Real-Time Systems & Fault Tolerance

Flávia Maristela

Instituto Federal da Bahia)
Especialização em Computação Distribuída e Ubíqua (ECDU)

Salvador, Outubro de 2013

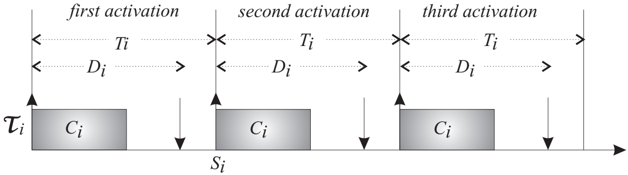
Schedule

- A real-time system is commonly defined as a set of n independent tasks:

$$\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\} \tag{1}$$

- A task can be defined as a basic system unit scheduled for execution
- Each real-time task has important attributes, such as:
    1. Arrival Time (Release Time) (r_i)
    2. Period (T_i)
    3. Absolute Deadline (d_i)
    4. Relative Deadline (D_i)
    5. Execution Cost(C_i)

Tasks Attributes



Tasks Activation

- Real-time tasks can be classified according to their *activation period* in
  1. Periodic
  2. Aperiodic
  3. Sporadic

Tasks Activation

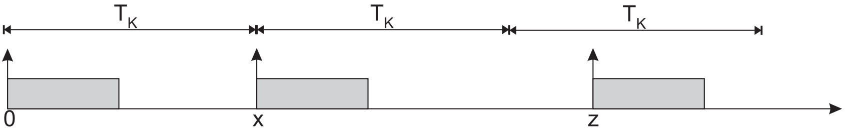
- **Periodic**: activation occurs in an infinite sequence, with a single activation per period (time-triggered on a regular basis)

Tasks Activation

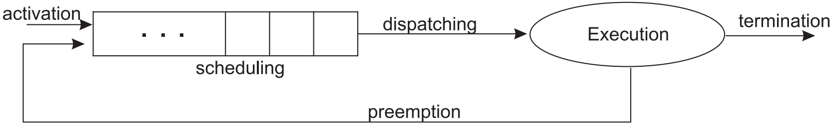
- **Aperiodic**: activation cannot be predicted (random activation time instants)

Tasks Activation

- **Sporadic**: aperiodic tasks whose minimum interval between two consecutive activations is known



General Approach



Example

Assume a control system responsible for landing an airplane. This system can be described by two periodic main tasks $\Gamma = \{\tau_1, \tau_2\}$. *Tasks may be described as:*

1. *Open landing gear*
2. *Land*

Briefly,

- ***Task 1****: activated at each 2 time units; deadline equals 2; takes 0.8 time units at CPU in each activation*
- ***Task 2****: activated at each 5 time units; deadline equals 5; takes 1 time unit at CPU in each activation*

Some Important schedule characteristics

- Considering these conditions, we may ask: “will the airplane land safely”?

Some Important schedule characteristics

- Considering these conditions, we may ask: "will the airplane land safely"?
- In order to guarantee that deadlines are met, tasks need to be ordered for execution according to some heuristic (**scheduling algorithm**)

Some Important schedule characteristics

- Considering these conditions, we may ask: “will the airplane land safely”?
- In order to guarantee that deadlines are met, tasks need to be ordered for execution according to some heuristic (**scheduling algorithm**)
- Mostly known scheduling heuristics
  - Deadline Monotonic (DM)
  - Rate Monotonic (RM)
  - Earliest Deadline First (EDF)

Taxonomy of Scheduling Algorithms

Optimal *vs.* Heuristic

An algorithm is said to be **optimal** if it minimizes some given cost function defined over the task set. On the other hand, it is characterized as an **heuristic** if it is guided by an **heuristic** function (tends toward an optimal function but does not guarantee finding it)

Taxonomy of Scheduling Algorithms

Preemptive *vs.* Non-Preemptive

In **preemptive** scheduling running tasks can be interrupted at any time to assign the processor to another task, according to its periodic. In **non-preemptive**, on the contrary, tasks are executed until completion

Taxonomy of Scheduling Algorithms

Static *vs.* Dynamic

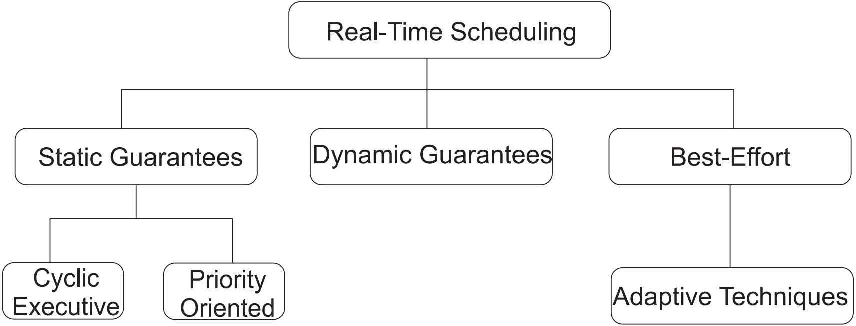
In **static** algorithms, scheduling decisions are based on fixed parameters (assigned to tasks before its activation). For **dynamic** algorithms scheduling decisions are based on parameters that may change during system evolution

