Ordonnancement temps réel et autonomie énergétique

Maryline Chetto

IUT de Nantes

IRCCyN, UMR CNRS 6597

OUTLINE

- Classical Real-Time Scheduling
- Introduction to Energy Harvesting Technology
- Scheduling Issue for Real-time Energy Harvesting
- Main results

Classical embedded systems

- **An embedded application**
  - permanently resides in an industrial or consumer device
  - provides some type of control function and/or user interface

- **Examples**

They are employed in automobiles, planes, trains, space vehicles, machine tools, cameras, consumer electronics, office appliances, cell phones, PDAs and other handhelds as well as robots and toys.

In summary, everywhere

Requirements for embedded syst.

Performance : responsive, predictability

Economic : cost, time-to-market

Consequential : safety, reliability, security avoid cascading of failure

Debugging: verification, reliable software..
Replicated hardware

In summary: **Smaller, Cheaper, Better, and Faster**

Embedded often means real-time

- Control and monitor physical processes in a **timely** fashion.
- Perform **reliably for long periods**.
- Perform **without direct human intervention**.
- Operate under more **severe constraints** than “normal” software systems.
- Must operate with a minimum memory footprint and **with minimum of support hardware.**

Embedded often means real-time

- **Identify the stimuli to be processed** and the required responses to these stimuli
- For each stimulus and response, **identify the timing constraints**
- Aggregate the stimulus and response processing into **concurrent processes**
- **Design algorithms to process** each class of stimulus and response. These must meet the given timing requirements
- **Design a scheduling system** which will ensure that processes (tasks) are started in time to meet their deadlines
- Integrate using a **real-time executive** or **Real-Time Operating System (RTOS)**

Real-Time Tasks

- **Periodic tasks**
  - Time-driven. Characteristics are known *a priori*
  - Task T_i is characterized by (p_i, c_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)

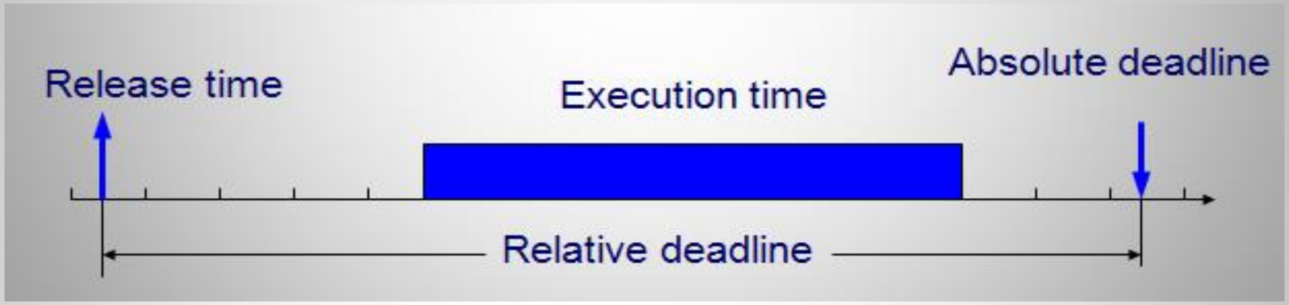
    E.g.: Task activated upon detecting change in patient's condition.

- **Sporadic Tasks**
  - Aperiodic tasks with known minimum inter-arrival time.

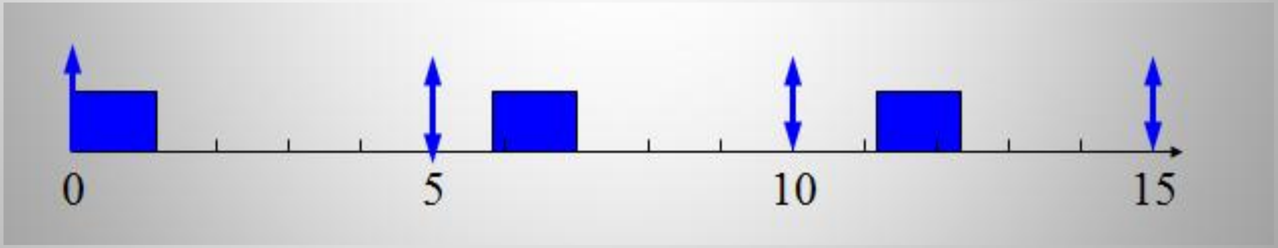
p_i : task period a_i : arrival time r_i : ready time
d_i : deadline c_i : worst case execution time.

