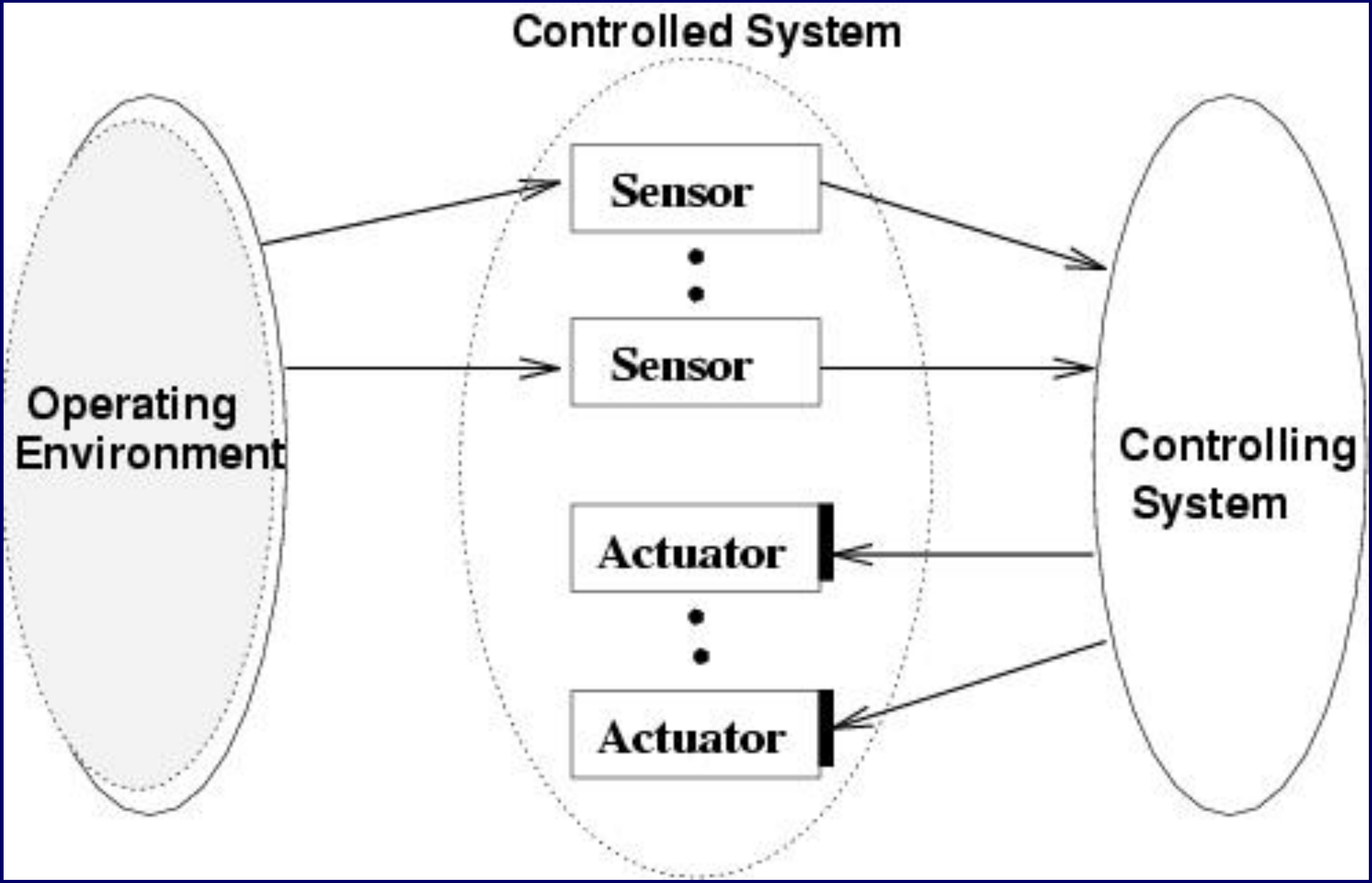
Real-Time Systems

Introduction to Real-Time Systems

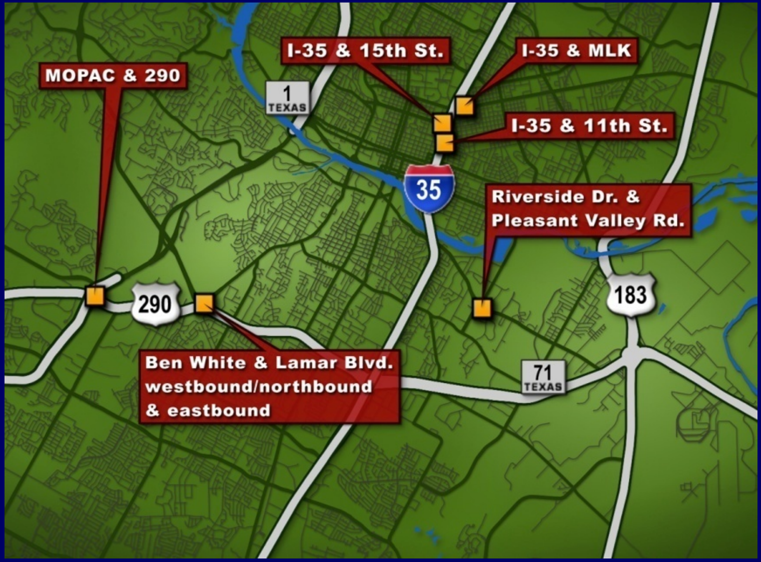
A typical real-time system



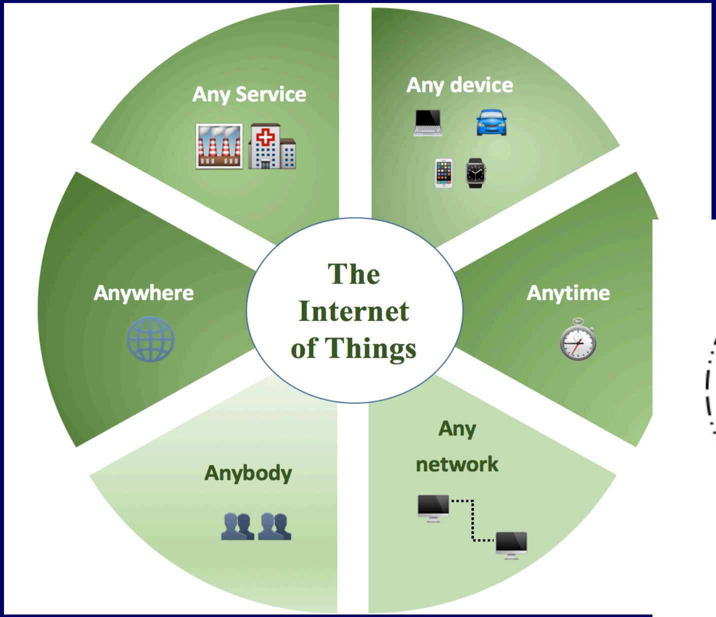
Sample Applications

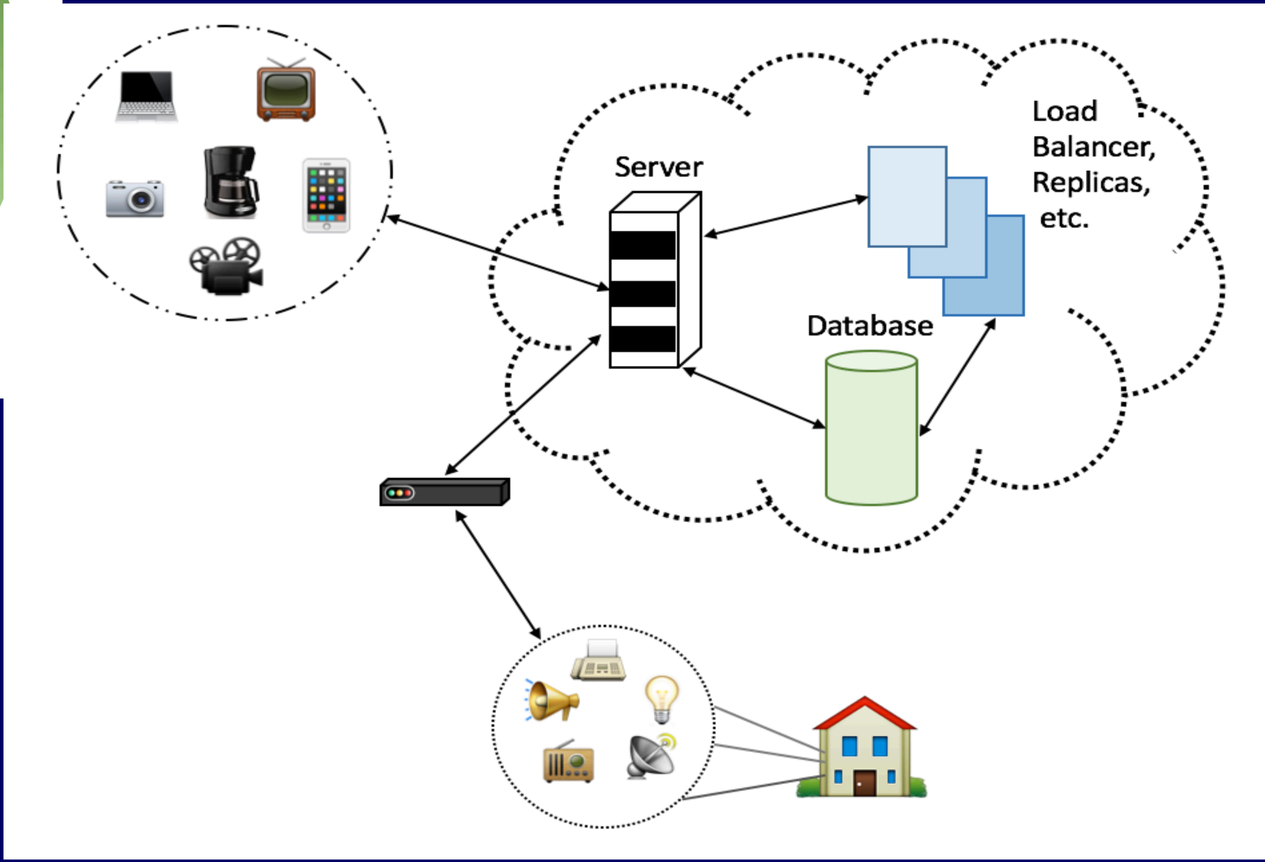
Agile Manufacturing

Traffic control system


MOPAC & 290
I-35 & 15th St.
I-35 & MLK
I-35 & 11th St.
Riverside Dr. & Pleasant Valley Rd.
Ben White & Lamar Blvd. westbound/northbound & eastbound
1 TEXAS
35
290
183
71 TEXAS

Industrial Internet & Internet of Things





Real-time Systems -- Introduction

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., Avionics, Command & Control Systems).
- Firm real-time systems (e.g., Banking, Online transaction processing).
- Soft real-time systems (e.g., Video streaming).

Real-Time Systems -- Introduction

- Hard RT: if missing its deadline may cause catastrophic consequences on the environment under control
- (In the aeronautics domain engine control or aerodynamic control)
- Firm RT: if missing its deadline makes the result useless, but missing does not cause serious damage
- (weather forecast or decisions on stock exchange orders)
- Soft RT: if meeting its deadline is desirable (e.g. for performance reasons) but missing does not cause serious damage
- (Video Streaming)

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, Laser telemeter, Navigation system, Street maps.
- **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 (Workload)

- **Periodic tasks**
  - Time-driven. Characteristics are known *a priori*
  - Task T_i is characterized by (c_i, p_i)
    E.g.: Task monitoring temperature of a patient in an ICU.

- **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**
  – Known minimum inter-arrival time among successive instances of a (periodic) task, rather strictly being periodic.

