Scheduling Algorithms for Soft Real-Time Systems

Luca Abeni

ReTiS Lab

Scuola Superiore S.Anna, Pisa, Italy

luca@hartik.sssup.it

http://hartik.sssup.it/~luca

Summary

- ◇ Introduction to Real-Time
- ◇ Hard Real-Time
- ◇ Soft Real-Time
  - ⇒ Introduction
  - ⇒ Proportional Share
  - ⇒ Resource Reservation
- ◇ Adaptive Scheduling

Real-Time

- ⋄ Lots of applications present precise timing requirements
  - $\Rightarrow$ control applications: periodic activations
  - $\Rightarrow$ applications that must react to external events
- ⋄ Conventional scheduling $\rightarrow$ minimize response times
- ⋄ Real-Time theory:
  - $\Rightarrow$ each task can be assigned temporal constraints (deadlines)
  - $\Rightarrow$ goal: respect these constraints

Task Model

◇ Real-Time task τ_i: stream of jobs $J_{i,j}$

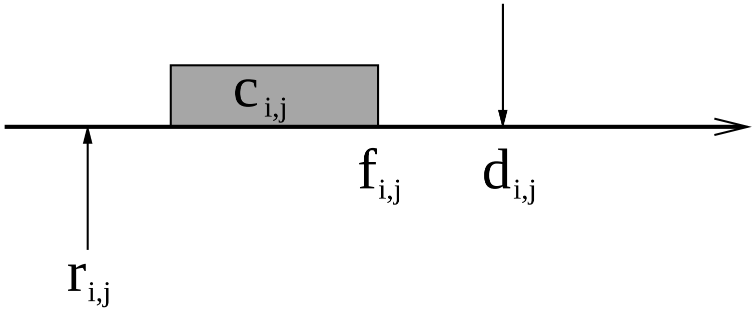
◇ Job $J_{i,j}$:

⇒ arrives at time $r_{i,j}$

⇒ executes for time $c_{i,j}$

⇒ finishes at time $f_{i,j}$

⇒ deadline $d_{i,j}$



Hard Real-Time

◇ Goal: respect **all** the deadlines

⇒ conservative (worst case) assumptions

⇒ a-priori knowledge required

◇ How to obtain it:

⇒ scheduling algorithm

⇒ admission test

Liu & Layland

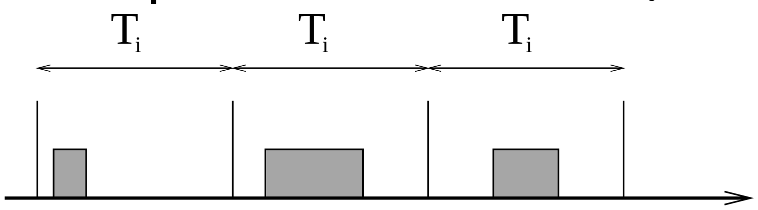
- Periodic tasks

$\Rightarrow r_{i,j+1} = r_{i,j} + T_i$

$\Rightarrow d_{i,j} = r_{i,j} + D_i$

$\Rightarrow D_i = T_i$

- Worst Case assumption: $WCET_i = \max_j \{c_{i,j}\}$



Rate Monotonic - Earliest Deadline First

- Rate Monotonic:

  ⇒ schedule tasks by relative deadlines

- Earliest Deadline First:

  ⇒ schedule tasks by absolute deadlines (dynamic priorities)

- Admission test:

  ⇒ all tasks meet all deadlines if

$$\sum_i \frac{WCET_i}{T_i} \leq U_{lub}$$

Hard Real-Time - Problems

- Underutilization

$$WCET_i >> \overline{c_i}$$

- Misbehaved tasks

$\Rightarrow$ if a task does not respect the declared WCET, the entire system guarantee is compromised...

