignment of tasks to processors and for each processor, a static schedule giving the order in which the tasks are to be run.

A comparison of Dynamic versus Static Scheduling

- Static is good for time-triggered design.
- 1. It must be carefully planned in advance, with considerable effort going into choosing the various parameters.
- 2. In a hard real-time system, wasting resources is often the price that must be paid to guarantee that all deadlines will be met.
- 3. An optimal or nearly optimal schedule can be derived in advance.

Cont..

- Dynamic is good for event-triggered design.
- 1. It does not require as much advance work, since scheduling decisions are made on-the-fly, during execution.
- 2. It can make better use of resources than static scheduling.
- 3. No time to find the best schedule.

ASSIGNMENT

- Q: What are the advantages of Real time distributed system.