REAL TIME OPERATING SYSTEMS

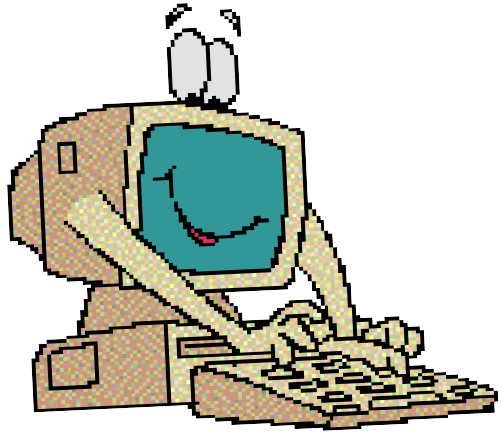
Ege Üniversitesi
Müh. Fakültesi Bilgisayar Mühendisliği Bölümü
Gerçek Zamanlı İşletim Sistemleri
2014
Yar. Doç. Dr. Mustafa Engin

What is an RTOS?

- An RTOS is a class of operating systems that are intended for real time-applications.
- What is a real time application?
- A real time application is an application that guarantees both correctness of result and the added constraint of meeting a deadline.

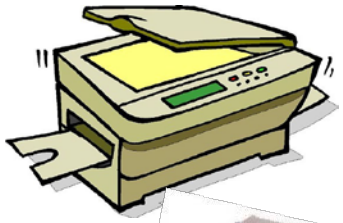
What is an Operating System?

- **The most important program that runs on your computer. It manages all other programs on the machine.**
- **Every PC has to have one to run other applications or programs. It's the first thing "loaded".**



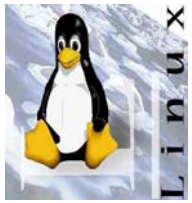
Operating System

- **It performs basic tasks, such as:**
  - **Recognizing input from the keyboard or mouse,**
  - **Sending output to the monitor,**
  - **Keeping track of files and directories on the disk, and**
  - **Controlling peripheral devices such as disk drives and printers.**



Types of Operating Systems

- **Generally, there are four types, based on the type of computer they control and the sort of applications they support.**
- **Single-user, single task**
  - **This type manages the computer so that one user can effectively do one thing at a time.**
- **Multi-user, multi-task**
  - **Allows two or more users to run programs at the same time. Some operating systems permit hundreds or even thousands of concurrent users.**

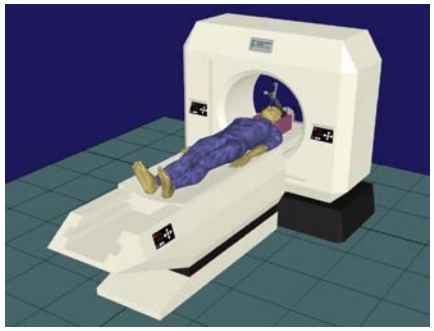


UNIX

Mainframe

Types of Operating Systems

- Single-user, Multi-tasking
  - *This is the type of operating system* ***most desktops and laptops use today****.*
  - *Microsoft's Windows* and *Apple's MacOS* are both examples of operating systems that will let a single user have several programs in operation at the same time.
- Real Time Operating Systems
  - **RTOS are used to control machinery, scientific instruments, and industrial systems.**
  - **There is typically very little user- interface capability.**
  - **Resources are managed so that a *particular operation executes precisely the same every time*.**