- no a-priori information (such as WCET) are provided $\rightarrow$ no guarantee

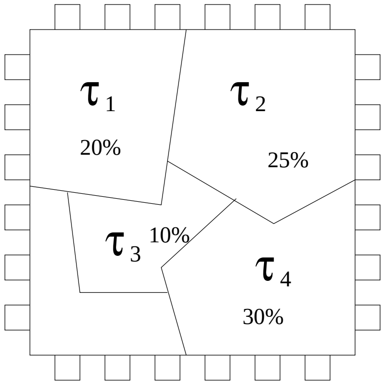
Soft Real-Time

- Each job should finish within its deadline...
  - $\Rightarrow$ ...but a missed deadline is not a disaster!!!
- Soft task:
  - $\Rightarrow$ weak a-priori knowledge
  - $\Rightarrow$ temporal isolation
- How to obtain it:
  - $\Rightarrow$ flexible scheduling
  - $\Rightarrow$ dynamic adaptation

Temporal Isolation

- Temporal isolation (temporal protection)

- each task is assigned a fraction of the available resources (example: the CPU bandwidth)



Temporal Isolation

- **Q:** *What happens if a task requests more resources than it is allowed?*

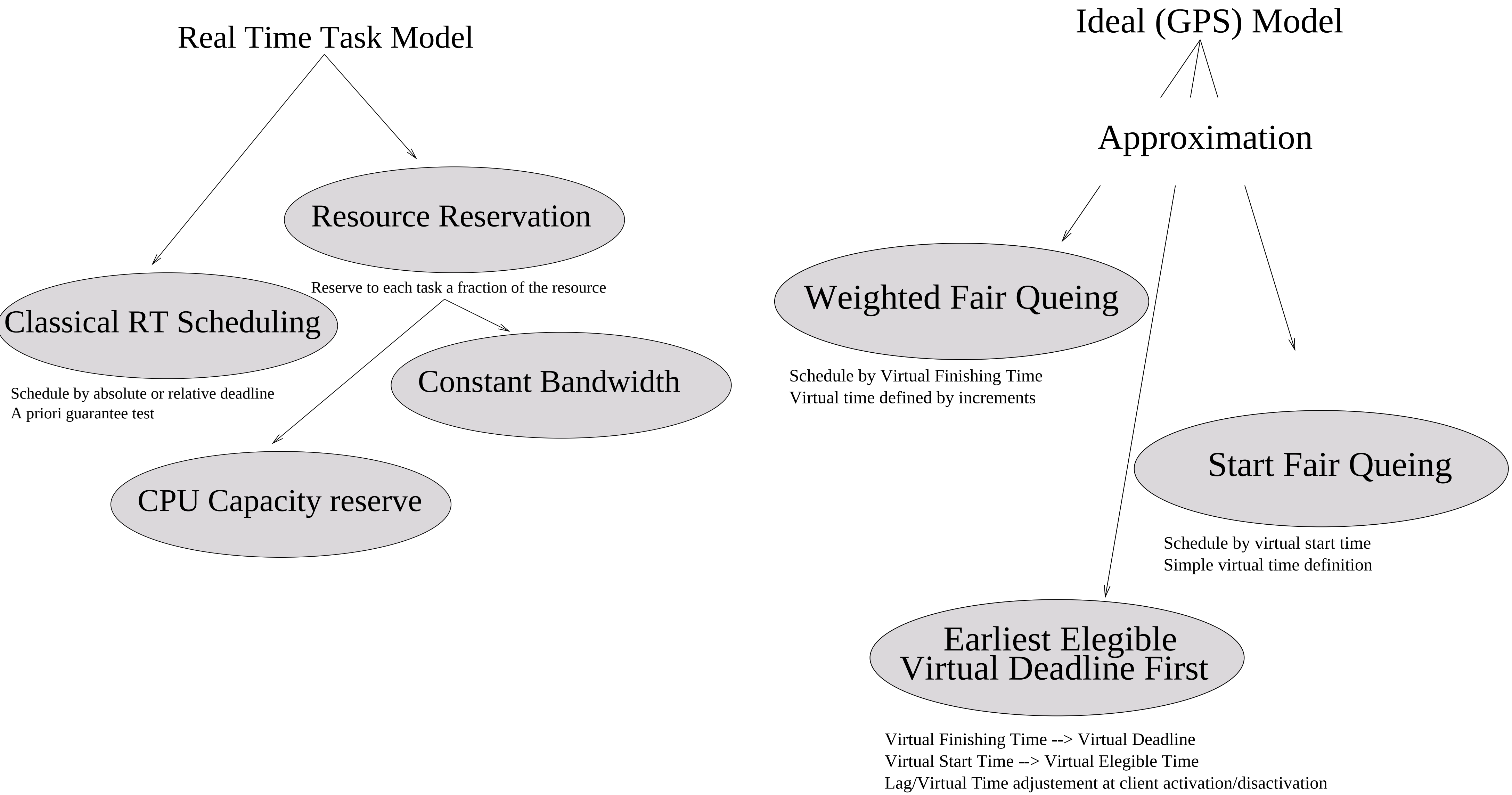
  ⇒ **A:** Only the task itself will suffer.

