MEI, MSI, MI
DI-FCUL, Portugal

Sistemas Embebidos e de Tempo-Real

Scheduling

Prof. António Casimiro

Summary

- Scheduling
  - Approaches
  - Definitions
  - Schedulability tests
  - Rate Monotonic scheduling algorithm
  - Earliest Deadline First scheduling algorithm

Scheduling

- **Scheduling:** defines how a resource is shared by the different tasks in the system, according to a given policy

- Resource examples:
  - Processor shared by concurrent tasks
  - Network used by several nodes
  - Shared memory region used by several tasks

- In a real-time context the scheduling paradigm is concerned with using the available resources in the right way in order to help the system (programs, algorithms,…) satisfying timeliness requirements

The scheduling problem

- A set of tasks must be executed such that all time-critical tasks must meet their deadlines
- Tasks need data and processing resources
- How to allocate available resources so that all timing requirements are satisfied?

Illustrating the scheduling problem

Scheduling approaches

- Time-sharing policy
  - Manage resources with fairness and justice
  - Achieve a reasonable performance for the different applications

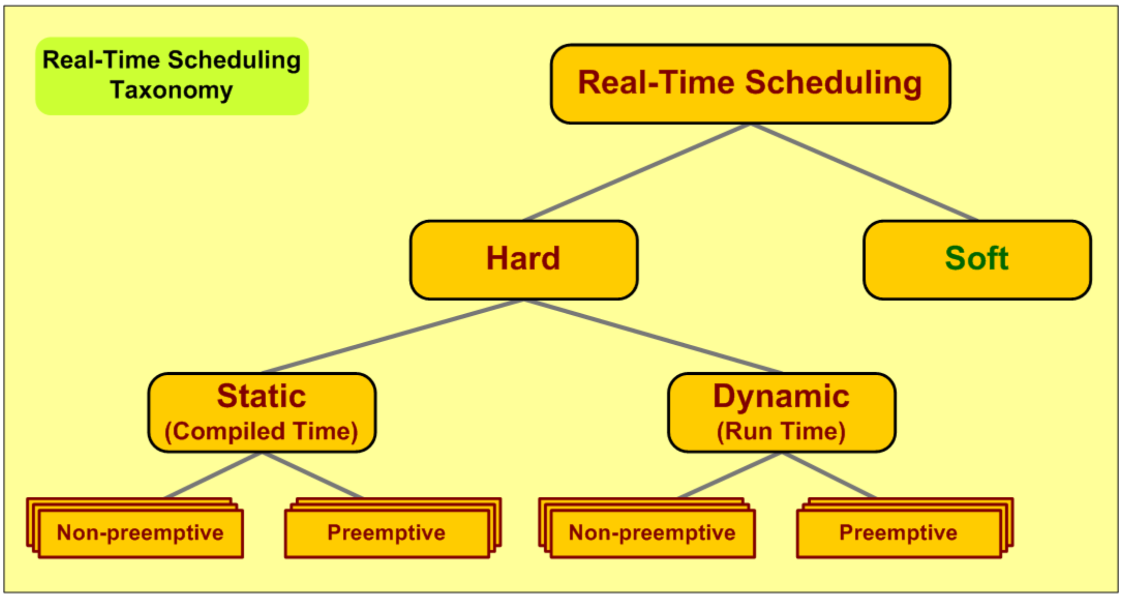
- Example: UNIX scheduler
  - The priority of a process is lowered with an increasing use of resources
  - Allocation of the processor is based on timeslots

Scheduling approaches

- Real-time scheduling
  - The primary objective is to achieve timely execution of tasks (to meet deadlines)
  - All RT applications should meet their deadlines
  - Preemption with basis on event occurrence requiring a timely processing (e.g. making use of priorities)

- Example: EDF
  - The priority of a process depends on the deadline proximity

Scheduling approaches



Scheduling approaches

Static *vs* dynamic scheduling

- Static scheduling:
  - According to a pre-defined plan
  - Assumes that it is possible to predict all timings, such as execution times, request times, resource conflicts, and so forth
- Dynamic scheduling:
  - Computes the schedule at runtime
  - Scheduling decisions based on the analysis of a list of tasks ready for execution