Real-Time Tasks

Real-time job

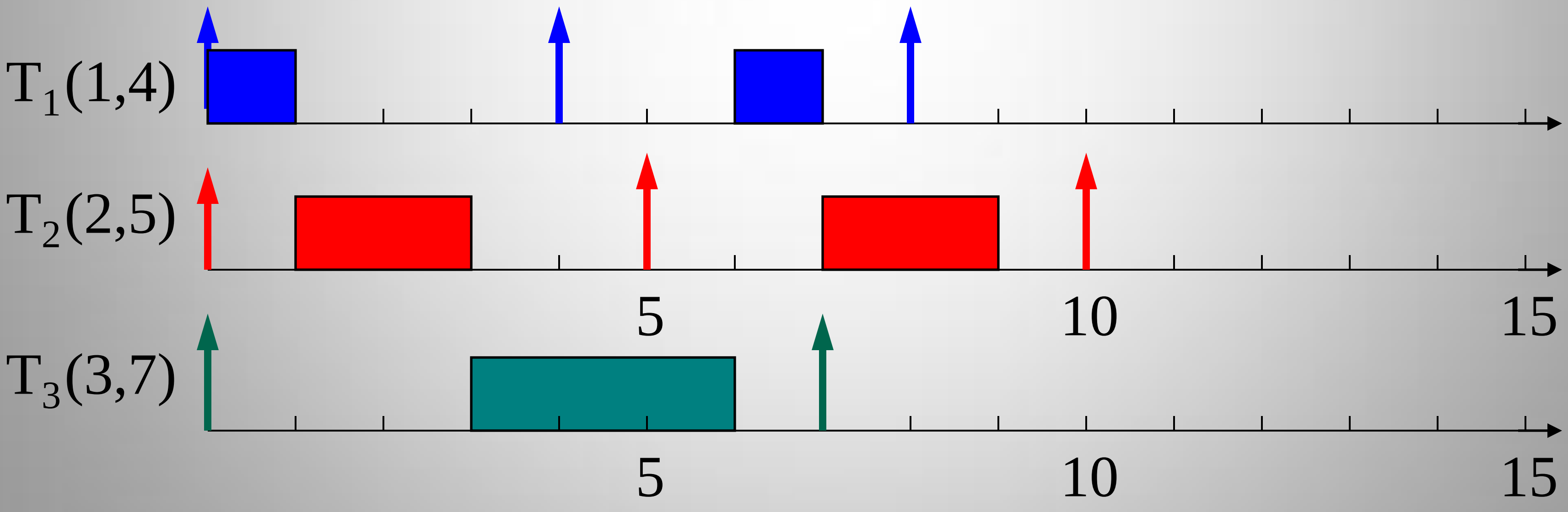


Real-time periodic task



EDF (Earliest Deadline First)

- Executes a job with the earliest deadline
  - Optimal and Full processor utilization
  - Non-clairvoyant and non-idling algorithm
  - BUT misbehavior during overload conditions



Power consumption

During about 30 years, the goal of scheduling has been
to meet deadlines requirements
taking into account **demands of processing time** of all the tasks

- **Given the limited availability of the processor** (limited speed and limited number)
- **Energy is assumed to be non limited**

From 20 years , the goal of scheduling has been

- **to meet deadlines requirements**
- **to minimize the total energy consumption** (battery operated systems)

Power consumption

- Digital devices use CMOS (*Complementary Metal Oxide Semiconductor*) circuits.

$$P_{CMOS} = P_{static} + P_{dynamic} = \sum_{k=1}^{M} C_k \cdot f_k \cdot V_{DD}^2 \propto f \cdot V_{DD}^2$$

- Reducing V_{DD} causes increase of circuit delay
  - CPUs can operate at a lower V_{DD}, but only if f is reduced to tolerate the increased propagation delay.

- Modern CPUs can change V_{DD} and f at runtime
- **DVFS and DPM techniques**

- Up to 2000, only Battery operated devices
- Scheduling issue= optimization of autonomy duration

Energy harvesting

Energy harvesting is the capture of ambient energy to provide electricity for small and or mobile equipment

- can derive energy from a variety of sources including solar, vibration, and temperature variation

Example: At an average power consumption of 100 mW, with 1 cm3 of lithium battery volume , operation during 1 year→ not always acceptable

Energy harvesting provides an average of 100 mW/cm3 **indefinitely**

Objectives:

- long life equipment
- Wired power has limitations in out-door applications
- reducing the need for batteries
- Maintenance-free

Environmental Sources

A wide variety of identifiable sources for Energy Harvesting.

