Schedulability Analysis for Fixed Priority Real-Time Systems with Energy- Harvesting

Yasmina Abdeddaïm[1] Younes Chandarli[1,2] Rob Davis [3] Damien Masson [1]

[(1)] *Université Paris-Est, LIGM UMR CNRS 8049, ESIEE Paris, France*

[(2)] *Université Paris-Est, LIGM UMR CNRS 8049, Université Paris-Est Marne-La-Vallée, France*

[(3)] *Real-Time Systems Research Group, University of York*

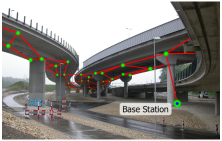
RTNS'14, 10 October 2014

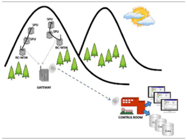
Outline

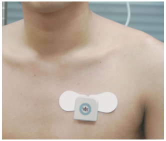
Energy Harvesting Systems

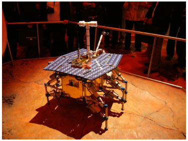
Energy-Harvesting

The process by which energy is captured from a system's environment and converted into usable electric power.





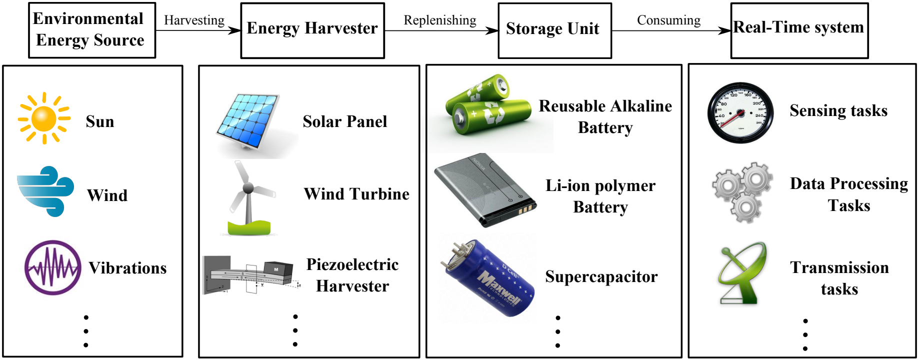




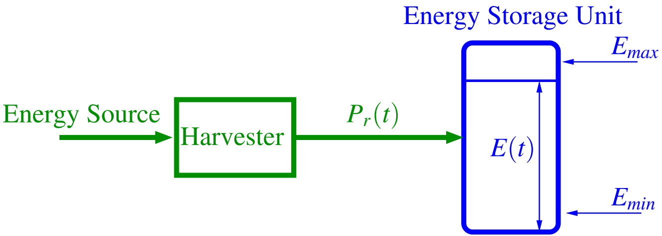
Energy Harvesting Systems

Energy-Harvesting

The process by which energy is captured from a system's environment and converted into usable electric power.



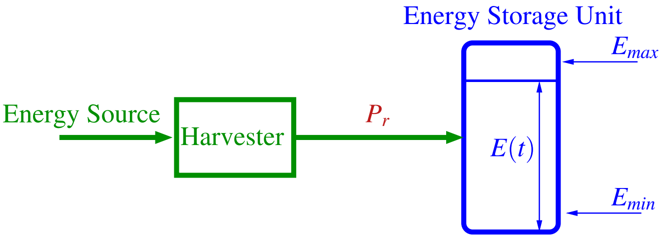
Energy Model



- Energy Source Model
  - Energy Sources: solar, thermal, mechanical, vibration, . . .
  - Harvester: transform the environmental energy into electrical power.
- Energy Storage Unit Model
  - Energy Unit: battery, super-capacitor, . . .
  - Store the harvested energy: $P_r(t)$ is the energy replenishment function.

    Constant rate of replenishment: $P_r(t) = P_r$
  - The energy stored may vary between two levels E_{min} and E_{max}.

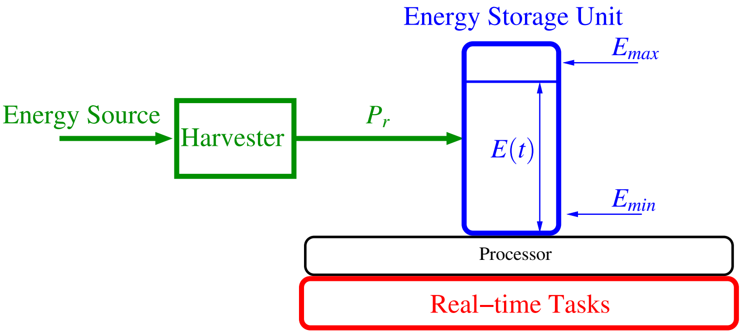
Energy Model



- Energy Source Model
  - Energy Sources: solar, thermal, mechanical, vibration, ...
  - Harvester: transform the environmental energy into electrical power.
- Energy Storage Unit Model
  - Energy Unit: battery, super-capacitor, ...
  - Store the harvested energy: $P_r(t)$ is the energy replenishment function.

    Constant rate of replenishment: $P_r(t) = P_r$
  - The energy stored may vary between two levels E_{min} and E_{max}.

Task Model



A set of sporadic tasks $\tau_i(C_i, P_i, E_i, T_i, D_i)$

C_i: worst-case execution time,

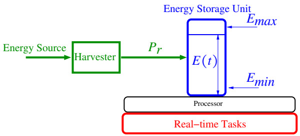
P_i: worst-case power consumption,

$E_i = P_i \times C_i$: worst-case energy consumption,

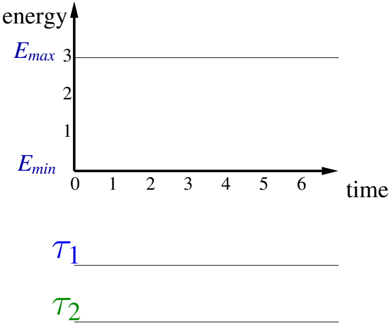
T_i: minimal inter arrival time,

D_i: relative deadline ($D_i \leq T_i$).