Taxonomy of Scheduling Algorithms

Off-line *vs.* Online

A scheduling algorithm is said to be **off-line** if it is executed on the entire task set before tasks activation. For **online** algorithms scheduling decisions are taken at runtime (whenever a new task arrives or an active task terminates

Scheduling Policies



Guaranteed Scheduling Algorithms - Static Systems

- Basically, we can identify two main approaches:
  - Cyclic Executive
  - Priority Oriented
- Where do we use them?
  - Embedded Applications
  - Control of complex production processes
  - Railway switching systems

Guaranteed Scheduling Algorithms - Cyclic Executive

- Characteristics
  - Task set is fixed and known a priori
  - Tasks activation are computed off-line
  - Timeliness Guarantees are given off-line
  - Schedulability Analysis is implicit
  - Entire schedule is stored in a table (which contains appropriated dispatching order)
  - At runtime dispatcher only removes tasks from table
- Advantages
  - At run time, overhead does not depend on the complexity of the scheduling algorithm
- Disadvantages
  - Resulting system is quite inflexible to environmental changes

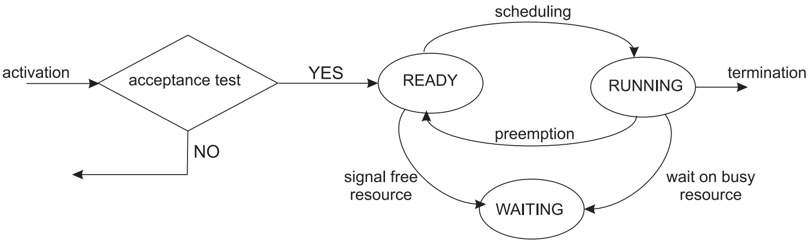
Guaranteed Scheduling Algorithms - Priority Oriented

- Characteristics
  - More flexible than Cyclic Executive
  - Task set is fixed and known a priori
  - Schedulability Analysis is off-line
  - Tasks activation are computed online
  - Tasks priorities are defined by scheduling policies
    - fixed (or static): online static
    - dynamic: online dynamic
  - Based on worst-case decisions, which are only assessed in schedulability analysis
- Advantages
  - Can be used in a large number of applications
  - Does not need to compute the whole schedule if a new task arrive (computed offline)

Guaranteed Scheduling Algorithms - Dynamic Systems

- Characteristics
  - Tasks computational cost and arrival time are not known a priori
  - Tasks can be created at runtime
  - Guarantee must be done online (on every task creation)
  - Tasks go through an acceptance test
- Advantages
  - Potential overload situations can be detected in advance (negative effects to the system can be avoided)
- Disadvantages
  - It is based on worst-case assumptions
  - Pessimistic assumptions made may unnecessarily cause tasks rejections

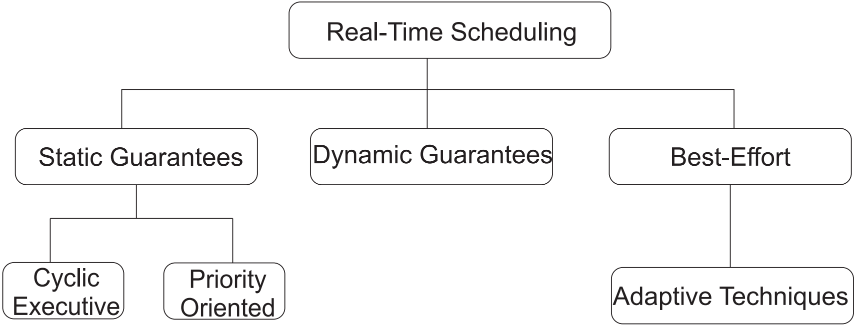
Guaranteed Scheduling Algorithms - Dynamic Systems



Best-Effort Algorithms

- Characteristics
  - Tasks computational cost and arrival time are not known a priori
  - Typically used for multimedia applications
  - Timing constraints depends on users desired Quality of Service (QoS)
  - "Tries to do the best to meet tasks deadlines, but there is no guarantee of finding a feasible schedule"
- Advantages
  - Performs much better that guaranteed-based approach in average case
- Disadvantages
  - Feasibility is not checked → Enqueued tasks can be aborted
  - Tasks are rejected only in overloaded conditions

Scheduling Policies



Scheduling Policies

- Rate Monotonic
- Deadline Monotonic
- Earliest Deadline First

Rate Monotonic - RM

Priority Definition

Tasks with smaller **period** are scheduled first

- Preemptive schedulers
- Tasks with fixed priorities
- Online and static
- Optimal for fixed priorities algorithms in the same class