Scheduling approaches

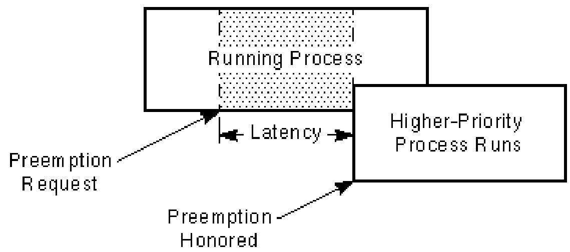
Fixed *vs* dynamic priorities

- Definition of priority:
  - An indication of the importance (or urgency) of a task
  - Often used by schedulers for taking scheduling decisions
- Fixed priority:
  - Priority of a task is statically defined and is unchanged in run-time
  - Common in real-time operating system kernels
- Dynamic priority:
  - Priority may change during execution, to reflect the varying importance of tasks, e.g. as deadlines approach

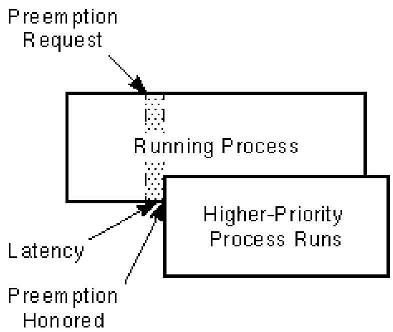
Scheduling approaches

Preemptive *vs* non-preemptive

- Non-preemption:
  - A task is not interrupted unless it runs to completion or deliberately releases the resource (e.g. the processor)



- Preemption:
  - A task can be interrupted, usually by another task of higher priority



Scheduling approaches

Centralized *vs* distributed

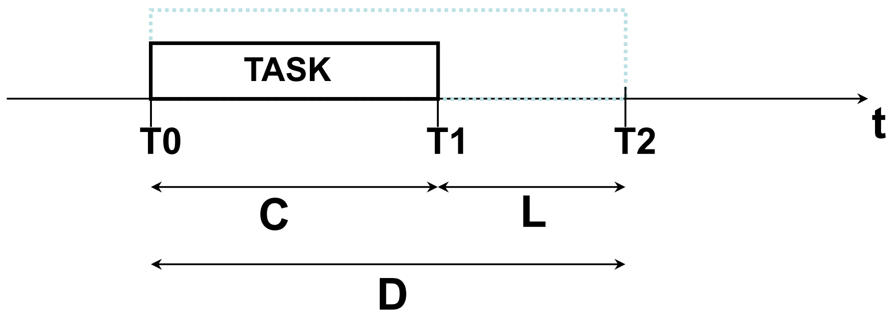
- Centralized:
  - Scheduling decisions are taken by a single entity

- Distributed:
  - Scheduling decisions are taken based on different entities at different nodes of the system

Scheduling
Generic definitions

- Feasible schedule
  - Can be executed with the available resources
- Schedulability testing
  - Determine the existence (sufficient conditions) or inexistence (lack of necessary conditions) of a schedule for a given problem
- Optimal scheduler
  - When it always finds a feasible schedule if one exists

Scheduling
Typical parameters



- Parameters:
  - **C**: Worst Case Execution Time (a.k.a. capacity or computation time)
  - **D**: Deadline
  - **L**: Laxity, L = D-C
  - **U**: Utilization factor, U = C/D

