Real Time System

KAPITA SELEKTA TEKNIK KOMPUTER

AGUS MULYANA

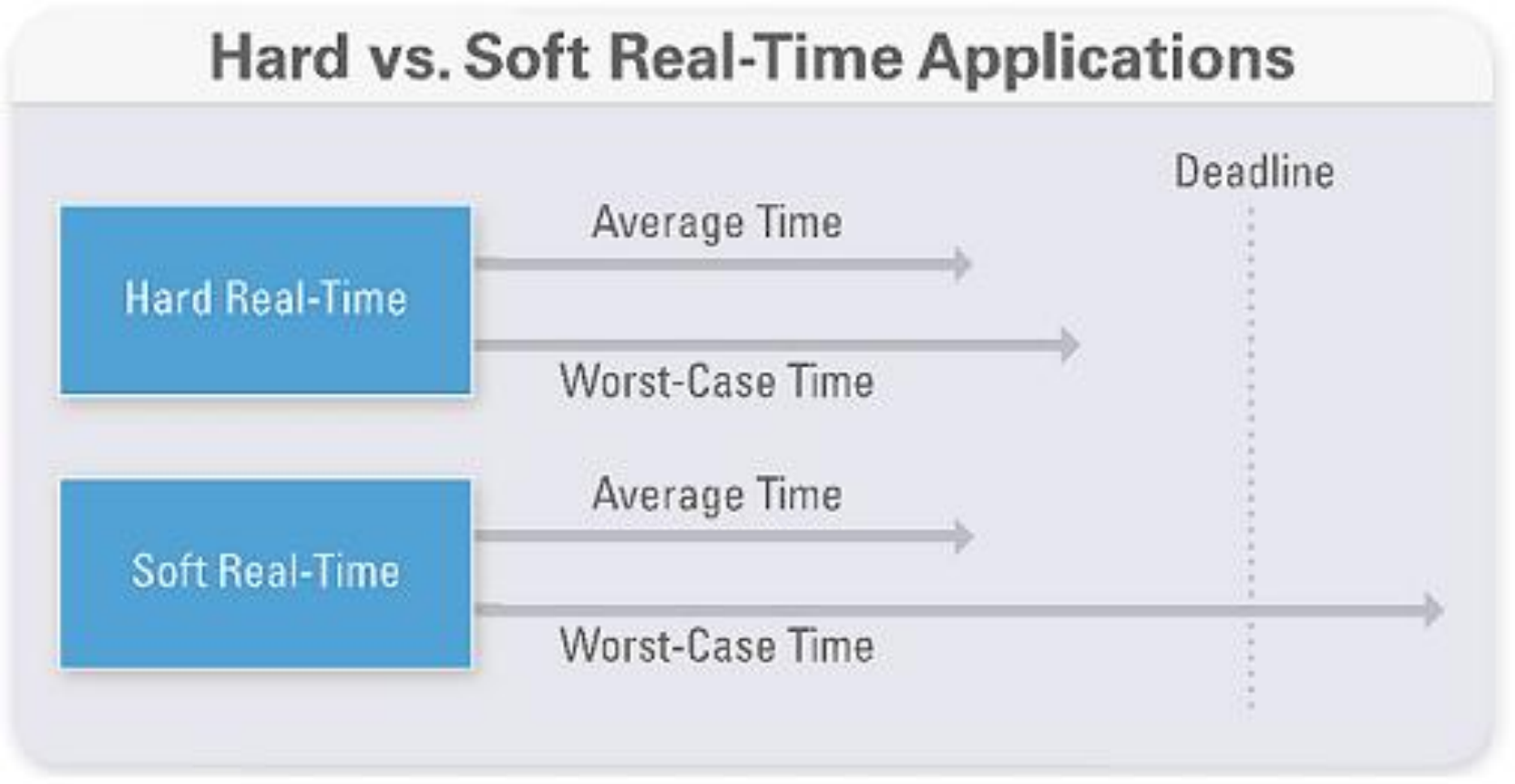
EMAIL : AGUS.MULYANA@EMAIL.UNIKOM.AC.ID

WA : 082116871007

Definition

Real-time embedded 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.