- Natural Energy: wind, water flow
- Thermal Energy
- Light Energy: Solar or room light
- Mechanical Energy: Vibration, mechanical stress
- Bio Energy: mechanical and thermal energy generated by everyday human actions …

Weakness of current systems

Most embedded systems constructed to date do not extract power efficiently from the source.
Consequently, this approach has disadvantages, such as high costs and large space.

They use a much larger harvester (e.g. solar panel) than necessary to yield the same level of power as a more efficient one.

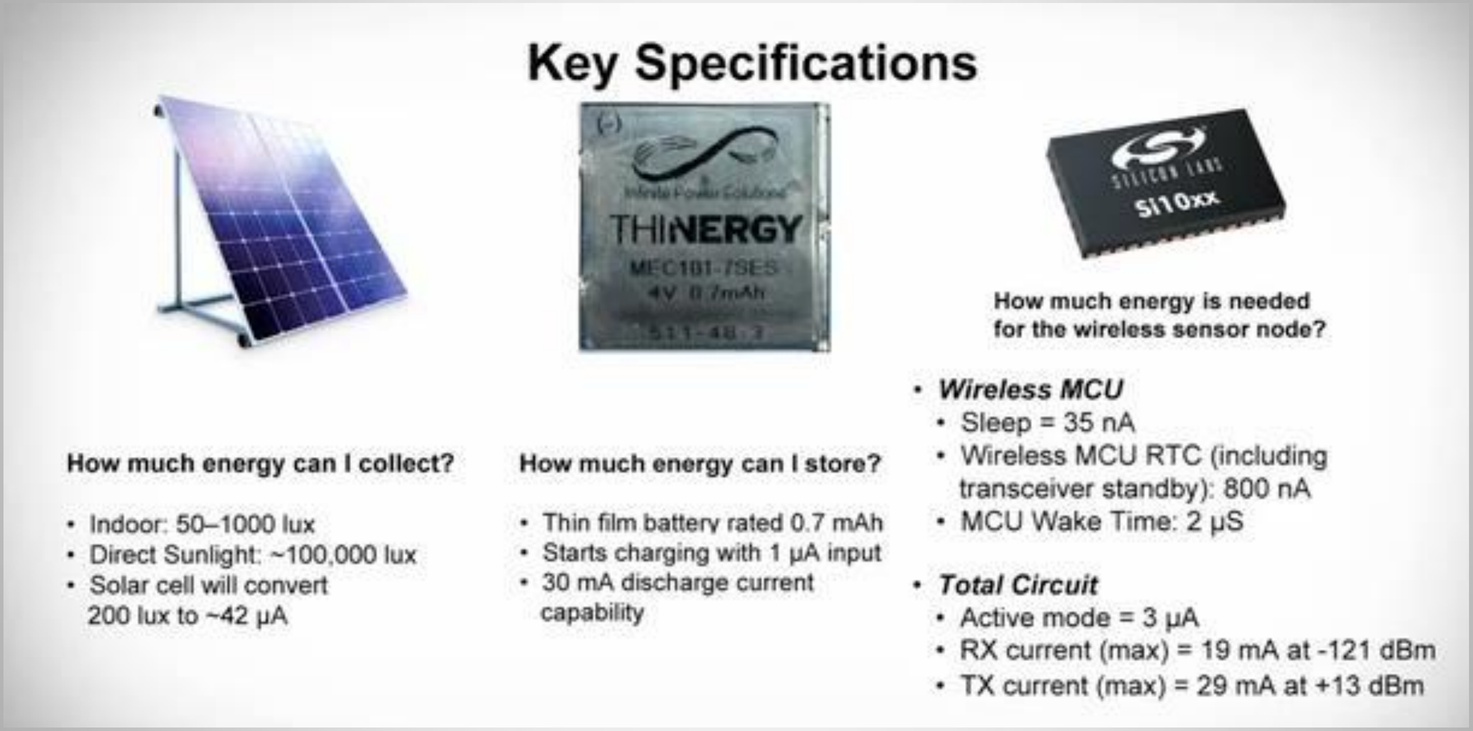
They rely on a larger, more expensive, higher capacity battery than needed in order to sustain extended operation.