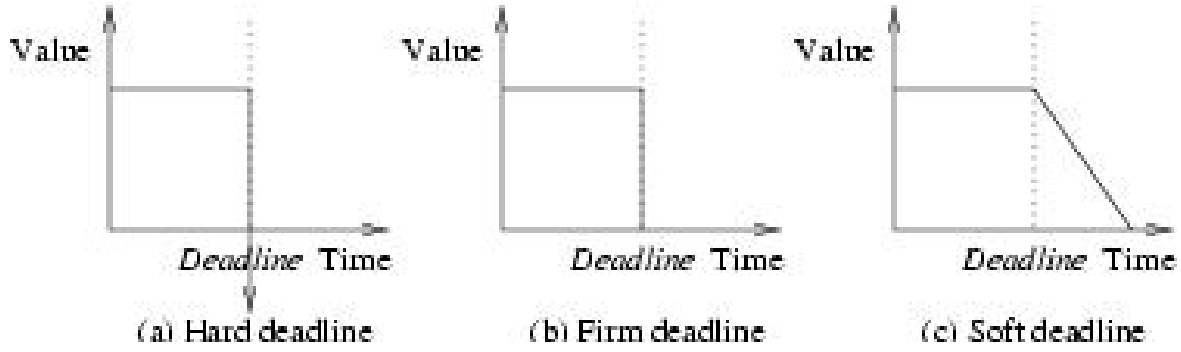


Real Time Systems

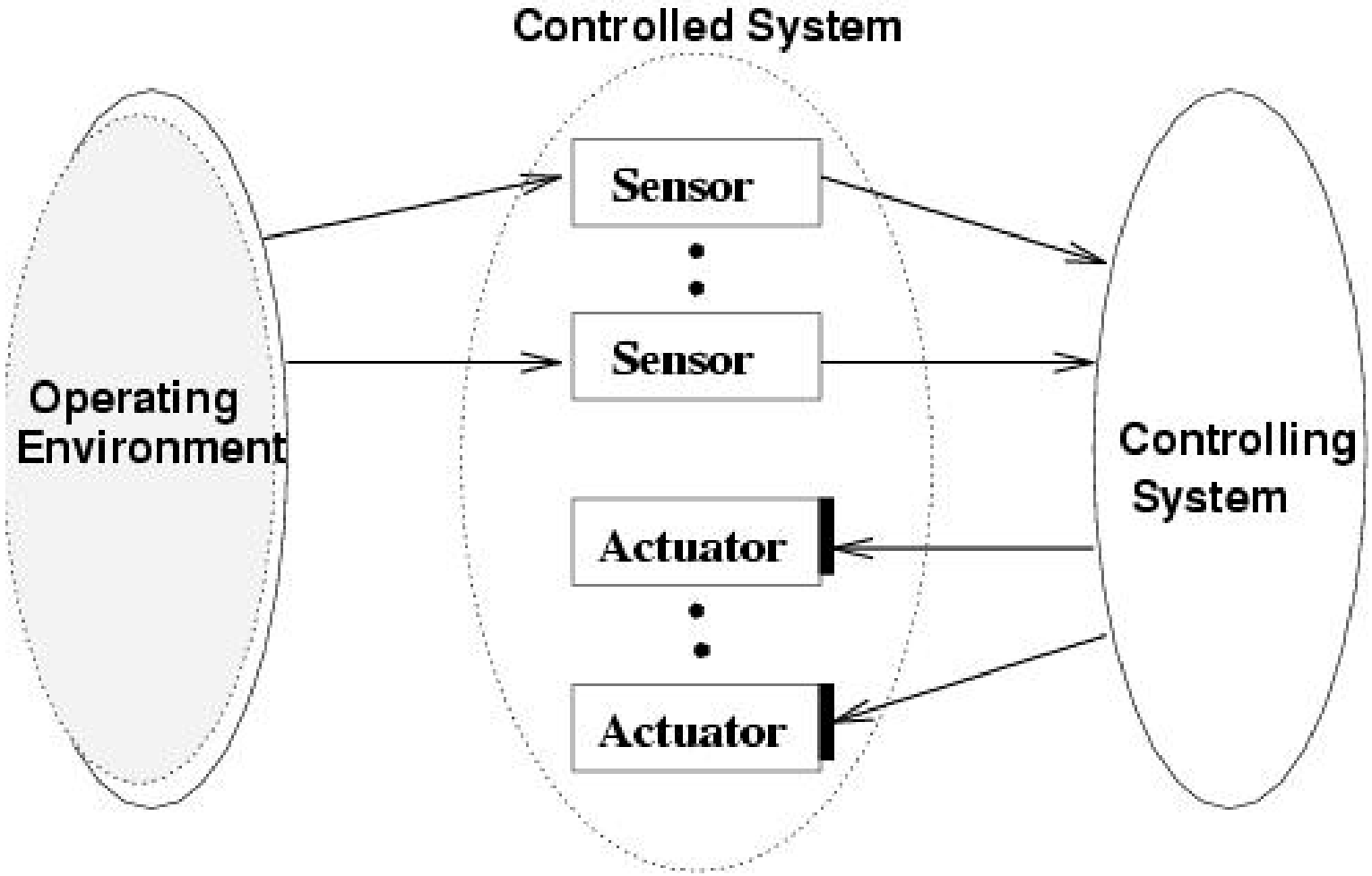
- Real-time systems are defined as those systems in which the correctness of the system depends not only on the **logical result** of computation, but also on the **time** at which the results are produced.
- Hard real-time systems (e.g., Avionic control, robot control, Biomedical instruments).
- Firm real-time systems (e.g., Banking, POS Machine).
- Soft real-time systems (e.g., Video on demand).