- **Q:** *What to do in this case?*

  ⇒ **If hard, the worst-case analysis is wrong!**

  ⇒ **If soft the task will slow down in a predictable way.** Nothing catastrophic happens: only QoS degradation.

Luca Abeni - 2000



Proportional Share

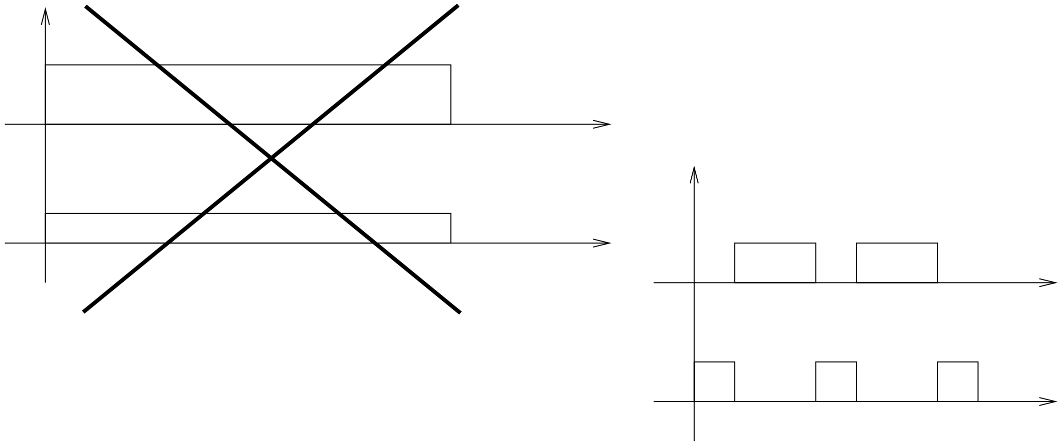
- ideal model: Generalized Processor Sharing (GPS)

- $\forall \tau_i$ and $\forall [t_1, t_2]$, $\frac{e_i(t_1,t_2)}{e_j(t_1,t_2)} \geq \frac{w_i}{w_j}$ $j = 1, 2, ..., n$

- *rate* $F_i = \frac{w_i}{\sum_{\tau_j \in \Gamma} w_j}$ *share* $f_i(t) = \frac{w_i}{\sum_{\tau_j \in \Gamma_a(t)} w_j}$

Proportional Share

- The allocation is quantum-based $\rightarrow$ allocation error



- $\epsilon_{i,j} = \left| \frac{e_i(t_1,t_2)}{w_i} - \frac{e_j(t_1,t_2)}{w_j} \right|$

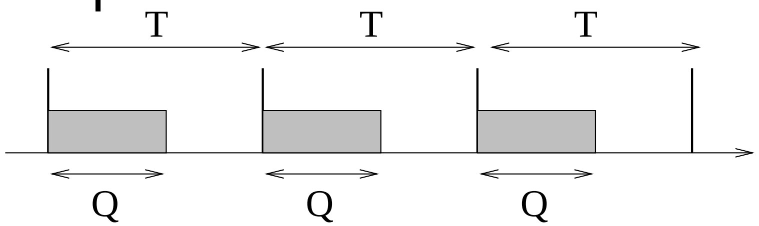
- $lag_i(t_1) = \int_{t_0}^{t_1} f_i(t)dt - e_i(t_0, t_1)$

Resource Reservation

◇ Each task is assigned a *reserved time* in a *period*

$\Rightarrow$ Q: reserved time

$\Rightarrow$ T: reservation period



Bandwidth Reservation

- Each task is reserved a fraction of the total CPU bandwidth

$\Rightarrow$ Demanded Bandwidth:

$$B_i = \max_{t_1, t_2 : t_2 > t_1} \frac{D_i(t_1, t_2)}{t_2 - t_1}$$

$\Rightarrow$ a task will never demand more than its reserved bandwidth

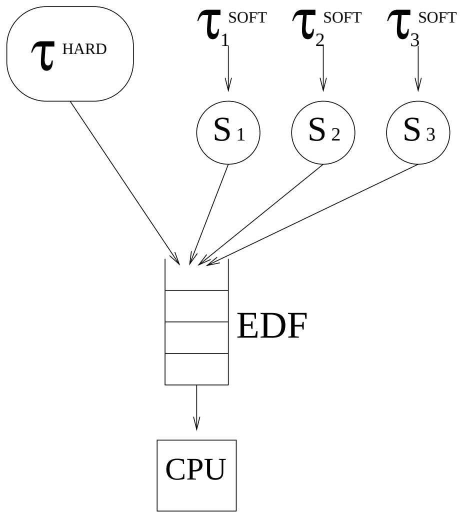
- Enforced by CBS

Bandwidth Reservation

◇ We use EDF+CBS scheduling

⇒ EDF Low-Level Scheduler

⇒ Each task served by a dedicated CBS



The Constant Bandwidth Server

◇ Basic Ideas:

