Creating New Real-Time Scheduling Policies : Why and How ?

Nicolas NAVET Mathieu GRENIER

LORIA - INRIA
Real-Time and Interoperability Group (TRIO)
Nancy, France
Email: *Nicolas.Navet@loria.fr*
web: *http://www.loria.fr/~nnavet*

03/23/2005

Introduction

Real-Time Systems

Temporal correctness is mandatory

Scheduling : ordering the execution of activities - tasks or messages - is crucial and has been extensively studied since the 70s

Scheduling policies : Earliest Deadline First, Least-Laxity first, Fixed-Priority Preemptive, ...

Open question

Can we devise other scheduling policies of interest for real-time computing ?

Introduction

Real-Time Systems

Temporal correctness is mandatory

Scheduling : ordering the execution of activities - tasks or messages - is crucial and has been extensively studied since the 70s

Scheduling policies : Earliest Deadline First, Least-Laxity first, Fixed-Priority Preemptive, ...

Open question

Can we devise other scheduling policies of interest for real-time computing ?

Description of the scheduling

Characteristics of the tasks

$\tau_{k,n}$: n^{th} instance of τ_k

$A_{k,n}$: release date

$C_{k,n}$: computation time

$D_{k,n}$: absolute deadline

$\overline{D}_k$: relative deadline, i.e. $D_{k,n} = A_{k,n} + \overline{D}_k$

Results of the scheduling

$W_{k,n}(t)$: remaining work for $\tau_{k,n}$ at time t

$E_{k,n}$: end of execution time

$B_{k,n}$: beginning of execution

$R_{k,n} = E_{k,n} - A_{k,n}$: response time

Description of the scheduling

Characteristics of the tasks

$\tau_{k,n}$: n^{th} instance of τ_k

$A_{k,n}$: release date

$C_{k,n}$: computation time

$D_{k,n}$: absolute deadline

$\overline{D}_k$: relative deadline, i.e. $D_{k,n} = A_{k,n} + \overline{D}_k$

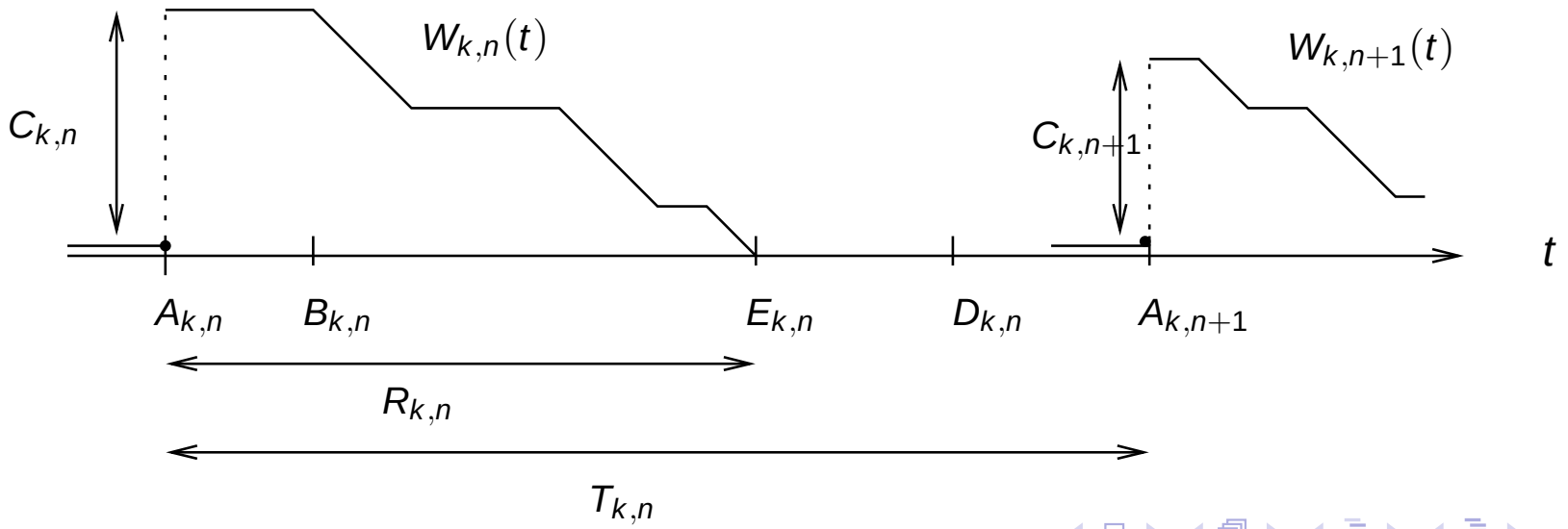
Results of the scheduling

$W_{k,n}(t)$: remaining work for $\tau_{k,n}$ at time t

$E_{k,n}$: end of execution time

$B_{k,n}$: beginning of execution

$R_{k,n} = E_{k,n} - A_{k,n}$: response time



Scheduling performance criteria

Mandatory : “feasibility” or “schedulability” of the system