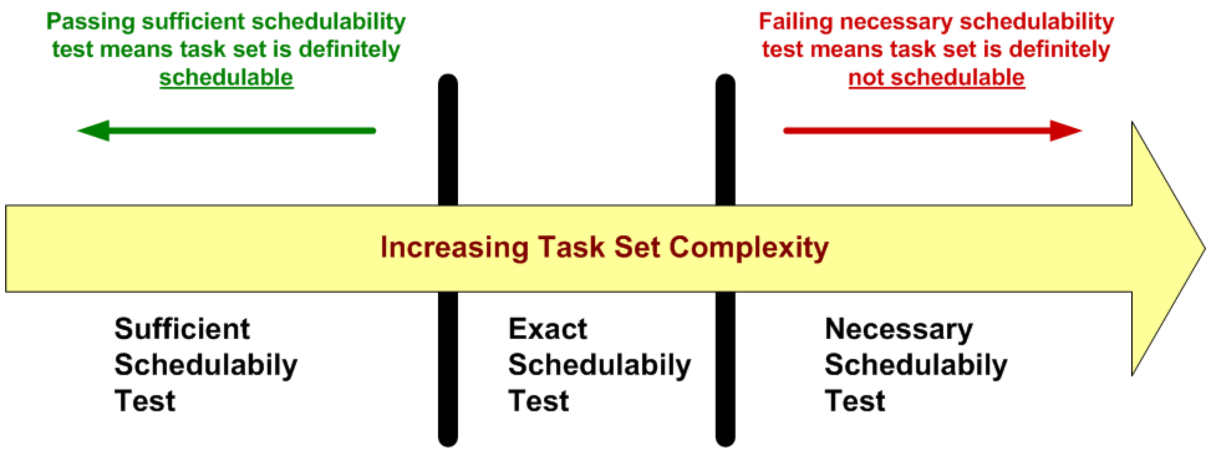
Scheduling

Schedulability tests

- Sufficient:
  – Passing it indicates that the task set is schedulable
  – Nothing is known if the test fails
- Necessary:
  – Failing it indicates that the task set is not schedulable
  – Nothing is known if the test succeeds
- Exact:
  – Passing indicates schedulability
  – Failing indicates non-schedulability

Scheduling

Schedulability tests



- Examples of schedulability tests:
  – $D - C \geq 0$
  – $\sum Ci/Di \leq 1$, for a task set

Scheduling
Response-Time Analysis

- Alternative approach for schedulability testing:
  - Response-time-based analysis
- Methodology:
  - WCET of each task (C_i) must be known
  - Derive the actual worst-case termination time (or response-time) for each task of the task set
  - Must take into account interference time (T_{int})
  - Compare the response-times with task deadlines to check if deadlines are always met
- Response-time-based analysis is an **exact test**

Scheduling
Response-Time Analysis

$$R_i^0 = c_i$$

$$R_i^k = c_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i^{k-1}}{T_j} \right\rceil \cdot c_j$$

Scheduling

Considering events with periodic distribution

- Periodic tasks
  - Triggered by time (*time-driven*), at regular intervals
  - Timeliness parameters are attributes **known** a priori
  - Task *i* is characterized by (T_i, c_i, d_i, r_i, Φ_i)

    T_i : period

    c_i : worst-case execution time (WCET)

    d_i : deadline

    r_i : release (ready) time

    Φ_i : first release time (phase)

  - Example: monitoring the temperature of a liquid in a storage tank

Scheduling

Considering events with aperiodic distribution

- Aperiodic tasks
  - Tasks are triggered by events (*event-driven*)
  - Timeliness parameters are **unknown** *a priori*
  - Task *i* is characterized by (a_i, c_i, d_i, r_i)

    a_i : interval between activations (unbounded)

    c_i : worst-case execution time (WCET)

    d_i : deadline

    r_i : release (ready) time

  - Example: temperature alarm

Scheduling

Considering events with sporadic distribution

- Sporadic tasks
  - Tasks are triggered by events (*event-driven*)
  - Timeliness parameters are **known** a priori
  - Task *i* is characterized by (m_i, c_i, d_i, r_i)

    m_i : minimum inter-arrival time (lower bound is known)

    c_i : worst-case execution time (WCET)

    d_i : deadline

    r_i : release (ready) time

  - Example: temperature alarm

Scheduling

Utilization factor

- Utilization factor: resource utilization by task *i*:
  - Periodic tasks:

$$U_i = c_i / T_i$$

  - Sporadic tasks:

$$U_i = c_i / m_i$$

Scheduling
Resource utilization

- Resource utilization: utilization of a resource by a given task set $\{T_1, T_2, \ldots, T_n\}$

$$U = \sum_{i=1}^{n} U_i$$

  - The condition $U \leq r$ must hold, with r being the number of available resources
  - Example: r = number of processors

Static scheduling

- Performed off-line, based on the search for sufficient schedulability conditions
- Requires periodic/sporadic distributions
- Time-triggered
  - Periodic event handling
  - Periodic release of processing tasks
  - Schedulability: event handling time plus task duration should be shorter than the deadline or event period
- Handling uncertainty:
  - Sporadic to periodic requests transformation – allowing sporadic events to be handled by schedules established and analyzed off-line
  - Mode changes, with a off-line static schedule defined for each mode