Issues:

•Need for **a methodology** for designing a Real-Time Energy Harvesting System in order to determine

- ❑The best suitable energy storage unit
- ❑The best suitable harvester

Key specifications for energy harvesting


THINERGY
MEC101-7SES
4V 0.7mAh
SILICON LABS
Si10xx

Key Specifications

How much energy can I collect?

- Indoor: 50–1000 lux
- Direct Sunlight: ~100,000 lux
- Solar cell will convert 200 lux to ~42 µA

How much energy can I store?

- Thin film battery rated 0.7 mAh
- Starts charging with 1 µA input
- 30 mA discharge current capability

How much energy is needed for the wireless sensor node?

- ***Wireless MCU***
  - Sleep = 35 nA
  - Wireless MCU RTC (including transceiver standby): 800 nA
  - MCU Wake Time: 2 µS
- ***Total Circuit***
  - Active mode = 3 µA
  - RX current (max) = 19 mA at -121 dBm
  - TX current (max) = 29 mA at +13 dBm

Source: Silicon Labs

Design objectives

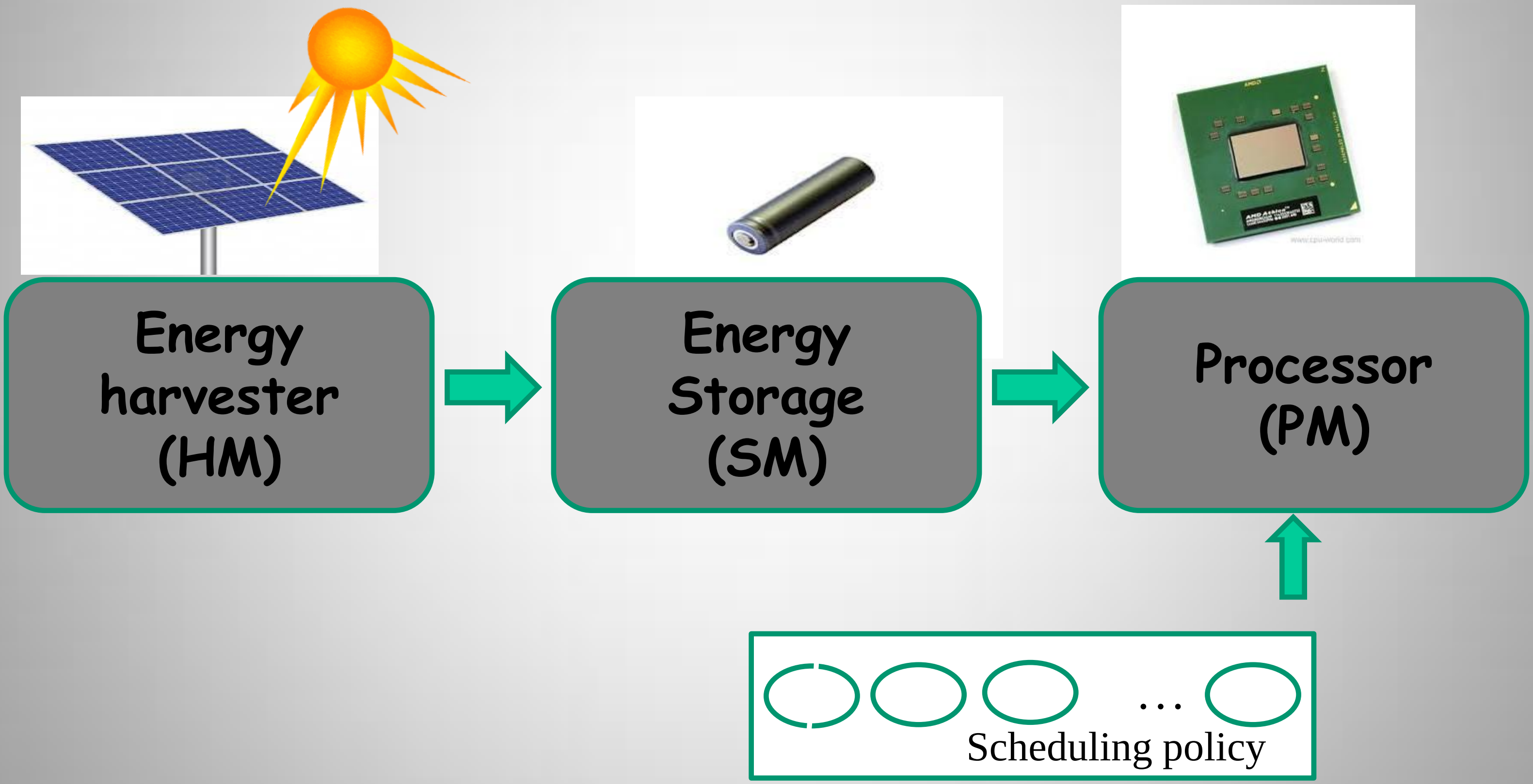
To execute, each task requires only processor time under the previous assumptions.