⇒ *budget* → decreases when the served task executes

⇒ *server deadline* → assigned to served task

⇒ task arrival → check if the last deadline can be used

⇒ budget exhausted → deadline postponed

◇ Server parameters:

⇒ Q_s: maximum server budget

⇒ T_s: server period (soft relative deadline)

CBS Properties

- Reserved Bandwidth: $B_s = \frac{Q_s}{T_s}$
- If

$$\sum_i B_i \leq 1$$

$\Rightarrow$ isolation: each task τ_i will never demand more than B_i

$\Rightarrow$ each task τ_i is guaranteed to receive a bandwidth B_i

$\Rightarrow$ a task τ_i requiring more than B_i is slowed down

CBS Properties

- Individual task guarantee
- Equivalence with EDF

  $\Rightarrow$ if $Q_s > WCET$ and $T_s < MIT$, hard guarantee!!!
- Equivalence with PS

  $\Rightarrow$ $Q_s = Q$; $T_s = \frac{Q}{B_s}$
- Probabilistic Guarantee

  $\Rightarrow$ a CBS can be modeled by a Markov Chain

Probabilistic Guarantee

- Soft Task $\rightarrow \tau_i = (U_i(c), V_i(t))$

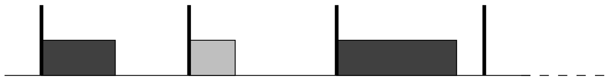
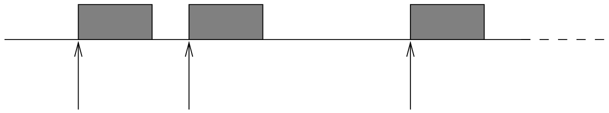
$\Rightarrow U_i(c) = P\{c_{i,j} = c\}$
$\Rightarrow V_i(t) = P\{r_{i,j+1} - r_{i,j} = t\}$

$$d_{i,j} = r_{i,j} + T_i$$

- $\begin{cases} U_i(c) & = & 0 & \forall & c & > & WCET_i \\ V_i(t) & = & 0 & \forall & t & < & MIT_i \end{cases}$ $\rightarrow$ Liu & Layland

Analysis

Two Cases:

SemiPeriodic Tasks	Generalized Sporadic Tasks
Variable Execution time	Constant Execution time
Constant Interarrival time	Variable Interarrival time
	
Used to handle CM streams	Activated by external events

SemiPeriodic Tasks - 1

- $U(c)$ generic; $V(t) = \begin{cases} 1 \quad if & t = T_i \\ 0 & \text{otherwise} \end{cases}$
- We assign $T_s = T_i$
- each T_s units of time:
  1. a request of c_j units arrives;
  2. at most Q_s units can be served.

SemiPeriodic Tasks - 2

- how much time have to be served when job $J_{i,j}$ arrives?
- v_j: units to be served at time jT_s:

$$
\begin{aligned}
v_1 &= c_1 \\
v_j &= max\{0, v_{j-1} - Q_s\} + c_{i,j}
\end{aligned}
$$

- $J_{i,j}$ will finish before $d_{i,j} = r_{i,j} + \left\lceil \frac{v_j}{Q_s} \right\rceil T_s$

Generalized Sporadic Tasks - 1

- $U(c) = \begin{cases} 1 \quad if & t = C_i \\ 0 & \text{otherwise} \end{cases}$; $V(t)$ generic.
- We assign $Q_s = C_i$
- The server deadline is increased by T_s each time that a job finishes $\rightarrow$ the server behave exactly as a TBS (or a $x = 1, y = T_i, D = T_i$ RBE task...)

$$d_{i,j} = \max\{r_{i,j}, d_{i,j-1}\} + T_s$$

Generalized Sporadic Tasks - 2

- The CBS can be modeled with a $G/D/1$ queue
- how much time is $J_{i,j}$'s deadline postponed? It depends by the jobs in queue when $J_{i,j}$ arrives...