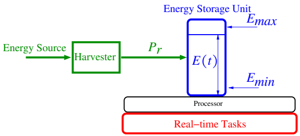
The Model



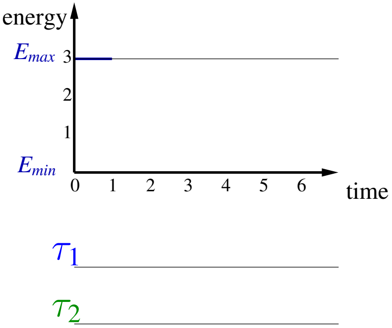
$$
\begin{array}{llll}
P_r = 3 & E_{max} = 3 & E_{min} = 0 & \\
\tau_1 : C_1 = 2 & P_1 = 6 & E_1 = 12 & T_1 = D_1 = 4 \\
\tau_2 : C_2 = 1 & P_2 = 1 & E_2 = 1 & T_2 = D_2 = 5
\end{array}
$$



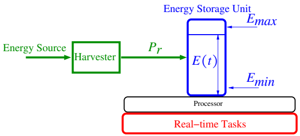
The Model



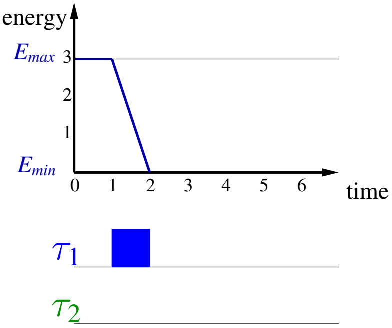
$$
\begin{array}{llll}
P_r = 3 & E_{max} = 3 & E_{min} = 0 & \\
\tau_1 : C_1 = 2 & P_1 = 6 & E_1 = 12 & T_1 = D_1 = 4 \\
\tau_2 : C_2 = 1 & P_2 = 1 & E_2 = 1 & T_2 = D_2 = 5
\end{array}
$$



The Model

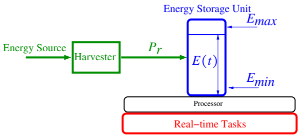


$$
\begin{array}{llll}
P_r = 3 & E_{max} = 3 & E_{min} = 0 & \\
\tau_1 : C_1 = 2 & P_1 = 6 & E_1 = 12 & T_1 = D_1 = 4 \\
\tau_2 : C_2 = 1 & P_2 = 1 & E_2 = 1 & T_2 = D_2 = 5
\end{array}
$$



Consuming Task: $P_1 > P_r$

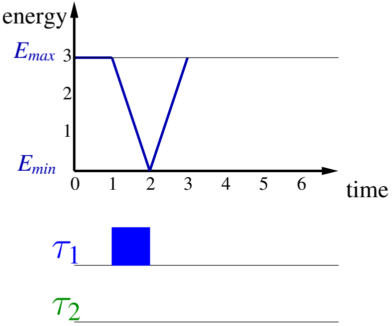
The Model



$P_r = 3$ $\quad E_{max} = 3$ $\quad E_{min} = 0$

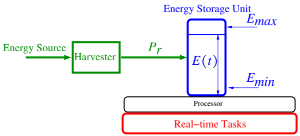
$\tau_1 : C_1 = 2$ $\quad P_1 = 6$ $\quad E_1 = 12$ $\quad T_1 = D_1 = 4$

$\tau_2 : C_2 = 1$ $\quad P_2 = 1$ $\quad E_2 = 1$ $\quad T_2 = D_2 = 5$

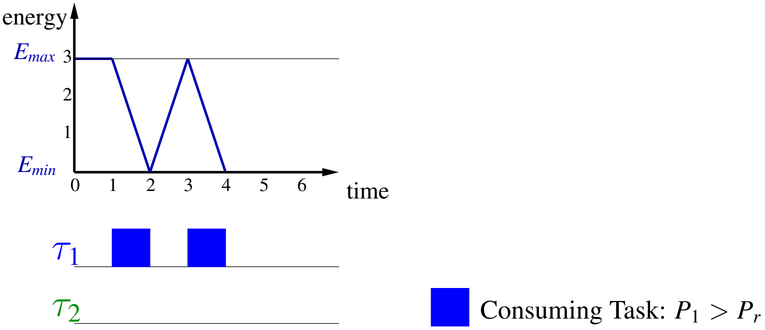


Consuming Task: $P_1 > P_r$

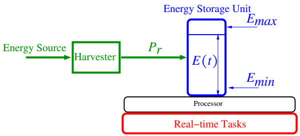
The Model



$$
\begin{array}{llll}
P_r = 3 & E_{max} = 3 & E_{min} = 0 & \\
\tau_1 : C_1 = 2 & P_1 = 6 & E_1 = 12 & T_1 = D_1 = 4 \\
\tau_2 : C_2 = 1 & P_2 = 1 & E_2 = 1 & T_2 = D_2 = 5
\end{array}
$$



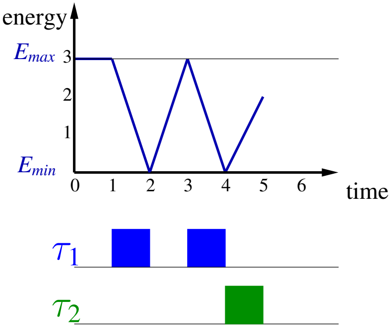
The Model



$P_r = 3$ $\quad E_{max} = 3$ $\quad E_{min} = 0$

$\tau_1 : C_1 = 2$ $\quad P_1 = 6$ $\quad E_1 = 12$ $\quad T_1 = D_1 = 4$

$\tau_2 : C_2 = 1$ $\quad P_2 = 1$ $\quad E_2 = 1$ $\quad T_2 = D_2 = 5$



Consuming Task: $P_1 > P_r$

Gaining Task: $P_2 \leq P_r$

The Scheduling Problem

The Model

- Storage Unit: Constant rate replenishment P_r and E_{max}, E_{min} the maximal and minimal level of energy.
- A set $\Gamma = \Gamma_c \cup \Gamma_g$ of sporadic tasks $\tau_i = (C_i, P_i, E_i, T_i, D_i)$ in priority order with $D_i \leq T_i$:
  - Consuming Tasks: $\Gamma_c = \{\tau_i \in \Gamma, P_i > P_r\}$
  - Gaining Tasks: $\Gamma_g = \{\tau_i \in \Gamma, 0 \leq P_i \leq P_r\}$

Feasibility

A task set is feasible if all the tasks meet their deadlines: *timing constraints* and $\forall t \geq 0$ the energy level is between E_{min} and E_{max}: *energy constraints*.