Our goal will be **to meet deadlines requirements** taking into account

- **demands of processing time**
- **demands of energy**

- **the energy availability**
  - **will be limited**
  - **will be fluctuant**
  - **will be not controllable**
  - **will be unpredictable**

Framework


Energy harvester (HM)
Energy Storage (SM)
Processor (PM)
…
Scheduling policy

Central Issues

- Classical task scheduling, including EDF and RM only accounts for *timing* parameters of the tasks and consequently is not suitable when considering *energy* constraints.

- **The problems we have to deal with are:**

  - **How to extend classical schedulers so as to suitably exploit both the processor and the available ambient energy ?**

  - **What is the feasibility test ?**

  - **How to choose the size of the energy storage?**

Two kinds of energy harvesting systems

- **Harvest-Use:** Energy is harvested just-in-time for use

Example:
The switches draw power from the mechanical energy that's generated when people press the light switch. Wireless signals are transmitted over an RF frequency

- **Harvest-Store-Use:** Energy is harvested whenever possible and stored for future use.

Energy storage is useful when the harvested energy available is more than the current usage

Example:
During the daytime, energy is used for work and also stored for later use. During night, the stored energy is conservatively used to power the sensor node.

Design objectives

1) to operate in an *energy neutral* mode, consuming only as much energy as harvested.
Such a mode of operation raises the possibility of indefinitely long lifetime, limited only by the hardware longevity.

Question: How to operate such that the energy used is always less than the energy harvested?

2) to deal with the real-time tasks (with timing constraints) under the strong variation of energy source with respect to time

Question: What is the maximum performance level that can be supported (in terms of deadline success) ?

Our research

Power management considerations are very different from those of maximizing lifetime i.e. **Power-aware ≠ low-power**

To execute, each task requires time and energy.

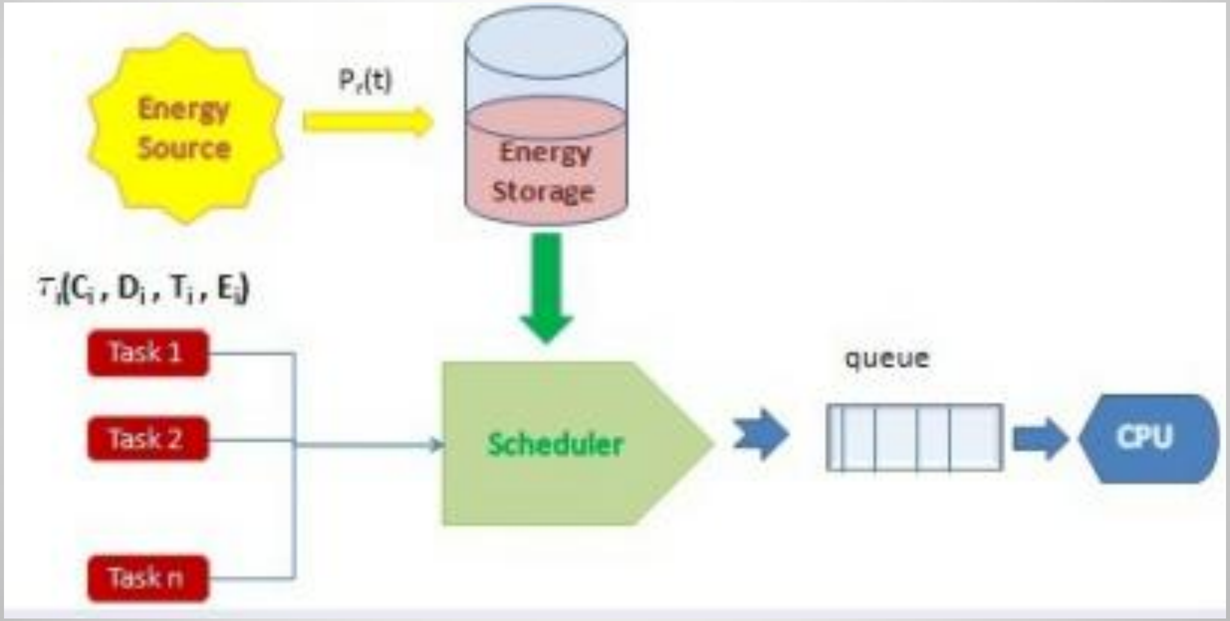
Our goal is **to meet deadlines requirements** taking into account:

1. **demands of energy and processing time** of all the tasks

2. **Limited availability of energy and processing time** due to
   - the energy reservoir (bounded capacity)
   - the energy source (variable and limited charging power)
   - the processor (limited speed)

Our primary goal is not to minimize energy consumption.