$$w_j = d_{i,j} - r_{i,j} - T_s$$

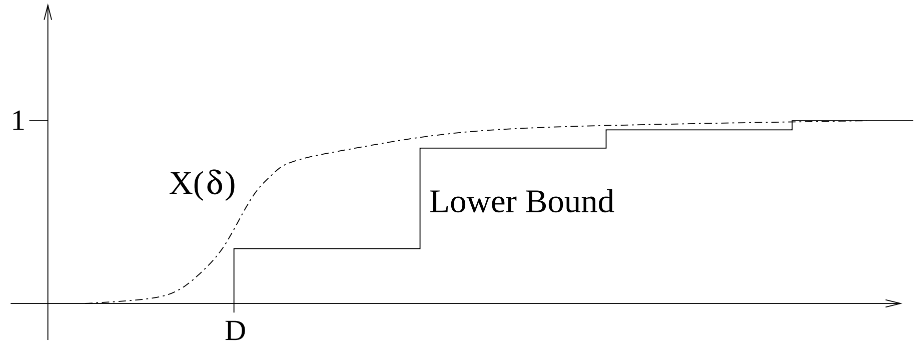
- $J_{i,j}$ will finish before $d_{i,j} = r_{i,j} + T_s + w_j$

Solution

- $\diamond$ The system is described by a Markov Chain (v_j or w_j)
- $\diamond$ State probability: $\pi_k^{(j)} = P\{v_j(\text{ or } w_j) = k\}$
- $\diamond$ if the chain is stable, stationary solution $\Pi = \lim_{j \to \infty} \Pi^{(j)}$ exists: $\Pi = M\Pi$.
- $\diamond$ the chain is stable if $\frac{E[U(c)]}{E[V(t)]} < B_i$

Solution

- Knowing Π, we can found a lower bound of the $X(\delta)$ CDF.



- $X_i(\delta) = P\{f_{i,j} < r_{i,j} + \delta_i\} >$ bound $\rightarrow$ pessimistic analysis.