Real Time Systems

- Hard deadline: penalty due to missing deadline is a higher order of magnitude than the reward in meeting the deadline
- Firm deadline: penalty and reward are in the same order of magnitude
- Soft deadline: penalty often lesser magnitude than reward



A typical real-time system



Example: Car driver

- ***Mission:*** Reaching the destination safely.
- **Controlled System:** Car.
- **Operating environment:** Road conditions.
- **Controlling System**
  - *Human driver:* Sensors - Eyes and Ears of the driver.
  - *Computer:* Sensors - Cameras, Infrared receiver, and Laser telemeter.
- **Controls:** Accelerator, Steering wheel, Break-pedal.
- **Actuators:** Wheels, Engines, and Brakes.

Example – Car driver (contd)

- **Critical tasks:** Steering and breaking.
- **Non-critical tasks:** Turning on radio.
- **Performance** is not an absolute one. It measures the goodness of the outcome relative to the best outcome possible under a given circumstance.
- **Cost** of fulfilling the mission → Efficient solution.
- **Reliability** of the driver → Fault-tolerance is a must.

Real Time Tasks

- **Periodic tasks**

  - Time-driven. Characteristics are known *a priori*

  - Task T_i is characterized by (p_i, c_i)

  E.g.: Task monitoring temperature of a patient in an ICU (Intensive Care Unit).

- **Aperiodic tasks**

  - Event-driven. Characteristics are **not** known *a priori*

  - Task T_i is characterized by (a_i, r_i, c_i, d_i)

  E.g.: Task activated upon detecting change in patient's condition.