- Feasible iff $\forall k \quad \hat{R}_k = \max_{n \in \mathbb{N}} R_{k,n} \leqslant \overline{D}_k$
- Techniques : simulation, feasibility test or **bound on response times**
- Feasibility has been a major research field since the 70s ..

Increasingly important criteria (e.g. for Networked Control Systems)

- **Average response times** (NCS: control delays)
- Variability (“jitter”) in successive $B_{k,n}$ (NCS: sampling times)
- Variability in successive $E_{k,n}$ (NCS: actuation times)
- Fault-tolerance: behaviour in overload condition or with transient faults

Scheduling performance criteria

Mandatory : “feasibility” or “schedulability” of the system

- Feasible iff $\forall k \quad \hat{R}_k = \max_{n \in \mathbb{N}} R_{k,n} \leqslant \overline{D}_k$
- Techniques : simulation, feasibility test or **bound on response times**
- Feasibility has been a major research field since the 70s ..

Increasingly important criteria (e.g. for Networked Control Systems)

- **Average response times** (NCS: control delays)
- Variability (“jitter”) in successive $B_{k,n}$ (NCS: sampling times)
- Variability in successive $E_{k,n}$ (NCS: actuation times)
- Fault-tolerance: behaviour in overload condition or with transient faults

Scheduling performance criteria

Mandatory : “feasibility” or “schedulability” of the system

- Feasible iff $\forall k \quad \hat{R}_k = \max_{n \in \mathbb{N}} R_{k,n} \leqslant \overline{D}_k$
- Techniques : simulation, feasibility test or **bound on response times**
- Feasibility has been a major research field since the 70s ..

Increasingly important criteria (e.g. for Networked Control Systems)

- **Average response times** (NCS: control delays)
- Variability (“jitter”) in successive $B_{k,n}$ (NCS: sampling times)
- Variability in successive $E_{k,n}$ (NCS: actuation times)
- Fault-tolerance: behaviour in overload condition or with transient faults

Networked Control systems

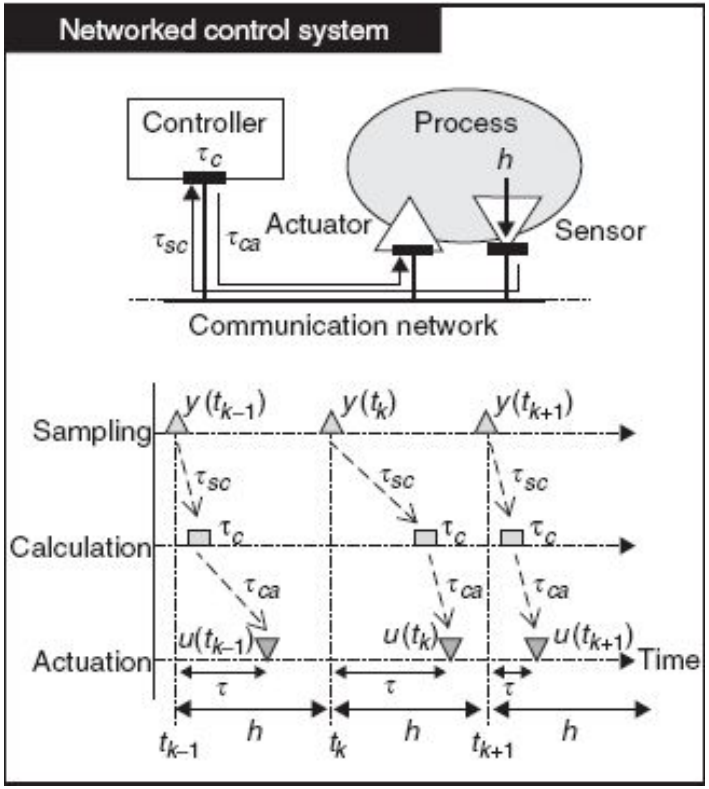


Figure: from [MaViFuFo:05]

Examples of NCS: plant control, building automation, in-vehicle systems , ..