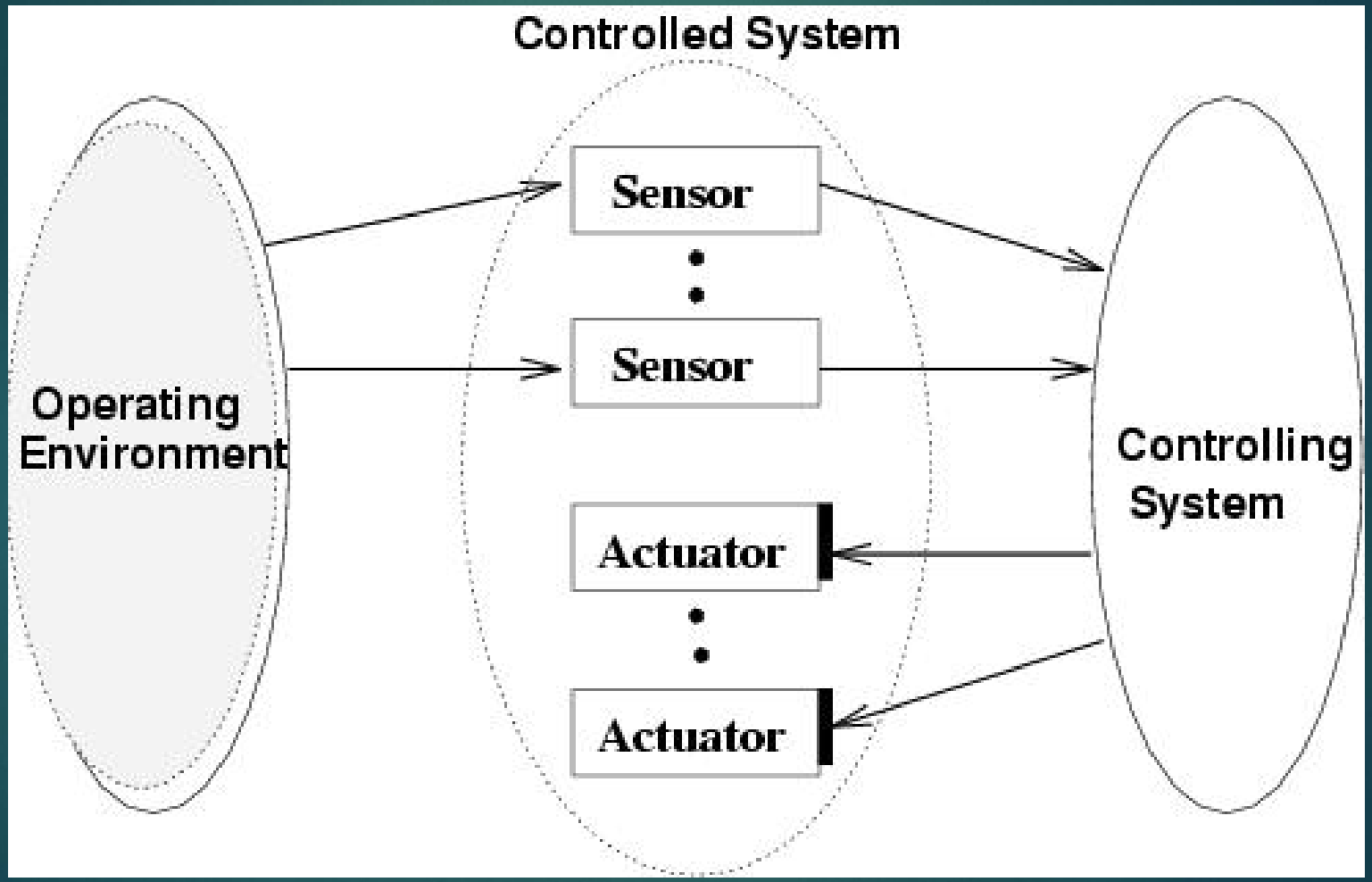
- Hard real-time systems (e.g., Avionic control).
- Firm real-time systems (e.g., Banking).
- Soft real-time systems (e.g., Video on demand).


Hard vs. Soft Real-Time Applications
Deadline
Hard Real-Time
Average Time
Worst-Case Time
Soft Real-Time
Average Time
Worst-Case Time

Deadline

- 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 embedded system



Car example

- ***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.

Car example (contd)

- **Critical tasks:** Steering and breaking.
- **Non-critical tasks:** Turning on radio.
- **Cost** of fulfilling the mission → Efficient solution.
- **Reliability** of the driver → Fault-tolerance needs to be considered.

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.

- **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.