Rate Monotonic - RM

- Rate monotonic assumes:
  - A1 - Periodic tasks
  - A2 - Tasks deadlines equal their period
  - A3 - Tasks are independent
  - A4 - Tasks computational cost are constant and worst-case based
  - A5 - Switch context costs are negligible

Rate Monotonic - RM

- Rate monotonic assumes:
  - A1 - Periodic tasks
  - A2 - Tasks deadlines equal their period
  - A3 - Tasks are independent
  - A4 - Tasks computational cost are constant and worst-case based
  - A5 - Switch context costs are negligible
- LET'S GO TO THE BOARD

Earliest Deadline First - EDF

Priority Definition

Tasks with smaller **absolute deadline** are scheduled first

- Preemptive schedulers
- Tasks with dynamic priorities
- Online and Dynamic
- Optimal for dynamic priorities algorithms

Earliest Deadline First - EDF

- EDF assumes
  - A1 - Periodic Tasks
  - A2 - Tasks deadlines equal their period
  - A3 - Tasks are independent
  - A4 - Tasks computational costs are constant and worst-case based
  - A5 - Switch context costs are negligible

Earliest Deadline First - EDF

- EDF assumes
  - A1 - Periodic Tasks
  - A2 - Tasks deadlines equal their period
  - A3 - Tasks are independent
  - A4 - Tasks computational costs are constant and worst-case based
  - A5 - Switch context costs are negligible
- LET'S GO TO THE BOARD

Deadline Monotonic - DM

Priority Definition

Tasks with smaller **relative deadline** (D_i) are scheduled first

- Preemptive schedulers
- Tasks with fixed priorities
- Online and static
- Optimal for fixed priorities algorithms in the same class

Deadline Monotonic - DM

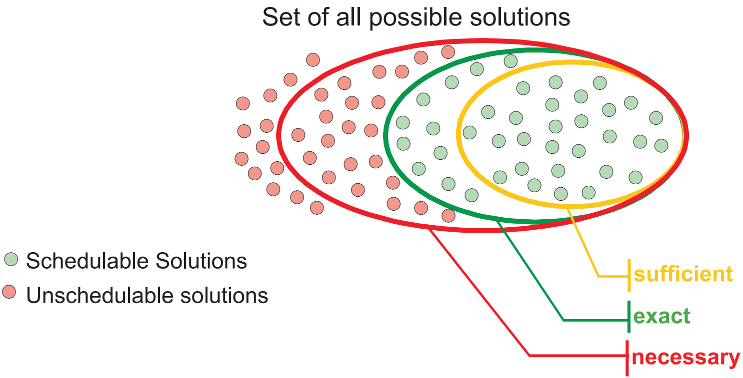
- Deadline Monotonic assumes:
  - A1 - Periodic Tasks
  - A2 - Tasks deadlines are smaller than or equal to their period
  - A3 - Tasks are independent
  - A4 - Tasks computational costs are constant and worst-case based
  - A5 - Switch context costs are negligible

Deadline Monotonic - DM

- Deadline Monotonic assumes:
  - A1 - Periodic Tasks
  - A2 - Tasks deadlines are smaller than or equal to their period
  - A3 - Tasks are independent
  - A4 - Tasks computational costs are constant and worst-case based
  - A5 - Switch context costs are negligible
- LET'S GO TO THE BOARD

1. System
2. Tasks
3. Scheduling
4. Scheduling Policies
5. Schedulability Analysis

- Aims at assessing system timeliness through simple mathematical verification
- Tests can be:
  1. **necessary**: identifies not only schedulable task sets, but also some unschedulable ones
  2. **sufficient**: identifies a subset os schedulable task sets
  3. **exact**: identifies all shcedulable task sets


Set of all possible solutions
Schedulable Solutions
Unschedulable solutions
sufficient
exact
necessary

Processor Utilization Analysis

- Off-line analysis
- Based on processor utilization

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2}$$

- Sufficient Analysis

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1) \text{ , where } n \text{ is the number of tasks} \tag{3}$$

Processor Utilization Analysis and Rate Monotonic

- When $n \to \infty$ then $U \approx 69\%$

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1) \tag{4}$$

- Is $U \approx 69\%$ acceptable?
- What are the implications?
- For harmonic task sets, $U \approx 100\%$ (sufficient and necessary)

Processor Utilization Analysis and Earliest Deadline First

- Necessary and Sufficient Analysis
- When $n \to \infty$ then $U = 100\%$

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1) \text{ , where } n \text{ is the number of tasks} \qquad (5)$$

Response Time Analysis

- Exact and Iterative Test
- Off-line analysis
- Test stops when there is a convergence in the response time
- If $\forall \tau_i \in \Gamma, R_i \leq D_i \Rightarrow$ task set is said to be schedulable

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \qquad (6)$$