Real-Time Systems & Fault Tolerance

Flávia Maristela

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

Salvador, Outubro de 2013

Schedule

In the context of a real-time system, what happens if a task fails?

- Any computational system can POTENTIALLY fail.
  - What happens if a given task $\tau_i \in \Gamma$ fail?
  - How faults can prevent tasks to meet their *deadlines*?
  - What can be done so that the system can survive, even in the presence of faults?

- How faults affect different real-time systems?

- How faults affect different real-time systems?
- **Hard** Real-Time Systems: a deadline miss **may have** "catastrophic"consequences

- How faults affect different real-time systems?
- **Hard** Real-Time Systems: a deadline miss **may have** "catastrophic"consequences
- **Soft** Real-Time Systems: most of the time, a deadline miss causes performance degradation.

How faults can affect real-time systems most important characteristics?

How faults can affect real-time systems most important characteristics?

- Timeliness: results have also to be correct in time domain (besides logical domain)

How faults can affect real-time systems most important characteristics?

- Timeliness: results have also to be correct in time domain (besides logical domain)
- Efficiency: related to the efficiency in managing available resources, specially in embedded devices (space, weight, energy, memory and computational power)

How faults can affect real-time systems most important characteristics?

- Timeliness: results have also to be correct in time domain (besides logical domain)
- Efficiency: related to the efficiency in managing available resources, specially in embedded devices (space, weight, energy, memory and computational power)
- Robustness: systems must be able to support eventual overloads

Predictability Definition

How faults can affect real-time systems predictability?

Definition

"(...) the system should be able to predict the evolution of tasks and guarantee in advance that all critical timing constraints will be met."

- We focus on **scheduling** aspects to improve fault tolerance
  $\Rightarrow$ scheduling decisions

Tasks Activation

- **Periodic**: activation occurs in an infinite sequence, with a single activation per period (time-triggered on a regular basis)
- **Aperiodic**: activation cannot be predicted (random activation time instants)
- **Sporadic**: aperiodic tasks whose minimum interval between two consecutive activations is known

Task Attributes for Fault Tolerance

- Fault-Tolerant real-time systems include a recovery action, which is modeled as a special task
- Consider a real-time system composed of a set of n tasks $\Gamma = \{\tau_1, \ldots, \tau_n\}$. For such a system, a given task τ_i has a specific attribute, related to fault occurrence:
  - Arrival Time (Release Time) (R_i)
  - Period (T_i)
  - Absolute Deadline (d_i)
  - Relative Deadline (D_i)
  - Execution Cost(C_i)
  - Recovery Execution Cost($\bar{C}_i$)

- A fault tolerant real-time system must have an appropriate **scheduling policy** and a suitable **recovery scheme**, which aims at putting the system in a safe state.

- There are **several** fault-tolerant scheduling approaches for real-time systems
- Indeed, they are strictly related to the assumed fault model
- Some literature fault models:
  - Faults are random events (aperiodic tasks) ⇒ Probabilistic and Inference Methods / Stochastic Models (Ex.: Markov Chains)
  - Faults are modeled as sporadic events (sporadic tasks) ⇒ Fault Tolerant scheduling ans analysis
  - Faults are modeled as periodic events (periodic tasks) ⇒ Worst-Case assumptions

- Faults are random events (aperiodic tasks) ⇒ Probabilistic and Inference Methods / Stochastic Models (Ex.: Markov Chains)

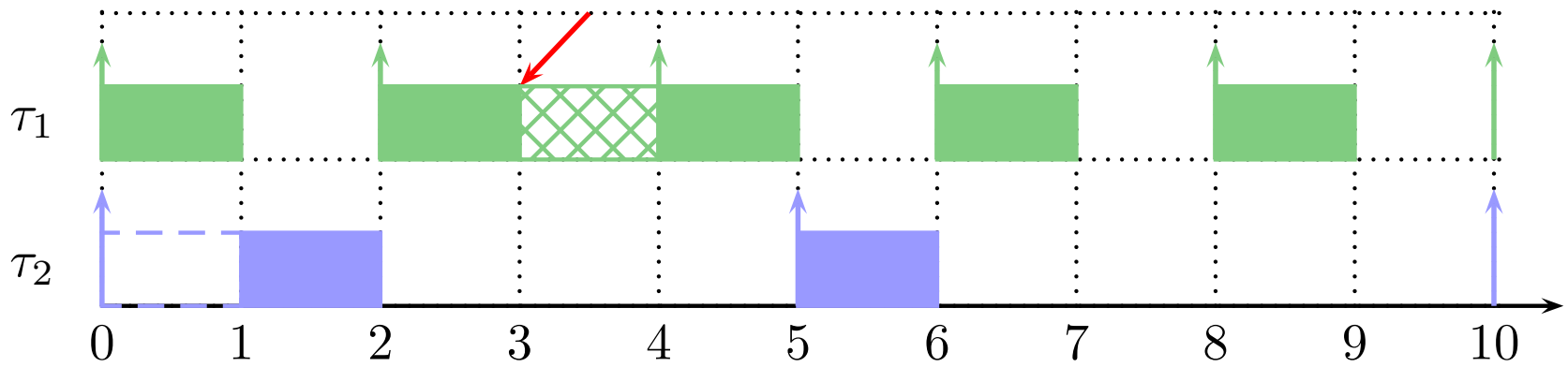
Sporadic Events

- Faults are modeled as sporadic events (sporadic tasks) $\Rightarrow$ Fault Tolerant scheduling ans analysis

- Faults are modeled as periodic events (periodic tasks) $\Rightarrow$ Worst-Case assumptions

- Fault-Tolerance is achieved recovery actions upon errors detection;
- Usually, recovery scheme is based on temporal redundancy, since transient faults are mentioned as the most frequent ones

Recovery based on the reexecution of the faulty task



Recovery based on executing an alternative task version