Related Work

- An algorithm for Frame-Based Model,

  A. Allavena and D. Mossé,
  "Scheduling of Frame-based Embedded Systems with Rechargeable Batteries",
  Workshop in conjunction with RTAS, 2001.

  LSA Algorithm assumes variable execution time,

  C. Moser, D. Brunelli, L. Thiele and L. Benini,
  "Real-time scheduling with regenerative energy",
  ECRTS, 2006.

- $EDeg$ Algorithm based on EDF priority assignment.

  H. EL Ghor, M. Chetto and R. Chehade,
  "A real-time scheduling framework for embedded systems with environmental energy harvesting",
  Computers & Electrical Engineering journal, 2011.

- PFP_{ASAP} Algorithm

Related Work

- An algorithm for Frame-Based Model,

  A. Allavena and D. Mossé,
  "Scheduling of Frame-based Embedded Systems with Rechargeable Batteries",
  Workshop in conjunction with RTAS, 2001.

  LSA Algorithm assumes variable execution time,

  C. Moser, D. Brunelli, L. Thiele and L. Benini,
  "Real-time scheduling with regenerative energy",
  ECRTS, 2006.

- $EDeg$ Algorithm based on EDF priority assignment.

  H. EL Ghor, M. Chetto and R. Chehade,
  "A real-time scheduling framework for embedded systems with environmental energy harvesting",
  Computers & Electrical Engineering journal, 2011.

- PFP_{ASAP} Algorithm

  Y. Abdeddaïm, Y. Chandarli and D. Masson,
  "The Optimality of PFP_{ASAP} Algorithm for Fixed-Priority Energy-Harvesting Real-Time Systems",
  ECRTS, 2013.

The PFP_{ASAP} Algorithm

- Execute tasks whenever there is enough energy available in the battery.
- Replenish as long as needed to execute one time unit of the highest priority active task.

PFP_{ASAP} is an Energy Work-Conserving FPPS Algorithm

The processor is idle only if there is insufficient energy to schedule at least one time unit of the highest priority active task.

Optimality

PFP_{ASAP} is optimal in the class of energy work conserving fixed priority pre-emptive scheduling algorithms in the case where all the task consume energy $(\Gamma = \Gamma_c)$.

Our Goal

Provide a schedulability test for PFP_{ASAP} when the set of tasks is composed of both consuming and gaining tasks.

The PFP_{ASAP} Algorithm

- Execute tasks whenever there is enough energy available in the battery.
- Replenish as long as needed to execute one time unit of the highest priority active task.

PFP_{ASAP} is an Energy Work-Conserving FPPS Algorithm

The processor is idle only if there is insufficient energy to schedule at least one time unit of the highest priority active task.

Optimality

PFP_{ASAP} is optimal in the class of energy work conserving fixed priority pre-emptive scheduling algorithms in the case where all the task consume energy ($\Gamma = \Gamma_c$).

Our Goal

Provide a schedulability test for PFP_{ASAP} when the set of tasks is composed of both consuming and gaining tasks.

Classical Response Time Analysis

Method

1. Find the wost-case scenario: scenario where τ_i is subject to the maximum possible delay,
2. Compute R_i the longest response time of task τ_i: is the response time of τ_i in the worst-case scenario,
3. Exact schedulability test: If $\forall \tau_i, R_i \leq D_i$ the task set is schedulable.

Work-Conserving FPPS with $D_i \leq T_i$

1. Worst-case scenario for task τ_i: Synchronous release of all the tasks,
2. R_i is given by the smallest $t > 0$ that satisfies $t = F(i,t)$ with:
   $F(i,t) = C_i +$ Maximum interference from higher priority tasks in $[0,t)$

$$F(i,t) = \sum_{h \leq i} \left\lceil \frac{t}{T_h} \right\rceil \times C_h$$

Classical Response Time Analysis

Method

1. Find the wost-case scenario: scenario where τ_i is subject to the maximum possible delay,
2. Compute R_i the longest response time of task τ_i: is the response time of τ_i in the worst-case scenario,
3. Exact schedulability test: If $\forall \tau_i, R_i \leq D_i$ the task set is schedulable.

Work-Conserving FPPS with $D_i \leq T_i$

1. Worst-case scenario for task τ_i: Synchronous release of all the tasks,
2. R_i is given by the smallest $t > 0$ that satisfies $t = F(i,t)$ with:
   $F(i,t) \quad = C_i \quad + \quad$ Maximum interference from higher priority tasks in $[0,t)$

$$F(i,t) = \sum_{h \leq i} \left\lceil \frac{t}{T_h} \right\rceil \times C_h$$

Response Time Analysis for PFP_{ASAP}

Response Time of a task τ_i

C_i + Replenishment time + Interference from higher priority tasks.

$P_r = 3$	$E_{max} = 10$	$E_{min} = 0$		
$\tau_1 : C_1 = 2$	$P_1 = 1$	$E_1 = 12$	$T_1 = 8$	$D_1 = 3$
$\tau_2 : C_2 = 3$	$P_2 = 5$	$E_2 = 15$	$T_2 = 10$	$D_2 = 9$

Response Time Analysis for PFP_{ASAP}

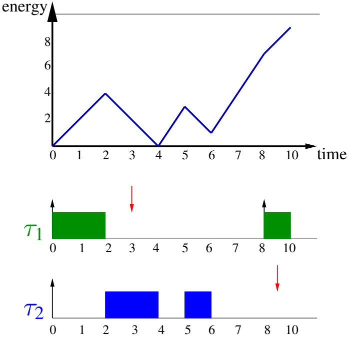
Response Time of a task τ_i

C_i + Replenishment time + Interference from higher priority tasks.