Usual assumptions in control theory: sampling times and actuation times are strictly periodic

Improving control : existing work

1. Control theory : control laws robust to delays or integration of the delays as a parameter of the control law [ArCeEk:00]
2. Adapt task model : subdivide tasks in sampling - computation - actuation [BaRiViCr:04] or maximize frequency
3. Optimize parameters of well-known scheduling policies : FPP [GoRi:04], EDF [BaBuGoLi:99] or RR+FPP (Posix1003.1b) [NaMi:03]
4. Synthesize static schedules (off-line) : stored in timetables or using automata [AlGoSi:02]

Our proposal : find new on-line scheduling algorithms optimized for a particular system or a class of systems

- Can be used with 1) and 2)
- Better performance than 3)
- Better suited to industrial requirements than 4) : problem sizes, incremental design process, evolutivity, efficient use of resources, use of COTS...

Improving control : existing work

1. Control theory : control laws robust to delays or integration of the delays as a parameter of the control law [ArCeEk:00]
2. Adapt task model : subdivide tasks in sampling - computation - actuation [BaRiViCr:04] or maximize frequency
3. Optimize parameters of well-known scheduling policies : FPP [GoRi:04], EDF [BaBuGoLi:99] or RR+FPP (Posix1003.1b) [NaMi:03]
4. Synthesize static schedules (off-line) : stored in timetables or using automata [AlGoSi:02]

Our proposal : find new on-line scheduling algorithms optimized for a particular system or a class of systems

- Can be used with 1) and 2)
- Better performance than 3)
- Better suited to industrial requirements than 4) : problem sizes, incremental design process, evolutivity, efficient use of resources, use of COTS...

Class of scheduling policies under study

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \mid n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \mid n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \,|\, n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \,|\, n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \,|\, n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \,|\, n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)*
  $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Formally defining scheduling policies

Why ?

- Non-ambiguous description and implementation
- Identify distinct "classes" of policies for which generic results hold (e.g. feasibility) $\Longrightarrow$ unify existing work

How ? Priority functions [Mig99]

- $\Gamma_{k,n}(t)$ is the priority of instance $\tau_{k,n}$ at time t - the resource is granted by the *Highest Priority First* rule
- $\Gamma_{k,n}(t) \in \mathcal{P}$ with $\mathcal{P} = \{(p_1, \ldots, p_n) \in \mathbb{R}^n \,|\, n \in \mathbb{N}\}$
- A total order is defined on $\mathcal{P}$ *(leximin ordering)* $\Gamma_{k,n}(t) = (3,4,6) \preccurlyeq \Gamma_{i,j}(t) = (3,4,5)$ ($\tau_{i,j}$ has a lower priority than $\tau_{k,n}$ at time t)

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

Defining policies through priority functions

Examples

- First-In First-Out : $\Gamma_{k,n}^{FIFO} = (A_{k,n}, k, n)$
- Last-In First-Out : $\Gamma_{k,n}^{LIFO} = (-A_{k,n}, k, -n)$
- Shortest Job First : $\Gamma_{k,n}^{SJF} = (C_{k,n}, k, n)$
- Earliest Deadline First : $\Gamma_{k,n}^{EDF} = (A_{k,n} + \overline{D}_k, k, n)$
- $\Gamma_{k,n} = (A_{k,n} + \sin(D_{k,n}) + \frac{1}{10} \cdot C_{k,n}, k, n)$??

Issues

- Priority functions can be difficult to devise (e.g. Round Robin)
- Not all priority functions define “valid” scheduling policies !!