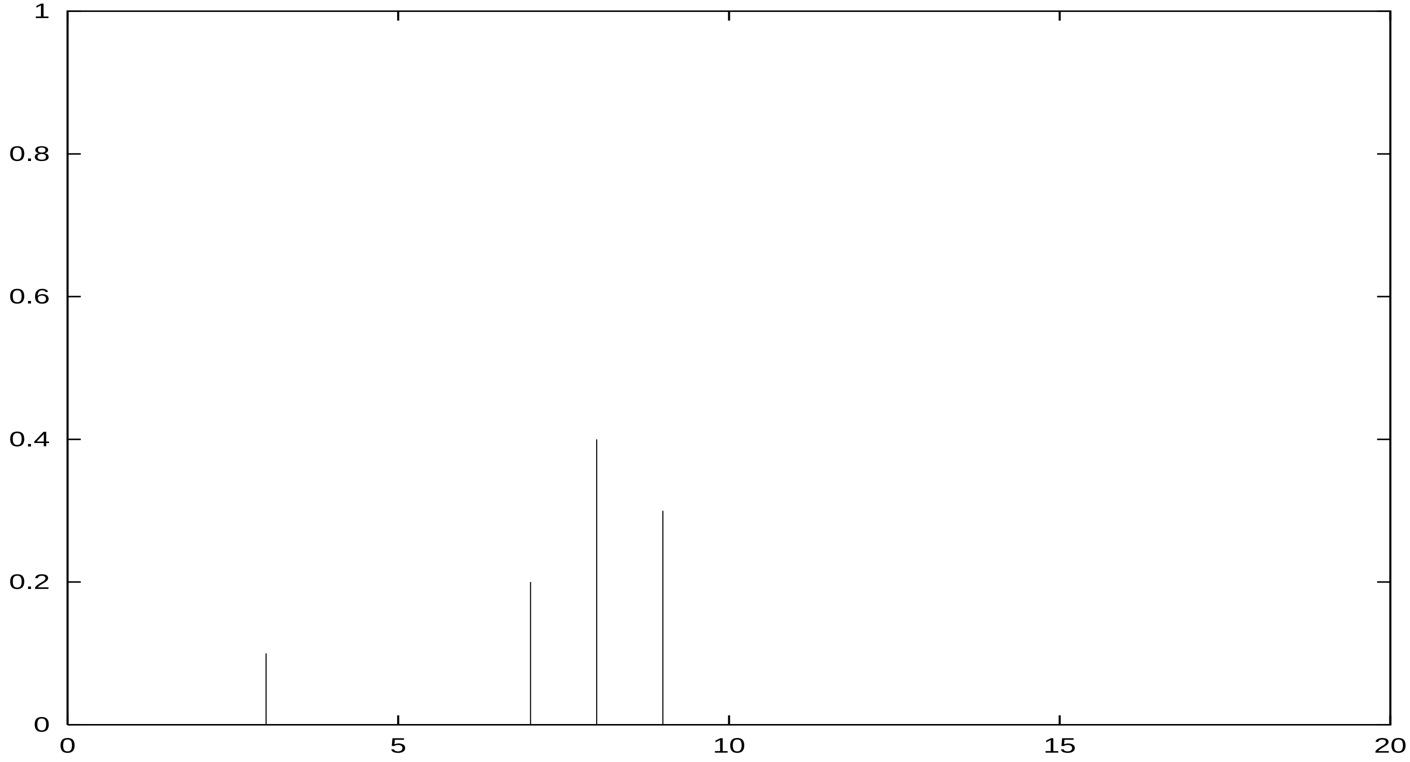
Example - 1

- Generalized sporadic task:

  $\Rightarrow c_{i,j} = C_i = 1 \forall j;$
  $\Rightarrow$ interarrival times distributed with $U(t)$ as in figure