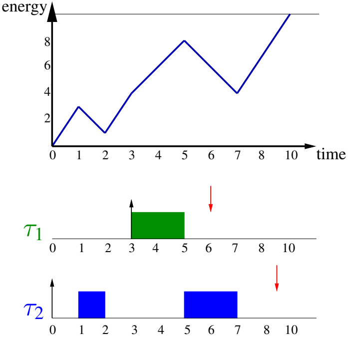
$P_r = 3$ $\quad E_{max} = 10$ $\quad E_{min} = 0$

$\tau_1 : C_1 = 2$ $\quad P_1 = 1$ $\quad E_1 = 12$ $\quad T_1 = 8$ $\quad D_1 = 3$

$\tau_2 : C_2 = 3$ $\quad P_2 = 5$ $\quad E_2 = 15$ $\quad T_2 = 10$ $\quad D_2 = 9$



Response Time $\tau_2 = 6$



Response Time $\tau_2 = 7$

Response Time Analysis for PFP_{ASAP}

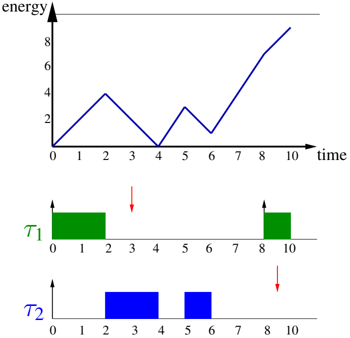
Worst-Case Scenario

The synchronous release of all the tasks is no longer the worst-case scenario.

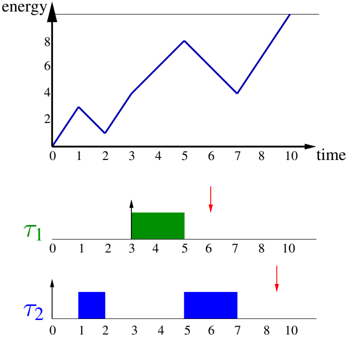
$P_r = 3$ $E_{max} = 10$ $E_{min} = 0$

$\tau_1 : C_1 = 2$ $P_1 = 1$ $E_1 = 12$ $T_1 = 8$ $D_1 = 3$

$\tau_2 : C_2 = 3$ $P_2 = 5$ $E_2 = 15$ $T_2 = 10$ $D_2 = 9$



Response Time $\tau_2 = 6$



Response Time $\tau_2 = 7$

Response Time Analysis for PFP_{ASAP}

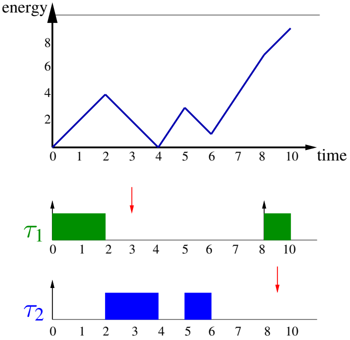
- worst-case scenario is unknown,
- cannot compute exactly R_i, the worst-case response time of task τ_i,
- cannot provide an exact schedulability test.

Upper bound R_i to build a sufficient schedulability test.

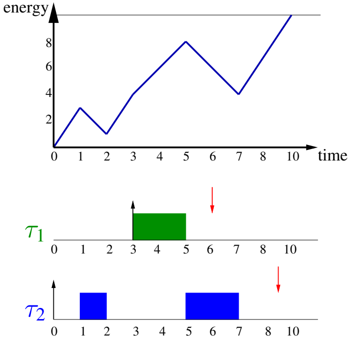
$P_r = 3$ $\quad E_{max} = 10$ $\quad E_{min} = 0$

$\tau_1 : C_1 = 2$ $\quad P_1 = 1$ $\quad E_1 = 12$ $\quad T_1 = 8$ $\quad D_1 = 3$

$\tau_2 : C_2 = 3$ $\quad P_2 = 5$ $\quad E_2 = 15$ $\quad T_2 = 10$ $\quad D_2 = 9$



Response Time $\tau_2 = 6$



Response Time $\tau_2 = 7$

Response Time Analysis for PFP_{ASAP}

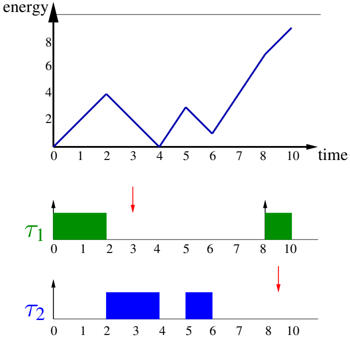
- worst-case scenario is unknown,
- cannot compute exactly R_i, the worst-case response time of task τ_i,
- cannot provide an exact schedulability test.

Upper bound R_i to build a sufficient schedulability test.

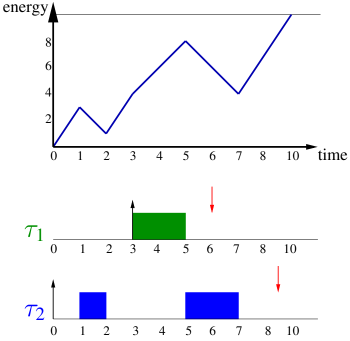
$P_r = 3$ $\quad E_{max} = 10$ $\quad E_{min} = 0$

$\tau_1 : C_1 = 2$ $\quad P_1 = 1$ $\quad E_1 = 12$ $\quad T_1 = 8$ $\quad D_1 = 3$

$\tau_2 : C_2 = 3$ $\quad P_2 = 5$ $\quad E_2 = 15$ $\quad T_2 = 10$ $\quad D_2 = 9$



Response Time $\tau_2 = 6$



Response Time $\tau_2 = 7$

Bounding Response Time

- We require a monotonically non-decreasing function $F(i, w)$ that upper bounds the length of the worst-case response time of task τ_i within an interval of length w,
- The upper bound R_i^{UB} of the worst-case response time R_i corresponds to the smallest $w > 0$ that satisfies $F(i, w) = w$.
- We define for every task τ_i a virtual scenario that:
  1. Maximizes the amount of interference from higher priority tasks,
  2. Maximizes the amount of replenishment time needed.