A generic energy harvesting system



Assumptions

Energy Storage

A nominal capacity :At any time, the stored energy is no more than the storage capacity, that is $E(t) < E$

$$E = E_{\max} - E_{\min}$$

- Recharching and discharging may overlap

Energy Source

- Energy produced with fluctuating power $P_r(t)$
- We define the WCCR (Worst Case Charging Rate), namely $P_r(t)$, which is a lower bound on the harvested source power output.

The energy drawn from the environment is **uncontrolled but predictable**

(**for example,** the intensity of direct sunlight cannot be controlled but it is predictable with daily and seasonal patterns and we can use prediction algorithms)

Assumptions

Task Set: each task is characterized by :

- **Worst Case Execution Time** (WCET).
- **Worst Case Energy Consumption** (WCEC)
- the **deadline** and **period** respectively.

Preemption is allowed and tasks execute with fluctuating consumption power

.

WCEC is <u>not necessary proportional </u>to WCET

Objective: To satisfy timing constraints on software execution

Summary of initial related works

DVS Approach **(University of Pittsburg, USA)**

- A. Allavena and D. Mossé, Scheduling of Frame based Embedded Systems with Rechargeable Batteries, Workshop on Power Management for Real-Time and Embedded Systems **2001**.

- C. Rusu, R. Melhem and D. Mossé, Multi-version Scheduling in Rechargeable Energy aware Real-time Systems, **ECRTS 2003**

Non DVS Approach

- A. Allavena and D. Mossé, Scheduling of Frame based Embedded Systems with Rechargeable Batteries, Workshop on Power Management for Real-Time and Embedded Systems **2001**.

(Swiss Federal Institute of Technology, ETH Zurich)

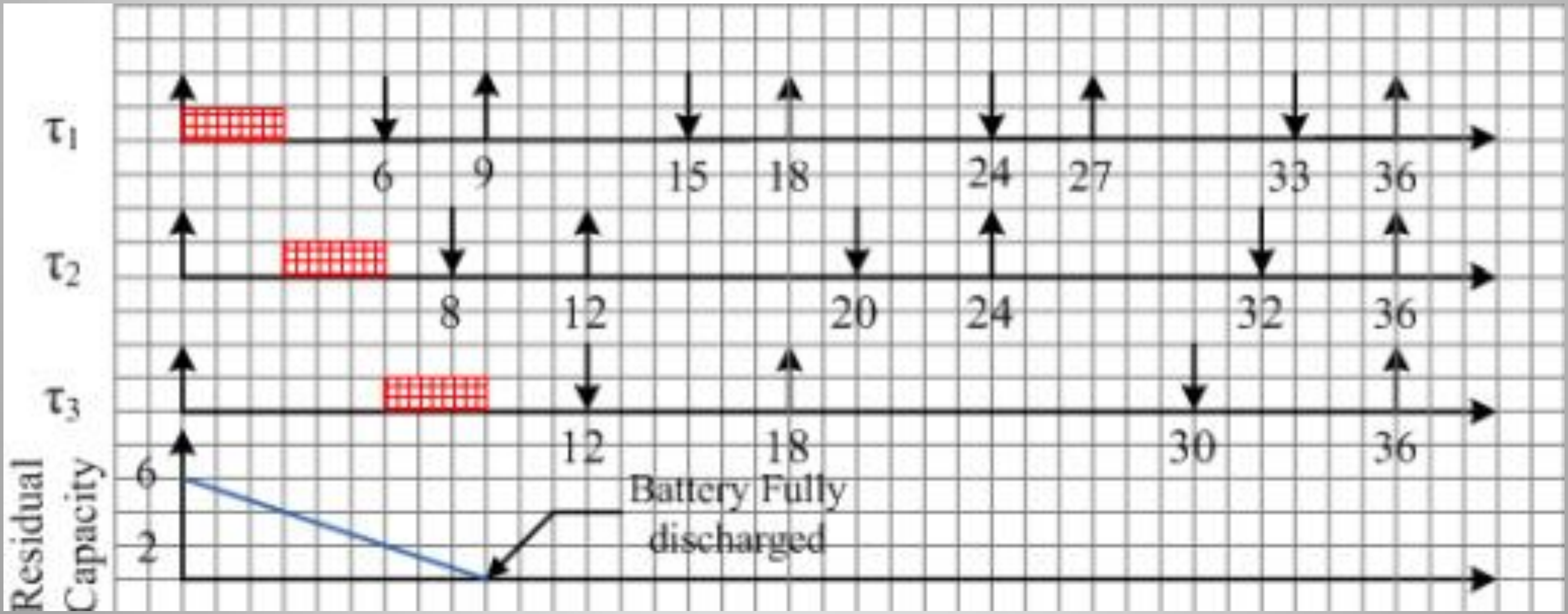
- C. Moser, D. Brunelli and L. Benini Real-time Scheduling with Regenerative Energy. **ECRTS 2006**