- **Sporadic Tasks**

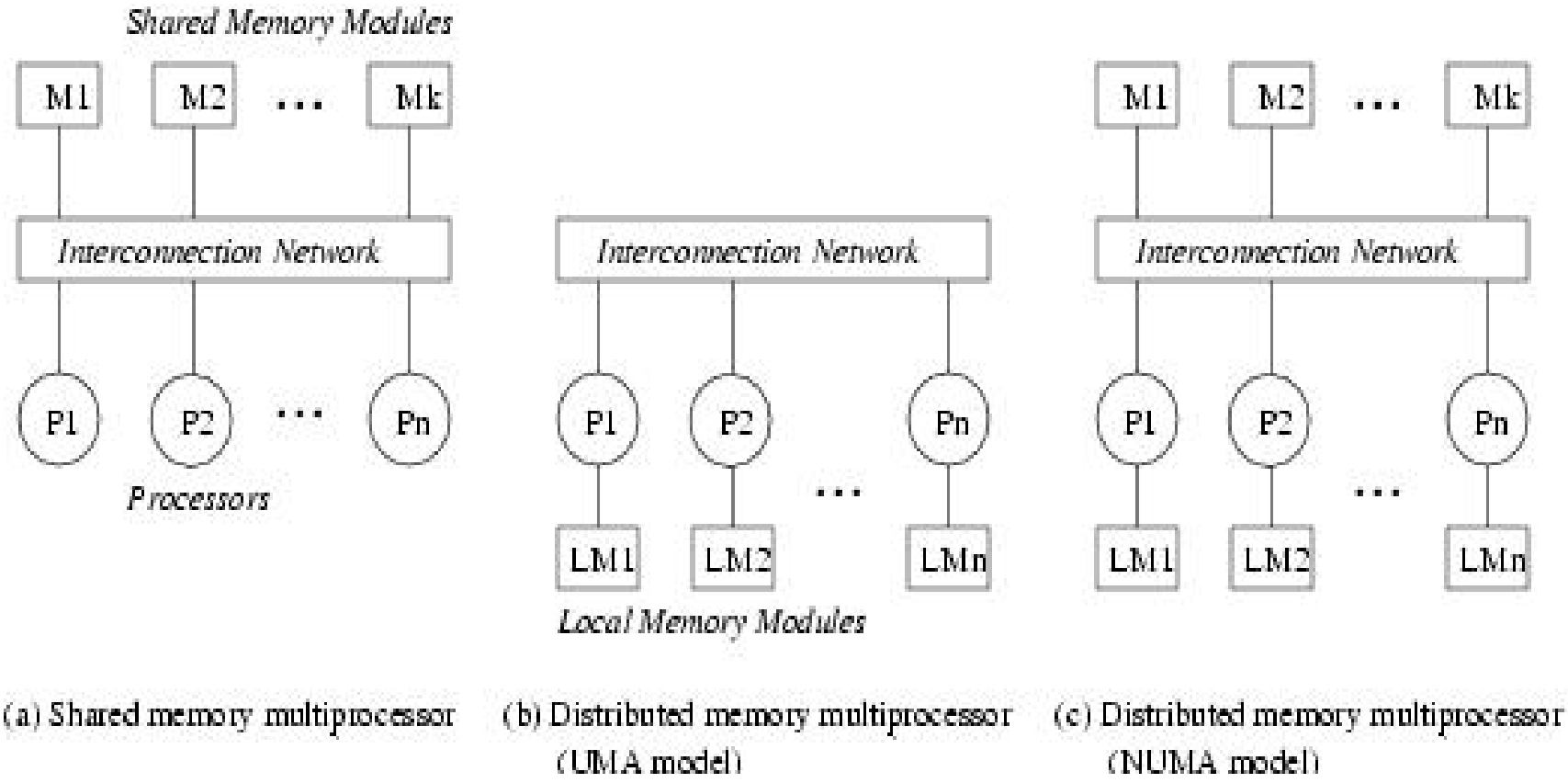
  - Aperiodic tasks with known minimum inter-arrival time.

p_i : task period a_i : arrival time r_i : ready time
d_i : deadline c_i : worst case execution time.

Task Constraints

- Deadline constraint
- Resource constraints
  - Shared access (read-read)
  - Exclusive access (write-x)
- Precedence constraints
  - T1 → T2: Task T2 can start executing only after T1 finishes its execution
- Fault-tolerant Requirements
  - To achieve higher reliability for task execution
  - Redundancy in execution

Computing systems



(a) Shared memory multiprocessor (b) Distributed memory multiprocessor (UMA model) (c) Distributed memory multiprocessor (NUMA model)

Notion of Predictability

- The most common denominator that is expected from a real-time system is *predictability*.
  - **The behavior of the real-time system must be predictable which means that with certain assumptions about workload and failures, it should be possible to show at "design time" that all the timing constraints of the application will be met.**
- For static systems, 100% guarantees can be given at design time.
- For dynamic systems, 100% guarantee cannot be given since the characteristics of tasks are not known a priori.
- In dynamic systems, predictability means that once a task is admitted into the system, its guarantee should never be violated as long as the assumptions under which the task was admitted hold.

Common Misconceptions

- Real-time computing is equivalent to fast computing.
- Real-time programming is assembly coding, priority interrupt programming, and writing device drivers.
- Real-time systems operate in a static environment.
- The problems in real-time system design have all been solved in other areas of computer science.