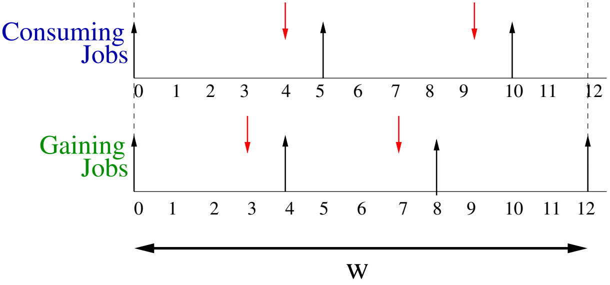
Bounding Response Time

- We require a monotonically non-decreasing function $F(i, w)$ that upper bounds the length of the worst-case response time of task τ_i within an interval of length w,
- The upper bound R_i^{UB} of the worst-case response time R_i corresponds to the smallest $w > 0$ that satisfies $F(i, w) = w$.
- We define for every task τ_i a virtual scenario that:
  1. Maximizes the amount of interference from higher priority tasks,
  2. Maximizes the amount of replenishment time needed.

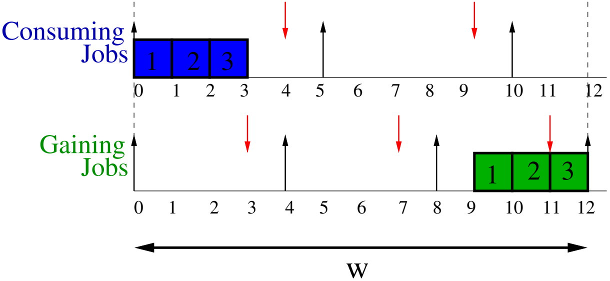
Maximal Interferences

For every task τ_i released in a window of length w, the maximal number of higher priority jobs that are active in this window is

$$\sum_{h<i} \left\lceil \frac{w}{T_h} \right\rceil = \sum_{h<i, \tau_h \in \Gamma_c} \left\lceil \frac{w}{T_h} \right\rceil + \sum_{h<i, \tau_h \in \Gamma_g} \left\lceil \frac{w}{T_h} \right\rceil$$



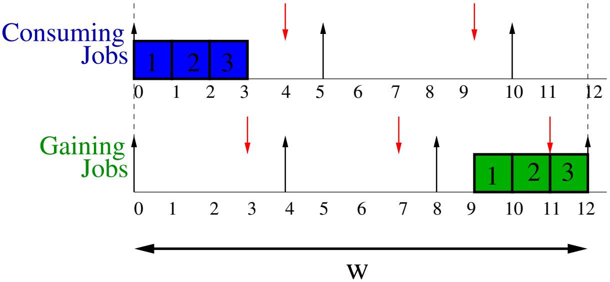
Maximal Replenishment



To upper bound the replenishment in a window of length w we consider a virtual sequence where:

1. The battery is empty at the beginning of the window,
   - to minimize the energy budget of interval w
2. All the consuming jobs are before all the gaining jobs.
   - to maximize replenishment periods

Upper Bound R_i^{UB1}



$$F^{UB1}(i,w) = \underbrace{\left[\sum_{h \le i, \tau_h \in \Gamma_c} \left\lceil \frac{w}{T_h} \right\rceil \times ((P_h - P_r) \times C_h / P_r) \right]}_{\text{maximum replenishment needed}} + \underbrace{\sum_{h \le i, \tau_h \in \Gamma_c} \left\lceil \frac{w}{T_h} \right\rceil \times C_h}_{\text{consuming jobs}}$$

$$+ \underbrace{\sum_{h \le i, \tau_h \in \Gamma_g} \left\lceil \frac{w}{T_h} \right\rceil \times C_h}_{\text{gaining jobs}}$$

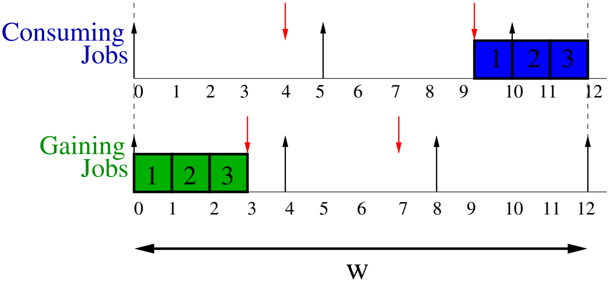
Sufficient Schedulability test $UB1$

- $F^{UB1}(i,w)$ is a monotonically non-decreasing function of w and $F^{UB1}(i,w) > C_i$
- R_i^{UB1} the upper bound of the longest response time of task τ_i is given by the smallest $t > 0$ that satisfies $w = F^{UB1}(i,w)$ with:

$$F^{UB1}(i,w) = \left\lceil \frac{\sum_{h \le i, \tau_h \in \Gamma_c} \left\lceil \frac{w}{T_h} \right\rceil \times E_h}{P_r} \right\rceil + \sum_{h \le i, \tau_h \in \Gamma_g} \left\lceil \frac{w}{T_h} \right\rceil \times C_h$$

- Sufficient Schedulability test $UB1$: $\forall \tau_i, R_i^{UB1} \le D_i$

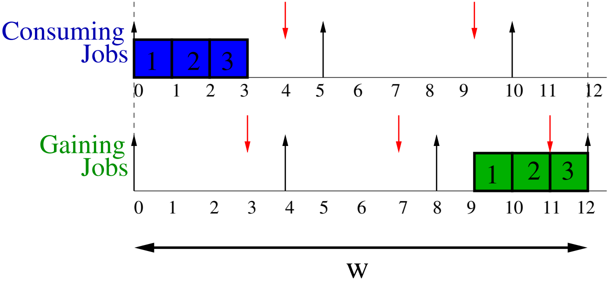
Necessary Schedulability Test $LB1$



To lower bound the worst-case response time of a task τ_i released in a window of length w we consider a virtual sequence where:

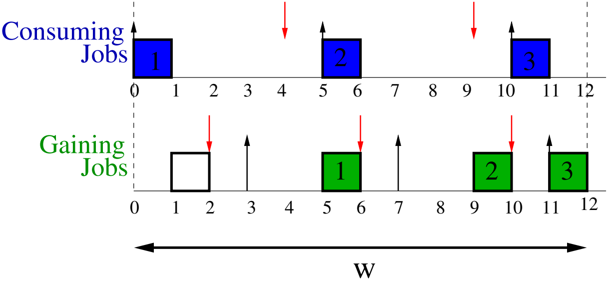
1. The battery is empty at the beginning of the window,
2. All the gaining jobs are before all the consuming jobs.