“Good” Real-time scheduling policies

A "good" scheduling policy (i.e. suited to real-time) shall be :

- Decidable [Mig99]: at any time t there is exactly one instance (with pending work) of maximal priority
- Implementable :
  - “piecewise order preserving” [Mig99] : during an interval of finite length, the number of changes of the highest priority instance is finite
  - coordinates of the priority functions must be representable with machine numbers
- “Shift temporal invariant” : the relative priority between any two instances does not depend on the numerical value of the clock

“Good” Real-time scheduling policies

A "good" scheduling policy (i.e. suited to real-time) shall be :

- Decidable [Mig99]: at any time t there is exactly one instance (with pending work) of maximal priority
- Implementable :
  - “piecewise order preserving” [Mig99] : during an interval of finite length, the number of changes of the highest priority instance is finite
  - coordinates of the priority functions must be representable with machine numbers
- “Shift temporal invariant” : the relative priority between any two instances does not depend on the numerical value of the clock

“Good” Real-time scheduling policies

A "good" scheduling policy (i.e. suited to real-time) shall be :

- Decidable [Mig99]: at any time t there is exactly one instance (with pending work) of maximal priority
- Implementable :
  - “piecewise order preserving” [Mig99] : during an interval of finite length, the number of changes of the highest priority instance is finite
  - coordinates of the priority functions must be representable with machine numbers
- “Shift temporal invariant” : the relative priority between any two instances does not depend on the numerical value of the clock

“Good” Real-time scheduling policies

A "good" scheduling policy (i.e. suited to real-time) shall be :

- Decidable [Mig99]: at any time t there is exactly one instance (with pending work) of maximal priority
- Implementable :
  - “piecewise order preserving” [Mig99] : during an interval of finite length, the number of changes of the highest priority instance is finite
  - coordinates of the priority functions must be representable with machine numbers
- “Shift temporal invariant” : the relative priority between any two instances does not depend on the numerical value of the clock

Domain of study (1/2)

"Time independent scheduling policies" $\Longrightarrow$ Preemptive except FIFO

- A policy $\mathcal{A}$ is time independent iff $\Gamma^{\mathcal{A}}_{k,n}(t) = \Gamma^{\mathcal{A}}_{k,n}(0) \quad \forall t$
- FIFO, LIFO, EDF, FPP, .. are time independent.

"Priority Promotion at Execution beginning" (PPEB) $\Longrightarrow$ Non-Preemptive

- $\mathcal{A}$ is a PPEB policy iff its priority function can be written as.

$$\Gamma^{\mathcal{A}}_{k,n}(t) = \begin{cases} \overrightarrow{\mathcal{P}^{\mathcal{A}}} \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

  where $\overrightarrow{\mathcal{P}^{\mathcal{A}}}$ defines a time independent policy.

- Property : there is one and only one change in the priority value - at the execution begin
- E.g. : Non-Preemptive EDF with $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$

Domain of study (1/2)

"Time independent scheduling policies" $\Longrightarrow$ Preemptive except FIFO

- A policy $\mathcal{A}$ is time independent iff $\Gamma^{\mathcal{A}}_{k,n}(t) = \Gamma^{\mathcal{A}}_{k,n}(0) \quad \forall t$
- FIFO, LIFO, EDF, FPP, .. are time independent.

"Priority Promotion at Execution beginning" (PPEB) $\Longrightarrow$ Non-Preemptive

- $\mathcal{A}$ is a PPEB policy iff its priority function can be written as.

$$\Gamma^{\mathcal{A}}_{k,n}(t) = \begin{cases} \overrightarrow{\mathcal{P}^{\mathcal{A}}} \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

  where $\overrightarrow{\mathcal{P}^{\mathcal{A}}}$ defines a time independent policy.

- Property : there is one and only one change in the priority value - at the execution begin
- E.g. : Non-Preemptive EDF with $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$

Domain of study (1/2)

"Time independent scheduling policies" $\Longrightarrow$ Preemptive except FIFO

- A policy $\mathcal{A}$ is time independent iff $\Gamma^{\mathcal{A}}_{k,n}(t) = \Gamma^{\mathcal{A}}_{k,n}(0) \quad \forall t$
- FIFO, LIFO, EDF, FPP, .. are time independent.

"Priority Promotion at Execution beginning" (PPEB) $\Longrightarrow$ Non-Preemptive

- $\mathcal{A}$ is a PPEB policy iff its priority function can be written as.

$$\Gamma^{\mathcal{A}}_{k,n}(t) = \begin{cases} \overrightarrow{\mathcal{P}^{\mathcal{A}}} \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

  where $\overrightarrow{\mathcal{P}^{\mathcal{A}}}$ defines a time independent policy.