Dynamic Scheduling

- Rate Monotonic (RM)
  - Preemptive, based on fixed priorities
  - For periodic tasks, priority is inversely proportional to period
- Deadline Monotonic (DM)
  - Preemptive, based on fixed priorities
  - For periodic tasks, priority is inversely proportional to deadline
- Earliest-Deadline-First (EDF)
  - Preemptive, based on dynamic priorities
  - Priority is inversely proportional to deadline
- Least-Laxity (LL)
  - Preemptive, based on dynamic priorities
  - Priority is inversely proportional to laxity
- First-Come-First-Served (FCFS)

Rate Monotonic (RM)

- Static scheduling in the sense priorities are fixed, defined according to task periods
- The task with the lower period is assigned the highest priority
- **Preemptive** scheduling of tasks
- Optimal scheduling:
  - No algorithm from the same class (i.e. fixed priority) is able to schedule a task set which is not feasible also with rate monotonic

Rate Monotonic

- Assumptions:
  - Tasks *i* independent and periodic
  - Deadline equal to the period (d_i=T_i)
  - Bounded and known worst case execution time (c_i)
  - Context switching time is negligible

Rate Monotonic
Feasibility test

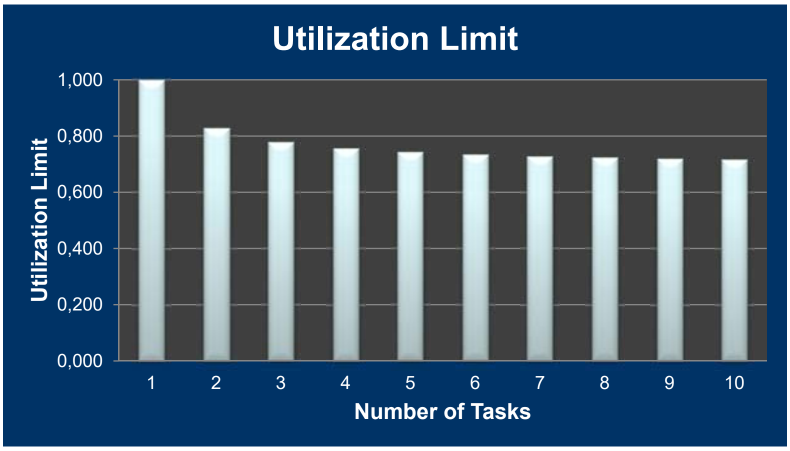
- Sufficient test: given a task set of independent and periodic tasks $\{T_1, T_2, \ldots, T_n\}$, the utilization limit is:

$$U = \sum_{i=1}^{n} c_i / T_i \leq n \cdot (2^{1/n} - 1)$$

  - For high *n*, the utilization factor converges to 0.69

- **Recall**:
  - Passing a sufficient test means the task set is schedulable
  - Failing a sufficient test does not mean the task set is not schedulable

Rate Monotonic
Feasibility test



Number of Tasks	1	2	3	4	5	6	7	8	9	10
Utilization Limit	1,000	0,828	0,780	0,757	0,743	0,735	0,729	0,724	0,721	0,718
Number of Tasks	15	20	25	30	35	40	45	50	55	60
Utilization Limit	0,709	0,705	0,703	0,701	0,700	0,699	0,699	0,698	0,698	0,697