A Tighter Upper Bound R_i^{UB2}



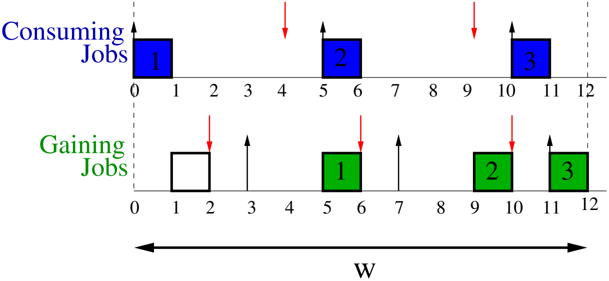
- Consuming jobs as soon as possible,
- Gaining jobs as late as possible

A Tighter Upper Bound R_i^{UB2}



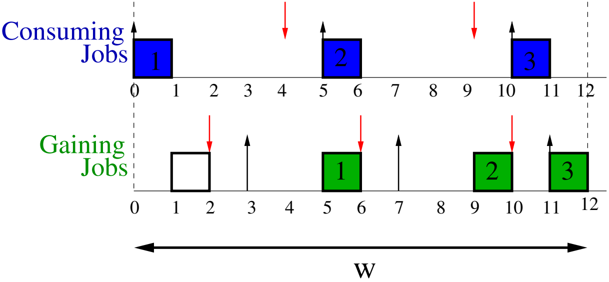
- Consuming jobs as soon as possible,
- Gaining jobs as late as possible

Sufficient Schedulability Test $UB2$



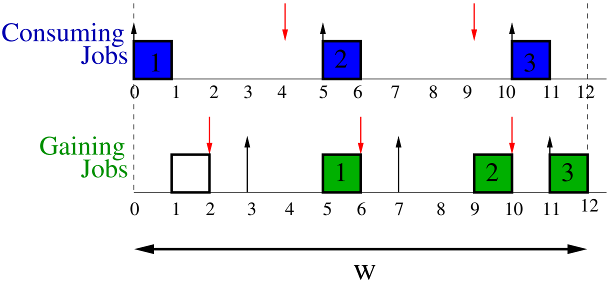
- To compute $F^{UB2}(i,w)$, we compute the response time of the virtual sequence:

Sufficient Schedulability Test $UB2$

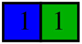


- To compute $F^{UB2}(i, w)$, we compute the response time of the virtual sequence:

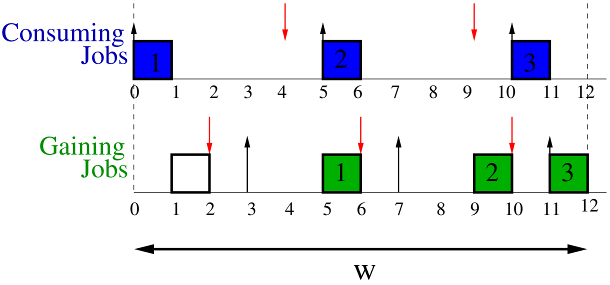
Sufficient Schedulability Test $UB2$



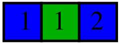
- To compute $F^{UB2}(i,w)$, we compute the response time of the virtual sequence:



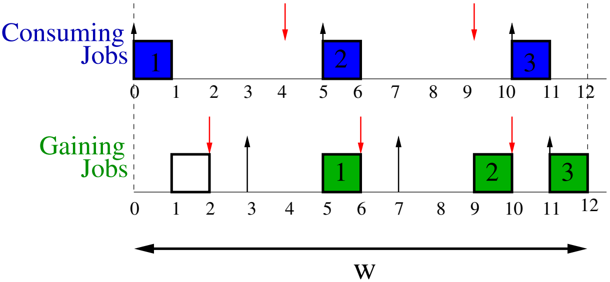
Sufficient Schedulability Test $UB2$



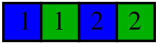
- To compute $F^{UB2}(i,w)$, we compute the response time of the virtual sequence:



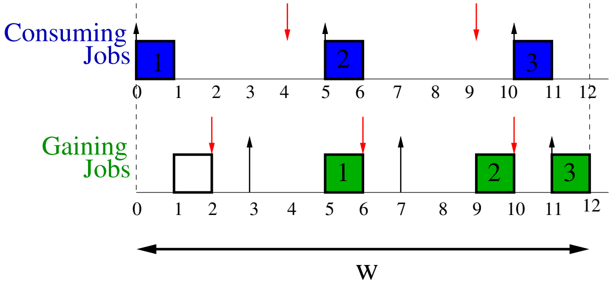
Sufficient Schedulability Test $UB2$



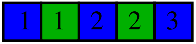
- To compute $F^{UB2}(i, w)$, we compute the response time of the virtual sequence:



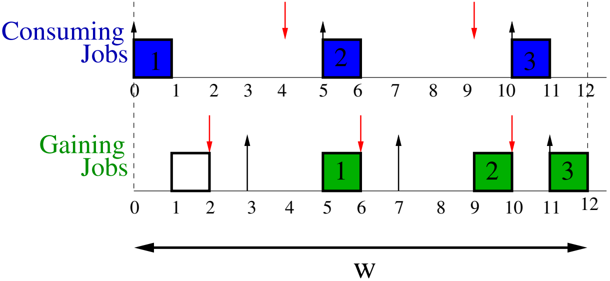
Sufficient Schedulability Test $UB2$



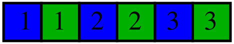
- To compute $F^{UB2}(i, w)$, we compute the response time of the virtual sequence:



