# Quality-Of-Service: The Role of Energy

Geovani Ricardo Wiedenhoft, Arliones Stevert Hoeller Junior and Antônio Augusto Fröhlich
Laboratory for Software and Hardware Integration – Federal University of Santa Catarina
PO Box 476 – 88049-900 – Florianópolis, SC, Brazil – {grw, arliones, guto}@lisha.ufsc.br

## Introduction

- Embedded systems are computational platforms dedicated to perform a known set of tasks.
- Due to the mobile nature of their applications, are powered by batteries.

- It is not enough to guarantee other QoS metrics if, by doing so, the system runs out of battery and is unable to complete its computations.

- We use the energy as a QoS parameter.

- Our work developed an approach to guarantee that embedded systems' batteries lifetime can last time enough to assure the execution of at least essential tasks.
- The developer define the period that the system must be operational.
- By monitoring batteries lifetime, a scheduler is able to decrease QoS levels in order to reduce energy consumption and enhance system lifetime.

- QoS control of applications was inspired by **imprecise computing**.

## Implementation

- Imprecise computing techniques had to be adapted to switch its scheduling focus from timing constraints to energy constraints.

- The prototype was developed in EPOS - Embedded Parallel Operating System.
- EPOS provides a infrastructure of energy consumption management for embedded systems and it also provides abstractions to access system batteries' information.

- The programmer provides two entry points when creating a thread.
- The programmer specifies in the mandatory subtask the point where the optional subtask should execute by calling the *optional* method.
  - The optional part can be positioned before, after or in the middle of the mandatory task.

- The application programmer also has to define how long the system is expected to last by passing a timestamp to the system through the *system_lifetime* method.

## Imprecise Computing

- Scheduling technique originally proposed to satisfy timing requirements of real-time tasks through decreasing QoS levels.

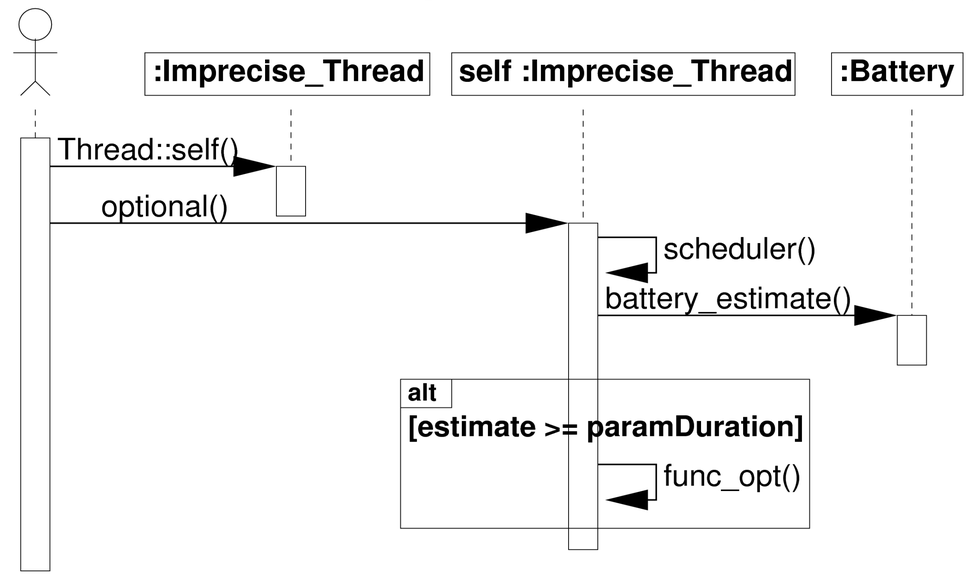
- Tasks are divided into two subtasks.
  - Mandatory execution flow.
    - Core of the task, and must always be avaliable for execution.
    - Generate imprecise results which reflects the minimum of QoS.
  - Optional flow.
    - Performs non-critical actions, and can be prevented from executing.
    - Enhances the imprecise results quality, generating the precise results.

- Worsen quality of results by not executing optional portions.
  - To guarantee that no execution deadline will be lost.

- Unites real-time computing and best-effort techniques.

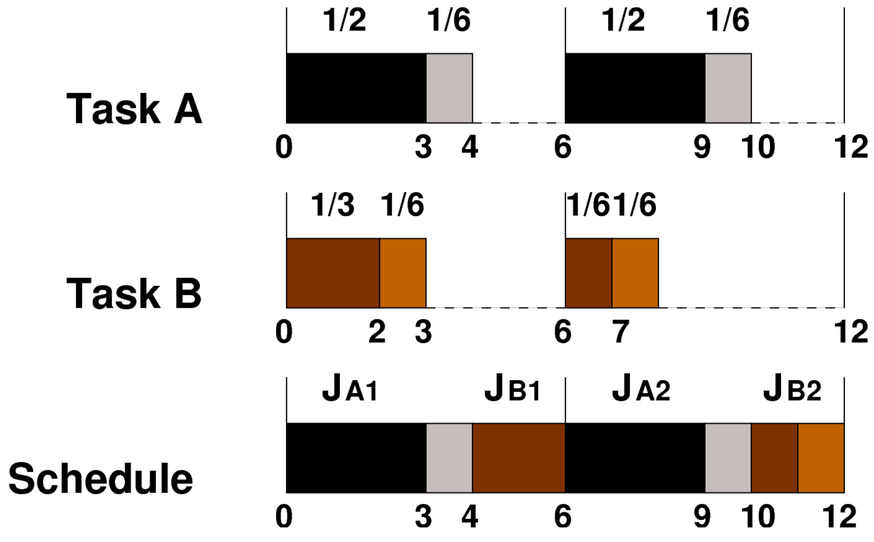
## Overview of an Optional Subtask Execution

- The following figure presents an UML sequence diagram that shows how the system handles the execution of the optional subtask called by mandatory subtask.




## Overview of an Imprecise Computing Scheduling

- The following figure presents an imprecise computing scheduling that shows two imprecise tasks to be scheduled and its schedule.




## Conclusion and Future Work

- Our work proposed an approach to satisfy a defined battery lifetime in mobile embedded systems.

- This approach was inspired by the concept of imprecise tasks.

- The scheduler prevents optional subtasks from executing when the energy budget is bellow a pre-defined limit.
  - This reduces system's processing demands and creates more idle periods.

- The scheduler can use the EPOS infrastructure to stopping or slowing down system components during idle periods.
  - To consume less energy.
  - To enhance battery lifetime.

- Future work will walk towards an approach to guarantee both energy consumption and timing requirements.