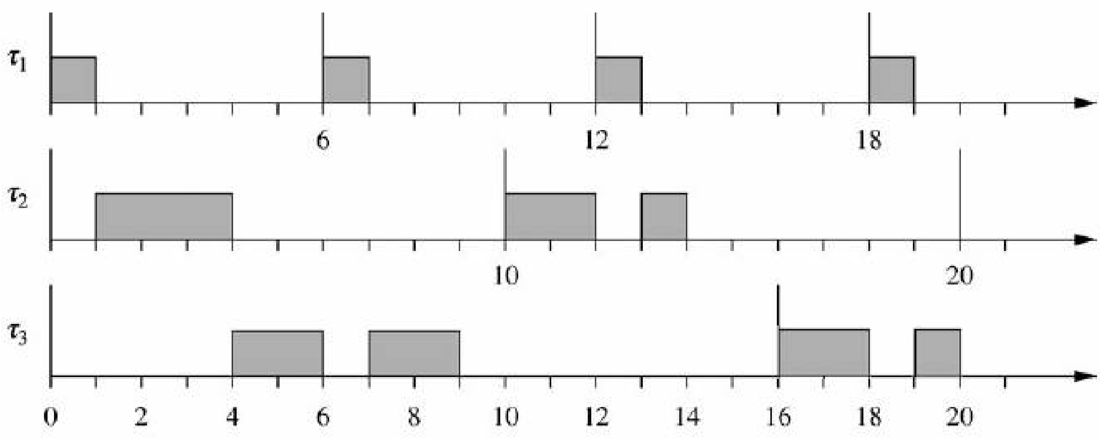
Rate Monotonic
Example

RM Example 1	Periodic Tasks	Period T_i	Deadline D_i	Execution Time - C_i	Priority	Utilization	Utilization Limit	Schedule
	A	6	6	1	3	0,167		
	B	10	10	3	2	0,300		
	C	16	16	4	1	0,250		
Number of tasks	3					0,717	0,780	✓
Clock (Max. granularity)		1						
Period (Max. granularity)		2						
Minimum Period		6						
Base Period		240						

← mmc {6,10,16}

Rate Monotonic

Example: temporal diagram



Can we conclude that the task set is schedulable by simply observing this temporal diagram? Why not?

Rate Monotonic

Example: response-time analysis

$$R_1^0 = 1$$

$$R_2^0 = 3$$

$$R_2^1 = 3 + \left\lceil \frac{3}{6} \right\rceil \cdot 1 = 4$$

$$R_2^2 = 3 + \left\lceil \frac{4}{6} \right\rceil \cdot 1 = 4$$

$$R_3^0 = 7$$

$$R_3^1 = 7 + \left\lceil \frac{7}{6} \right\rceil \cdot 1 + \left\lceil \frac{7}{10} \right\rceil \cdot 3 = 12$$

$$R_3^2 = 7 + \left\lceil \frac{12}{6} \right\rceil \cdot 1 + \left\lceil \frac{12}{10} \right\rceil \cdot 3 = 15$$

$$R_3^3 = 7 + \left\lceil \frac{15}{6} \right\rceil \cdot 1 + \left\lceil \frac{15}{10} \right\rceil \cdot 3 = 16$$

$$R_3^4 = 7 + \left\lceil \frac{16}{6} \right\rceil \cdot 1 + \left\lceil \frac{16}{10} \right\rceil \cdot 3 = 16$$

Schedule is feasible

Rate Monotonic

Feasibility for harmonic tasks

- Exact test:
  - Can be applied for a given task set of independent and periodic tasks *{T_1, T_2,…,T_n}*, *iff all periods are multiple of each other (harmonic period condition)*

$$U = \sum_{i=1}^{n} (c_i / T_i) \leq 1$$

Rate Monotonic

Example: harmonic tasks

RM Example 2	Periodic Tasks	Period T_i	Deadline D_i	Execution Time - C_i	Priority	Utilization	Utilization Limit	Schedule
	A	4	4	2	3	0,500		
	B	8	8	2	2	0,250		
	C	16	16	4	1	0,250		
Number of tasks	3					1,000	0,780	✗
Clock (Max. granularity)		2			Harmonic Bound	1,000	1,000	✓
Period (Max. granularity)		4						
Minimum Period		4						
Base Period		16						

Earliest Deadline First (EDF)

- Dynamic scheduling: task ordering (priority assignment) algorithm is executed upon new scheduling events (e.g. task arrival, task completion)
- Task ordering inversely proportional to the (absolute) deadline: the highest priority is assigned to the task with the lower deadline
- Assumes independent tasks
- Applicable to periodic and non-periodic tasks
- Optimal scheduling:
  - Class of scheduling algorithms with dynamic priorities

Earliest Deadline First

Feasibility test

- Exact test: given a set of periodic and independent tasks $\{T_1, T_2, \ldots, T_n\}$:

$$U = \sum_{i=1}^{n} c_i / T_i \leq 1$$

- **Recall**:
  - Passing an exact test means the task set is schedulable
  - Failing an exact test means the task set is not schedulable