Sufficient Schedulability Test $UB2$



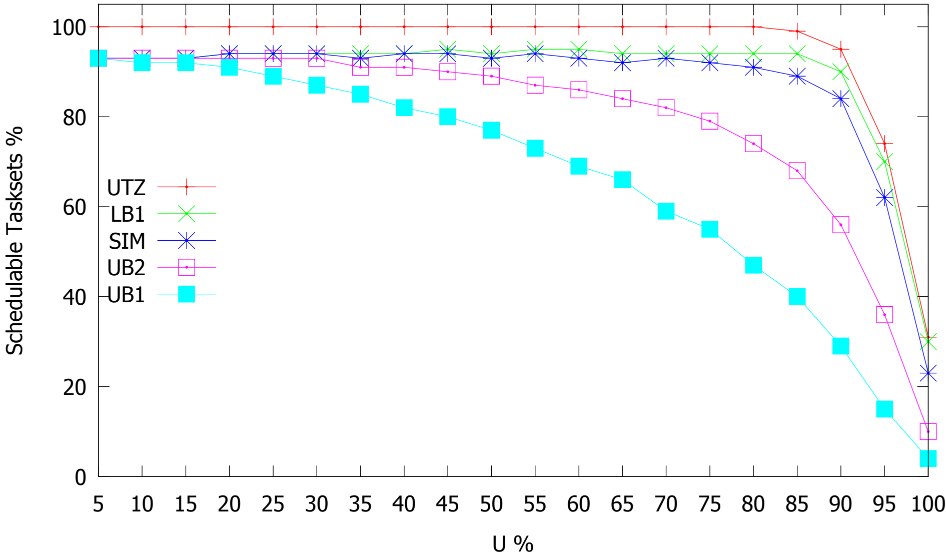
- To compute $F^{UB2}(i, w)$, we compute the response time of the virtual sequence:



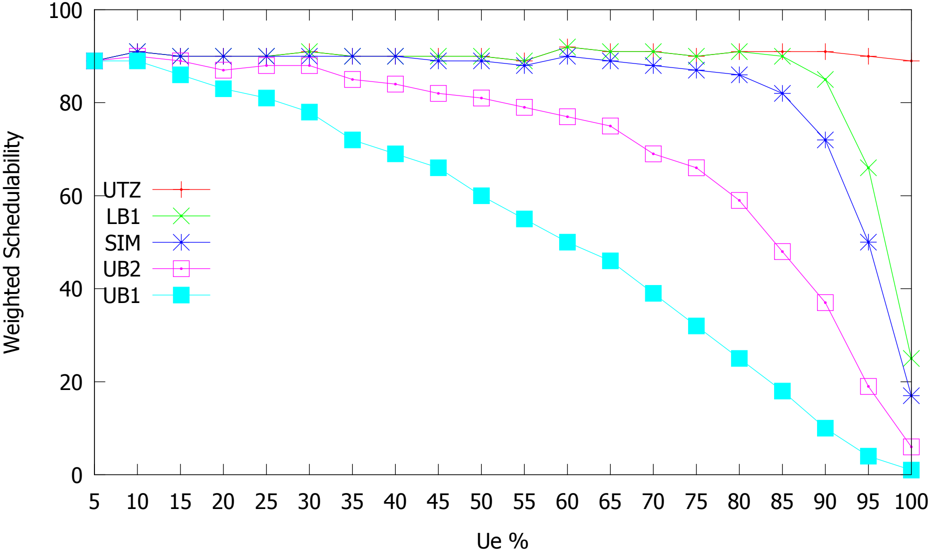
Performance Comparison

- Competitors
  - UTZ: the exact test for $FPPS$ ignoring energy constraints,
  - SIM: an empirical necessary test based on simulating the schedule of PFP_{ASAP} over more than twice the hyper-period,
  - $UB1$: sufficient schedulability test based on the upper bound R^{UB1},
  - $UB2$: sufficient schedulability test based on the upper bound R^{UB2},
  - $LB1$: necessary schedulability test based on the lower bound R^{LB1}.
- Input Data:
  - 40000 task sets randomly generated using UUniFast-Discard algorithm coupled with a technique of hyper-period limitation,
  - Processor utilization varied from 0.05 to 1,
  - Energy utilization varied from 0.05 to 1,
  - Percentage of gaining tasks varied from 0$\%$ to 100$\%$.
- Simulation tool: YARTISS Real-time systems simulator