- Hard guarantee: utilization $U = \frac{1}{3} = 0.333$

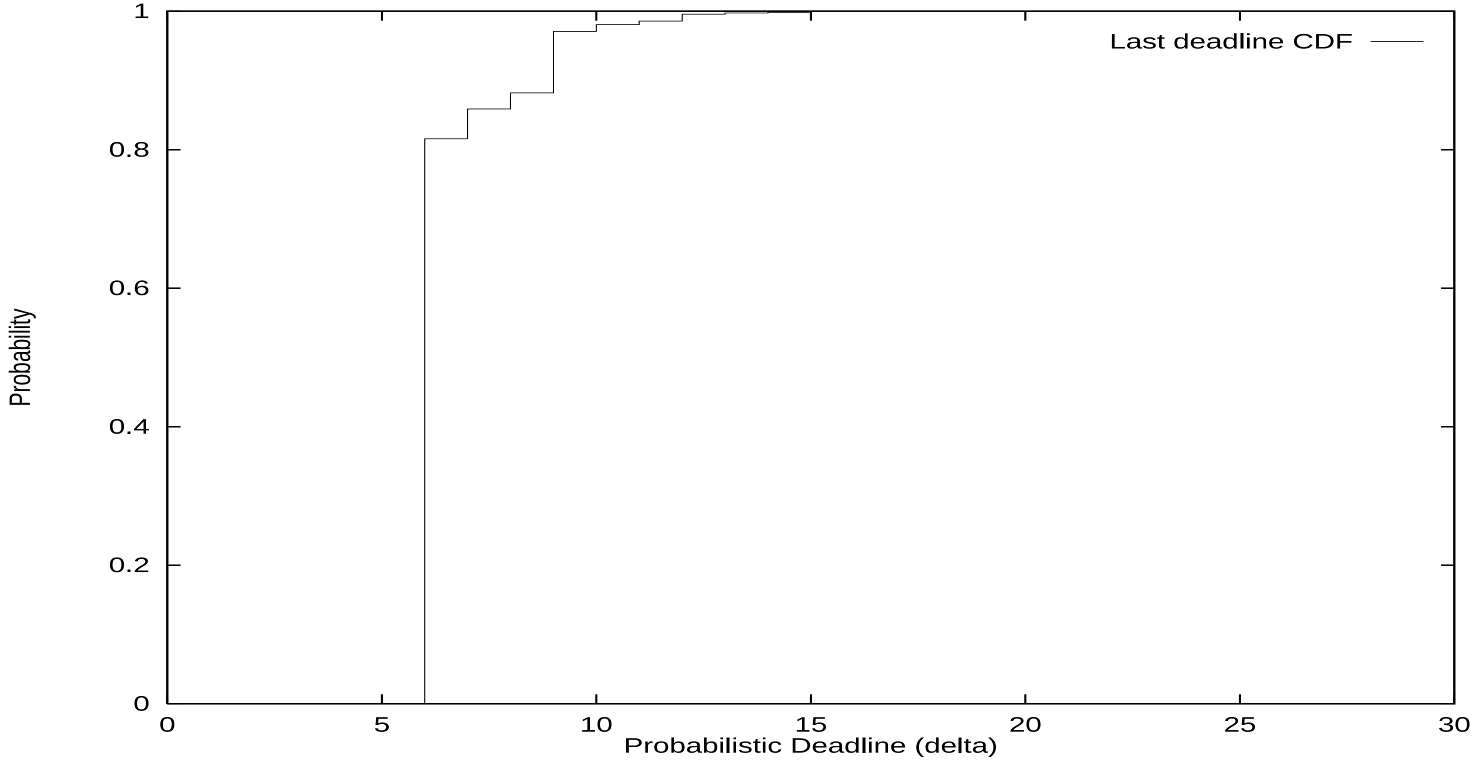
Interarrival Times PDF



Example - 2

- QoS guarantee: served with a $(1, T_s)$ CBS $\rightarrow B_i = \frac{1}{T_s}$
- The T_s parameter permits to choose the desired completion probability
- For Example, $T_s = 6 \rightarrow B_i = 0.16$.
- It results the $X(\delta)$ lower bound in figure:

Completion probability CDF Lower Bound



Unknown Task Parameters

◇ Server Bandwidth: $U_s = \frac{Q_s}{T_s}$

◇ **Problem**: How to assign Q_s and T_s???

⇒ if $Q_s > WCET$ and $T_s < MIT$, hard guarantee!!!

⇒ otherwise, the deadline will be postponed

⇒ CBS Scheduling Error:

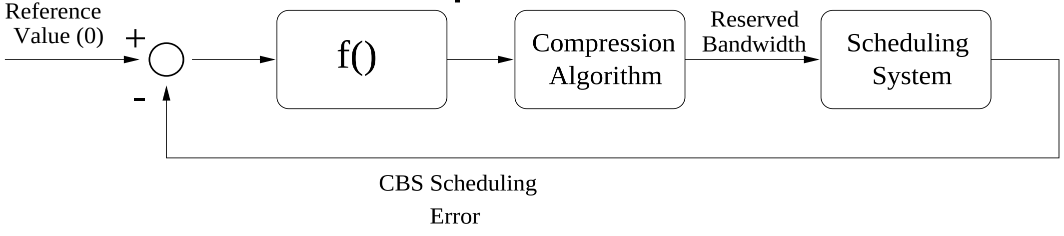
$$\epsilon = d_s - (r_{i,j} + T_s)$$

Adaptive Bandwidth Reservation

◇ Extension of the Bandwidth Reservation mechanism to dynamically adapt the reserved bandwidth

$\Rightarrow$ $T_s = T_i (= \frac{1}{R_i})$
$\Rightarrow$ Q_s starts from a small value
$\Rightarrow$ if $\epsilon > 0$, Q_s must be increased

◇ A feedback is introduced to perform Bandwidth Adaptation



Bandwidth Compression

- Aadaptive Bandwidth Reservation $\rightarrow$ the reserved bandwidth varies
- **Problem:** And if the reserved bandwidth increases too much?

  $\Rightarrow$ Schedulability condition: $\sum_j B_j \leq 1$
  $\Rightarrow$ The feedback can increase B_i such that $\sum_j B_j > 1$

- **Solution:** use a Bandwidth Compression algorithm

  $\Rightarrow$ Compressed bandwidth $\hat{B}_i = \begin{cases} B_i & \text{if } \sum_j B_j \leq 1 \\ f_i(B_1, \ldots, B_n) & \text{if } \sum_j B_j > 1 \end{cases}$

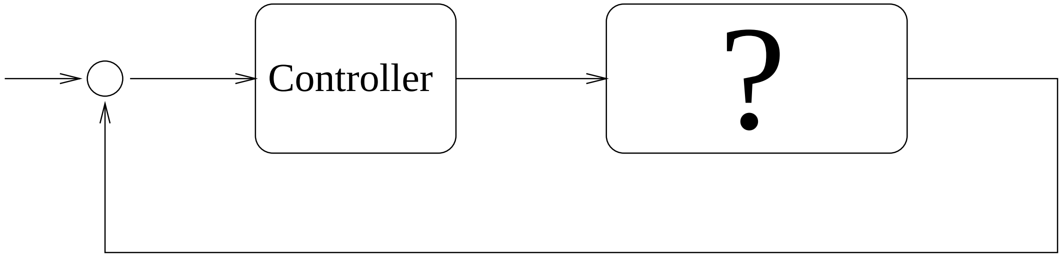
  $\Rightarrow$ $\sum_j \hat{B}_j \leq 1$