Earliest Deadline First
Example

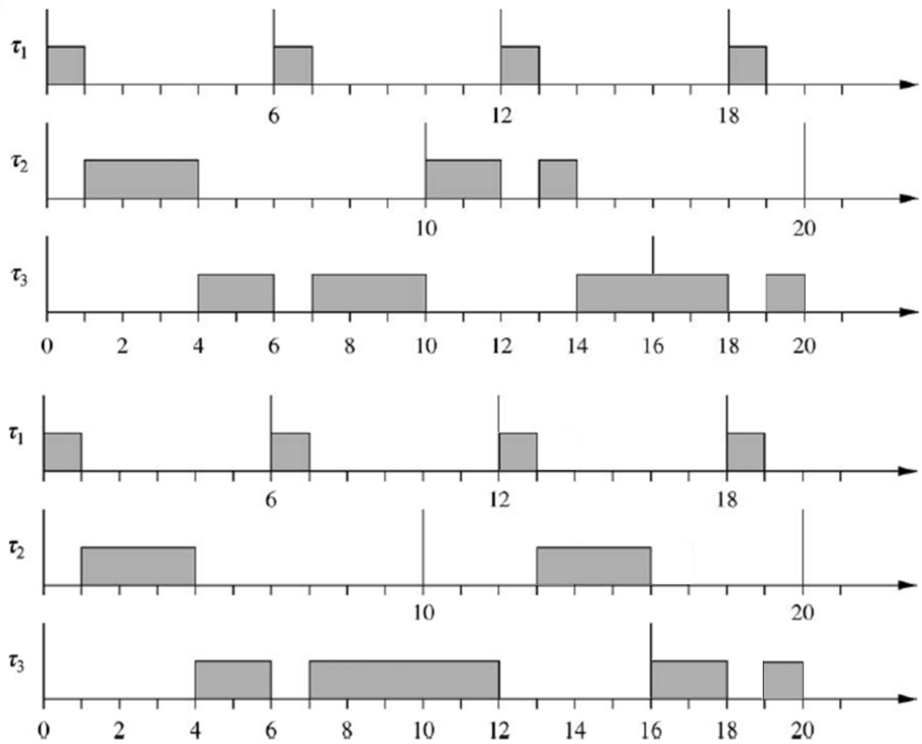
EDF Example	Periodic Tasks	Period T_i	Deadline D_i	Execution Time - C_i	Priority	Utilization	Utilization Limit	Schedule
	A	6	6	1	Dynamic	0,167		
	B	10	10	3	Dynamic	0,300		
	C	16	16	7	Dynamic	0,438		
Number of tasks	3					0,904	1,000	✓
Clock (Max. granularity)		1						
Period (Max. granularity)		2						
Minimum Period		6						
Base Period		240						

Would fail sufficient test for RM scheduling: 0,904 > 0,780

EDF vs. RM Comparison

RM

EDF



EDF vs. RM Comparison

Response-time analysis for RM scheduling

$$R_1^0 = 1$$

$$R_2^0 = 3$$

$$R_2^1 = 3 + \left\lceil \frac{3}{6} \right\rceil \cdot 1 = 4$$

$$R_2^2 = 3 + \left\lceil \frac{4}{6} \right\rceil \cdot 1 = 4$$

$$R_3^0 = 7$$

$$R_3^1 = 7 + \left\lceil \frac{7}{6} \right\rceil \cdot 1 + \left\lceil \frac{7}{10} \right\rceil \cdot 3 = 12$$

$$R_3^2 = 7 + \left\lceil \frac{12}{6} \right\rceil \cdot 1 + \left\lceil \frac{12}{10} \right\rceil \cdot 3 = 15$$

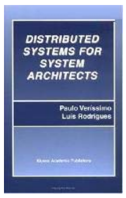
$$R_3^3 = 7 + \left\lceil \frac{15}{6} \right\rceil \cdot 1 + \left\lceil \frac{15}{10} \right\rceil \cdot 3 = 16$$

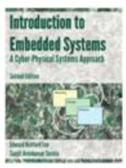
$$R_3^4 = 7 + \left\lceil \frac{16}{6} \right\rceil \cdot 1 + \left\lceil \frac{16}{10} \right\rceil \cdot 3 = 16$$

Schedule is feasible

Bibliography

- Textbook:
  - P. Veríssimo and L. Rodrigues, Distributed Systems for System Architects, Kluwer Academic Publishers, 2001, 650pp., Part III – Real-Time.
    - Section 12.7
  - Hermann Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, 1997.
    - Chapter 10
  - Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition, http://LeeSeshia.org
    - Chapter 12





Thank you for your participation.
Final questions?

https://moodle.ciencias.ulisboa.pt
Sistemas Embebidos e de Tempo-Real

The Persistence of Memory
Salvador Dali, 1931