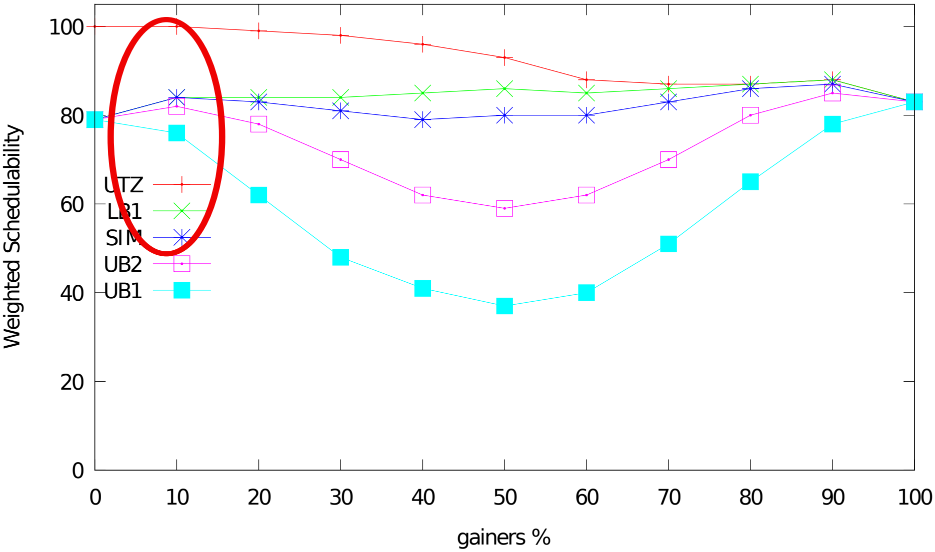
Varying the Processor Utilization



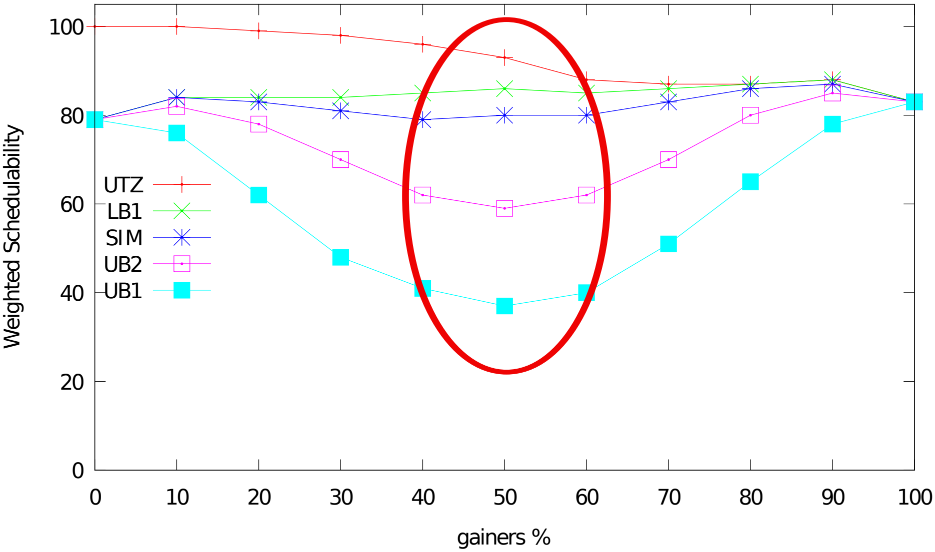
Varying the Energy Utilization



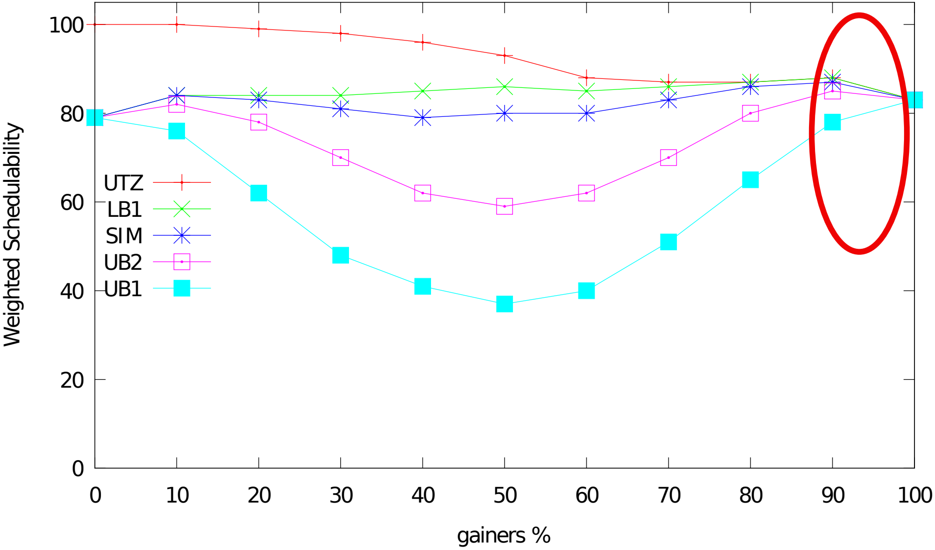
Varying the Gaining Tasks Ratio



Varying the Gaining Tasks Ratio



Varying the Gaining Tasks Ratio



Conclusion and Future Work

1. In this paper we
   - Showed that under PFP_{ASAP}, the worst-case scenario for task sets with both consuming and gaining tasks is not necessarily synchronous release with all other tasks,
   - Derived two sufficient schedulability tests based on two upper bounds on task response times,
   - Derived a necessary schedulability test based on a lower bound on task response time,
   - Evaluated the performance of the sufficient tests in comparison with a number of necessary tests.
2. As future work we plan to:
   - Investigate the problem of optimal priority assignment,
   - Investigate analysis for more complex replenishment functions and additional costs of entering and exiting low power modes needed for energy replenishment.

Conclusion and Future Work

1. In this paper we
   - Showed that under PFP_{ASAP}, the worst-case scenario for task sets with both consuming and gaining tasks is not necessarily synchronous release with all other tasks,
   - Derived two sufficient schedulability tests based on two upper bounds on task response times,
   - Derived a necessary schedulability test based on a lower bound on task response time,
   - Evaluated the performance of the sufficient tests in comparison with a number of necessary tests.
2. As future work we plan to:
   - Investigate the problem of optimal priority assignment,
   - Investigate analysis for more complex replenishment functions and additional costs of entering and exiting low power modes needed for energy replenishment.

Questions ?

Battery Capacity

For the upper bounds to be valid, the battery capacity must be sufficient to store the maximum amount of energy needed in the virtual sequences.

1. For $UB1$: E_{max} must be sufficient to store the energy needed to execute one time unit of the most consuming task:

$$E_{max}^{UB1} \geq \max(\max_{\forall i}(E_i/C_i) - P_r, P_r)$$

2. For $UB2$: E_{max} must be sufficient to store the energy needed to execute consuming jobs in any possible energy busy period:

$$E_{max}^{UB2} \geq \max\left(\sum_{\forall i} \left\lceil \frac{\max_{\forall j}(D_j)}{T_i} \right\rceil \times \max\left(E_i - C_i \times P_r, 0\right), P_r\right)$$