- Property : there is one and only one change in the priority value - at the execution begin
- E.g. : Non-Preemptive EDF with $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$

Domain of study (1/2)

"Time independent scheduling policies" $\Longrightarrow$ Preemptive except FIFO

- A policy $\mathcal{A}$ is time independent iff $\Gamma^{\mathcal{A}}_{k,n}(t) = \Gamma^{\mathcal{A}}_{k,n}(0) \quad \forall t$
- FIFO, LIFO, EDF, FPP, .. are time independent.

"Priority Promotion at Execution beginning" (PPEB) $\Longrightarrow$ Non-Preemptive

- $\mathcal{A}$ is a PPEB policy iff its priority function can be written as.

$$\Gamma^{\mathcal{A}}_{k,n}(t) = \begin{cases} \overrightarrow{\mathcal{P}^{\mathcal{A}}} \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

  where $\overrightarrow{\mathcal{P}^{\mathcal{A}}}$ defines a time independent policy.

- Property : there is one and only one change in the priority value - at the execution begin
- E.g. : Non-Preemptive EDF with $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$

Domain of study (1/2)

"Time independent scheduling policies" $\Longrightarrow$ Preemptive except FIFO

- A policy $\mathcal{A}$ is time independent iff $\Gamma^{\mathcal{A}}_{k,n}(t) = \Gamma^{\mathcal{A}}_{k,n}(0) \quad \forall t$
- FIFO, LIFO, EDF, FPP, .. are time independent.

"Priority Promotion at Execution beginning" (PPEB) $\Longrightarrow$ Non-Preemptive

- $\mathcal{A}$ is a PPEB policy iff its priority function can be written as.

$$\Gamma^{\mathcal{A}}_{k,n}(t) = \begin{cases} \overrightarrow{\mathcal{P}^{\mathcal{A}}} \text{ if } t < B_{k,n} \\ (-\infty, k, n) \text{ if } t \geq B_{k,n} \end{cases}$$

  where $\overrightarrow{\mathcal{P}^{\mathcal{A}}}$ defines a time independent policy.

- Property : there is one and only one change in the priority value - at the execution begin
- E.g. : Non-Preemptive EDF with $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$

Domain of study (2/2)

Why are PPEB important in practice ?

- "Good" scheduling policies easily implementable on low-cost COTS hardware : Controller Area Network or Ethernet
- Important expressiveness ! All widely used non-preemptive scheduling policies belong to that class

Sub-class of PPEB: Arrival Time Dependent (ATD) policies

- $\overrightarrow{P^{ATD}} = (A_{k,n} + p_k, k, n)$ where p_k depends from τ_k - in the following $p_k : \overline{D}_k \times C_k \mapsto \mathbf{R}$
- Why ? good tradeoff can be found between
  - EDF is "optimal" for schedulability - $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$
  - JSF is "optimal" for response times - $\overrightarrow{\mathcal{P}^{\mathcal{JSF}}} = (C_k, k, n)$

Domain of study (2/2)

Why are PPEB important in practice ?

- "Good" scheduling policies easily implementable on low-cost COTS hardware : Controller Area Network or Ethernet
- Important expressiveness ! All widely used non-preemptive scheduling policies belong to that class

Sub-class of PPEB: Arrival Time Dependent (ATD) policies

- $\overrightarrow{P^{ATD}} = (A_{k,n} + p_k, k, n)$ where p_k depends from τ_k - in the following $p_k : \overline{D}_k \times C_k \mapsto \mathbb{R}$
- Why ? good tradeoff can be found between
  - EDF is "optimal" for schedulability - $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$
  - JSF is "optimal" for response times - $\overrightarrow{\mathcal{P}^{\mathcal{JSF}}} = (C_k, k, n)$

Domain of study (2/2)

Why are PPEB important in practice ?

- "Good" scheduling policies easily implementable on low-cost COTS hardware : Controller Area Network or Ethernet
- Important expressiveness ! All widely used non-preemptive scheduling policies belong to that class

Sub-class of PPEB: Arrival Time Dependent (ATD) policies

- $\overrightarrow{P^{ATD}} = (A_{k,n} + p_k, k, n)$ where p_k depends from τ_k - in the following $p_k : \overline{D}_k \times C_k \mapsto \mathbb{R}$
- Why ? good tradeoff can be found between
  - EDF is "optimal" for schedulability - $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$
  - JSF is "optimal" for response times - $\overrightarrow{\mathcal{P}^{\mathcal{JSF}}} = (C_k, k, n)$

Domain of study (2/2)

Why are PPEB important in practice ?