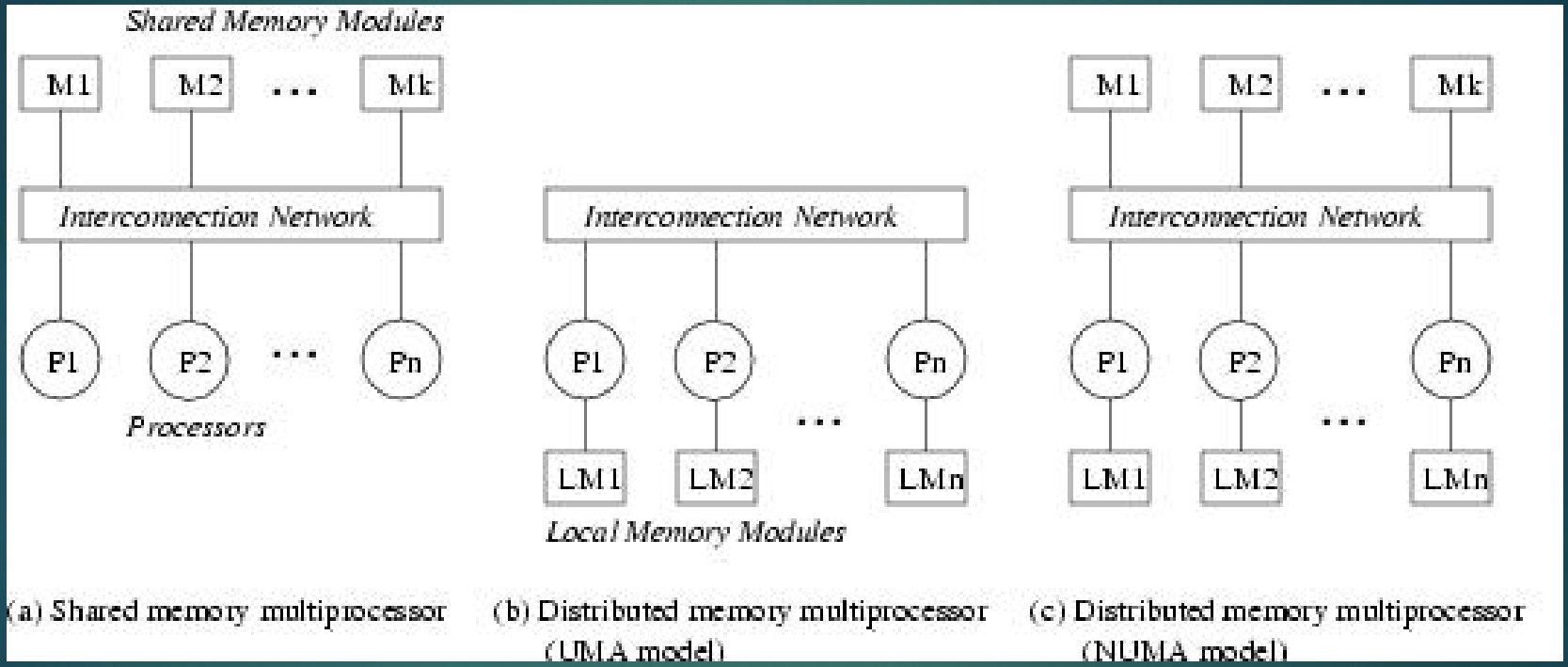
p_i : task period a_i : arrival time r_i : ready time
d_i : deadline c_i : worst case execution time.

Task constraints

- Deadline constraint
- Resource constraints
  - Shared access
  - Exclusive access
- 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

Uniprocessor, multiprocessor, distributed system



(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 been solved in other areas of computer science and engineering.

Major challenge

- The main challenge in real time embedded system design is real time scheduling
- Many existing scheduling techniques
  - Round robin scheduling
  - Planning scheduling
  - Priority scheduling
  - Static scheduling

Preemptive vs Non-preemptive scheduling

- **Preemptive Scheduling**
  - Task execution is preempted and resumed later
  - Preemption occurs to execute higher priority task.
  - Offers higher schedulability
  - Involves higher scheduling overhead due to context switching
- **Non-preemptive Scheduling**
  - Once a task starts executing, it completes its full execution
  - Offers lower schedulability
  - Less overhead due to less context switching

Optimal scheduling -- definition

- A **static scheduling** algorithm is said to be optimal if, for any set of tasks, it always produces a feasible schedule (i.e., a schedule that satisfies the constraints of the tasks) whenever any other algorithm can do so.

- A **dynamic scheduling** algorithm is said to be optimal if it always produces a feasible schedule whenever a static algorithm with complete prior knowledge of all the possible tasks can do so.

- Static scheduling is used for scheduling periodic tasks, whereas dynamic scheduling is used to schedule both periodic and aperiodic tasks.

Summary

- Real-time embedded systems require logical correctness and timeliness.
- Real-time embedded system consists of a controlling system, controlled system, and the environment.
- Real-time systems are classified as: hard, firm, and soft real time systems.
- Tasks are periodic, aperiodic, sparodic.
- The notion of predictability is important in real-time systems and the major challenge is real time scheduling.

Related Topics

1. Real Time OS (RTOS)
2. Parallel Computation
3. Multicore / Multi processor
4. Embedded System
5. Fault Tolerant
6. Distributed System
7. Network Sensor
8. Smart City