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

Task constraints

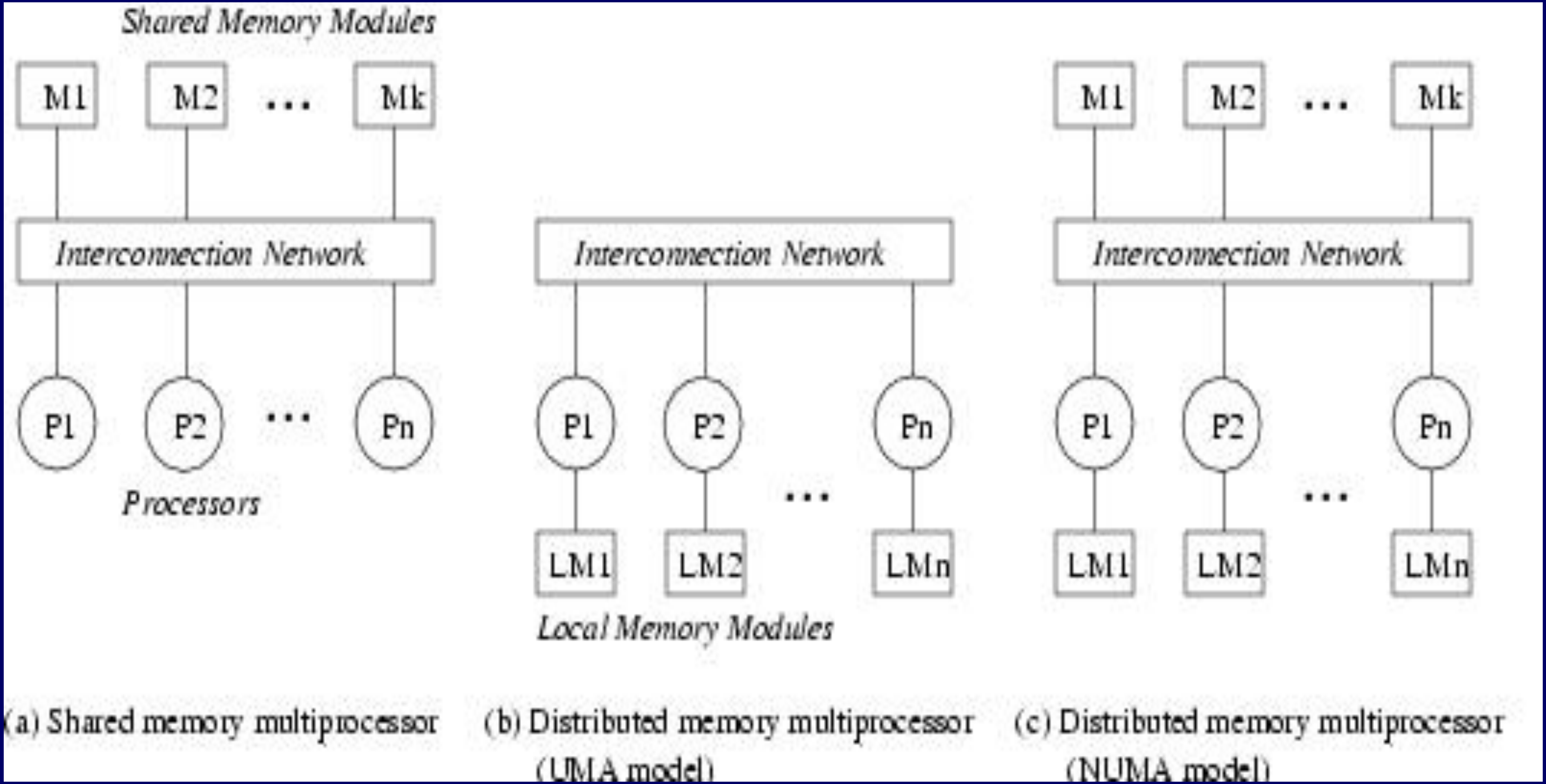
- Deadline constraint
- Resource constraints
  - Shared access (read-read)
  - Exclusive access (write-x)
- Precedence constraints
  - T1 → T2: Task T2 can start executing only after T1 finishes its execution
- Fault-tolerant Requirements
  - To achieve higher reliability for task execution
  - Redundancy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

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.

Computing systems

Uniprocessor, multiprocessor (multicore systems), distributed system, networked control systems



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