Compression Algorithms

- $f_i() : \forall (B_1, \ldots, B_n), \sum_i f_i(B_1, \ldots, B_n) \leq 1$
- Proportional Compression:
$$f_i(B_1, \ldots, B_n) = B_i w_i \frac{B_{max}}{\sum_j B_j w_j}$$
- Elastic Compression model

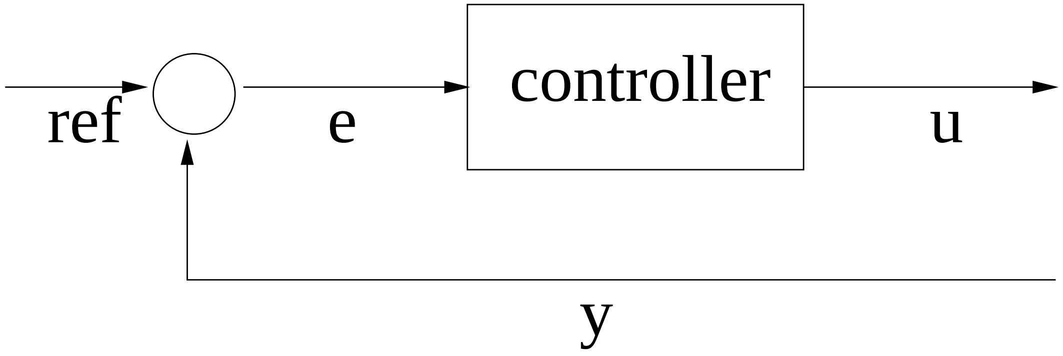
Adapting the Reserved Resources

- Adaptive Reservation
  - ⇒ Resource Reservation via CBS
  - ⇒ Closed Loop Adaptation of the server parameters



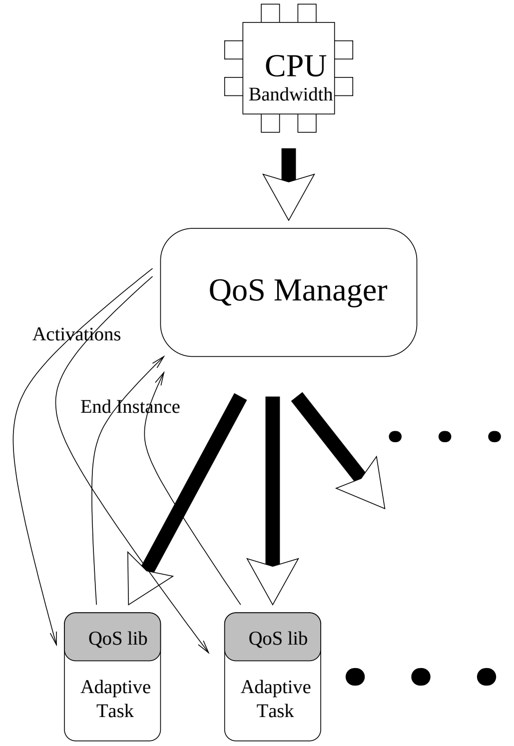
The Controller

- Takes $(y(k) - ref)$ as input
- Gives $u(k)$ as output



Implementation

- Implemented on the HARTIK kernel
- CBS scheduling in the kernel
- Adaptive Reservation through a Resource Manager



HARTIK

- Small Real-Time Kernel running on x86 PC (Protect Mode)
- EDF scheduling for hard tasks
- Periodic and Aperiodic soft tasks served through CBS
- Supports the most common hardware devices
- Adaptive Bandwidth reservations implemented through an external server
- Check http://hartik.sssup.it

Lego

- ◇ Real-Time Kernel providing flexible scheduling
- ◇ Runs on x86 PCs (Protect Mode); MultiBoot compliant
- ◇ Scheduling algorithms and resource protocols implemented by external modules
- ◇ DOS file system
- ◇ Web Site coming soon (informations: pj@hartik.sssup.it)

RTSIM

- ◇ Portable Scheduling simulator
- ◇ OO design; written in C++ (graphical interface in Java)
- ◇ Permits to model complex systems and simulate many scheduling algorithms
- ◇ RTCSIM → stay tuned!!!