Is EDF suitable for RTEH systems?

EDF scheduling is

NON CLAIRVOYANT (cannot anticipate future)

NON idling (does not let the processor inactive if there is at least one job to execute) → it consumes the energy greedily



Is EDF suitable for RTEH systems?

Main results:

1) There cannot exist an optimal online non clairvoyant scheduler **(2013)**

2) All online non clairvoyant schedulers are non competitive **(2013)**

3) An optimal scheduler is at least lookahead-D with D the greatest relative deadline

4) EDF scheduling is the Best Non idling scheduler (2013)

EDF is easy to implement
does not need prediction on energy production
does not need to know energy level at run time

But EDF behaves poorly

ED-H : an Optimal scheduler

- ED-H: online Idling Clairvoyant scheduler
- Its implementation requires:
  - Energy level at run time
  - Energy prediction at run time
  - Task arrival pattern at run time

Main results

1) ED-H is optimal
2) Prediction horizon: maximum relative deadline

CHETTO M., « Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems » IEEE Transactions on Emerging Topics in Computing, online since january 2014,

Rules of ED-H Scheduler

R1: The EDF order is used to select the future running job in $Lr(t)$.

R2: The processor is imperatively idle in $[t, t+1)$ if $Lr(t) = \emptyset$.

R3: The processor is imperatively idle in $[t, t+1)$ if $Lr(t) \neq \emptyset$ and either $E(t) = 0$ or $Slack.energy(t) = 0$.

R4: The processor is imperatively busy in $[t, t+1)$ if $Lr(t)) \neq \emptyset$ and either $E(t) = C$ or $Slack.time(t) = 0$

R5: The processor can equally be idle or busy if $Lr(t)) \neq \emptyset$, $0 < E(t) < C$, $Slack.time(t) > 0$ and $Slack.energy(t) > 0$.

Properties of ED-H Scheduler

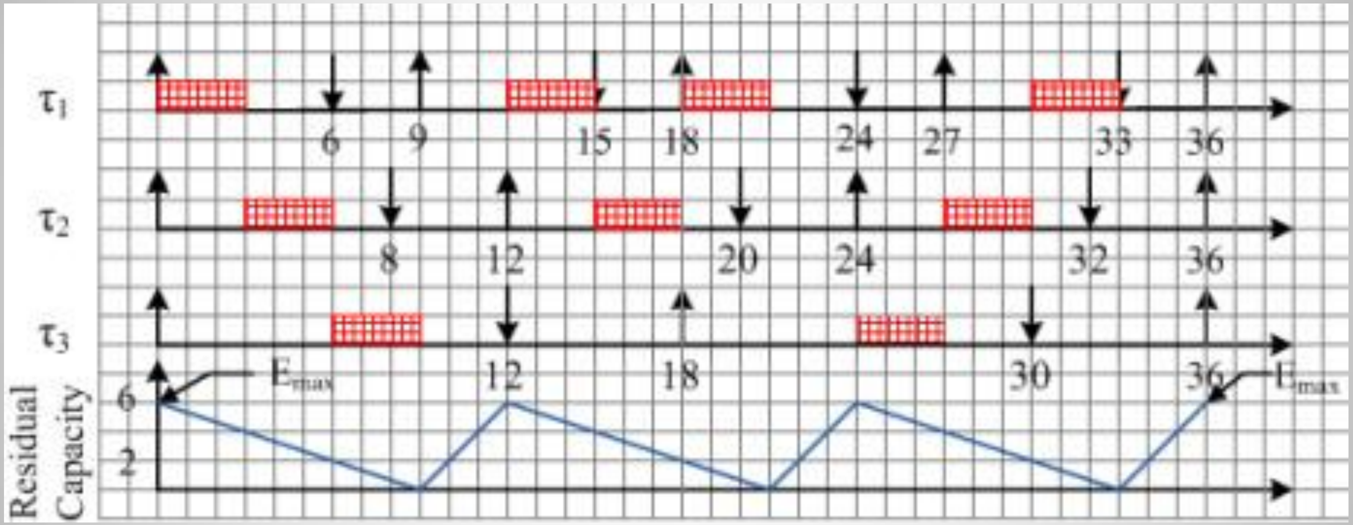
- **Two central rules:**

1. The highest priority task can be executed as long as there is **slack energy**.
2. The processor can be idle (for recharging energy) as long as there is **slack time**

- **Slack time (t) =0** means that the processor has to be busy from t so as to guarantee all future deadlines.

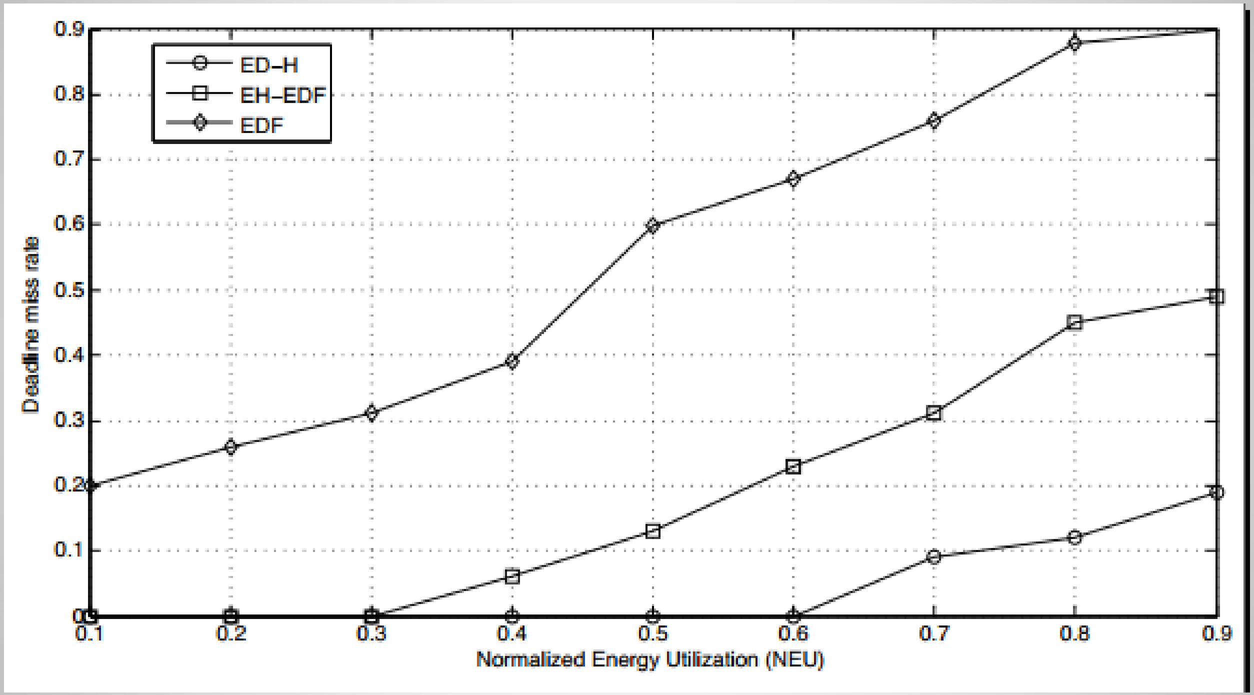
- **Slack energy (t) =0** means that the processor has to be idle from t so as to avoid energy starvation for future tasks.
- We only waste recharging power when there are no pending tasks and the storage unit is full.

Illustrative example



ED-H scheduling strategy

Some simulation results



RTOS Requirements

Adding Energy harvesting aware features to a RTOS requires capabilities for :

- measurement of the current available energy in the storage
- estimation of the future harvested energy
- estimation of the energy consumption of every task.

Here are the main technological challenges !!!

Merci de votre attention

Contact: maryline.chetto@univ-nantes.fr