- "Good" scheduling policies easily implementable on low-cost COTS hardware : Controller Area Network or Ethernet
- Important expressiveness ! All widely used non-preemptive scheduling policies belong to that class

Sub-class of PPEB: Arrival Time Dependent (ATD) policies

- $\overrightarrow{P^{ATD}} = (A_{k,n} + p_k, k, n)$ where p_k depends from τ_k - in the following $p_k : \overline{D}_k \times C_k \mapsto \mathbb{R}$
- Why ? good tradeoff can be found between
  - EDF is "optimal" for schedulability - $\overrightarrow{\mathcal{P}^{\mathcal{EDF}}} = (A_{k,n} + \overline{D}_k, k, n)$
  - JSF is "optimal" for response times - $\overrightarrow{\mathcal{P}^{\mathcal{JSF}}} = (C_k, k, n)$

Overview of the approach

Input of the problem

- Search space : class of ATD policies
- Description of the application:
  - set of tasks
  - performance criteria

Steps

1. Select one candidate policy $\mathcal{A}$ among ATD policies with $\Gamma^{\mathcal{A}}_{k,n} = (A_{k,n}+?, k, n)$

   $\mapsto$ "search" technique (random, GA,GP, ..)

1. Test feasibility : $\Gamma^{\mathcal{A}}_{k,n} \times taskset \mapsto \{feasible, non-feasible\}$

   $\mapsto$ generic feasibility analysis

1. If feasible evaluate other criteria : simulation or monitoring

   $\mapsto$ experiment platform

Overview of the approach

Input of the problem

- Search space : class of ATD policies
- Description of the application:
  - set of tasks
  - performance criteria

Steps

1. Select one candidate policy $\mathcal{A}$ among ATD policies with $\Gamma^{\mathcal{A}}_{k,n} = (A_{k,n}+?, k, n)$

   $\mapsto$ "search" technique (random, GA,GP, ..)

1. Test feasibility : $\Gamma^{\mathcal{A}}_{k,n} \times taskset \mapsto \{feasible, non-feasible\}$

   $\mapsto$ generic feasibility analysis

1. If feasible evaluate other criteria : simulation or monitoring

   $\mapsto$ experiment platform

Overview of the approach

Input of the problem

- Search space : class of ATD policies
- Description of the application:
  - set of tasks
  - performance criteria

Steps

1. Select one candidate policy $\mathcal{A}$ among ATD policies with $\Gamma^{\mathcal{A}}_{k,n} = (A_{k,n}+?, k, n)$

   $\mapsto$ "search" technique (random, GA,GP, ..)

1. Test feasibility : $\Gamma^{\mathcal{A}}_{k,n} \times taskset \mapsto \{feasible, non-feasible\}$

   $\mapsto$ generic feasibility analysis

1. If feasible evaluate other criteria : simulation or monitoring

   $\mapsto$ experiment platform

Overview of the approach

Input of the problem

- Search space : class of ATD policies
- Description of the application:
  - set of tasks
  - performance criteria

Steps

1. Select one candidate policy $\mathcal{A}$ among ATD policies with $\Gamma^{\mathcal{A}}_{k,n} = (A_{k,n}+?, k, n)$

   $\mapsto$ "search" technique (random, GA,GP, ..)

1. Test feasibility : $\Gamma^{\mathcal{A}}_{k,n} \times taskset \mapsto \{feasible, non-feasible\}$

   $\mapsto$ generic feasibility analysis

1. If feasible evaluate other criteria : simulation or monitoring

   $\mapsto$ experiment platform

“Generic” schedulability analysis

Response time basics

$S_{I_{k,n}}(t_1, t_2)$: amount of work of priority higher or equal to $\tau_{k,n}$ released between t_1 and t_2 (Work Arrival Function)

$U_{k,n}$: largest date before $A_{k,n}$ such that no higher priority work is pending

Figure: $E_{k,n} = U_{k,n} + \min\{x > 0 \mid S_{I_{k,n}}(U_{k,n}, U_{k,n}+x) = x\}$

Response time basics

$S_{I_{k,n}}(t_1, t_2)$: amount of work of priority higher or equal to $\tau_{k,n}$ released between t_1 and t_2 (Work Arrival Function)

$U_{k,n}$: largest date before $A_{k,n}$ such that no higher priority work is pending

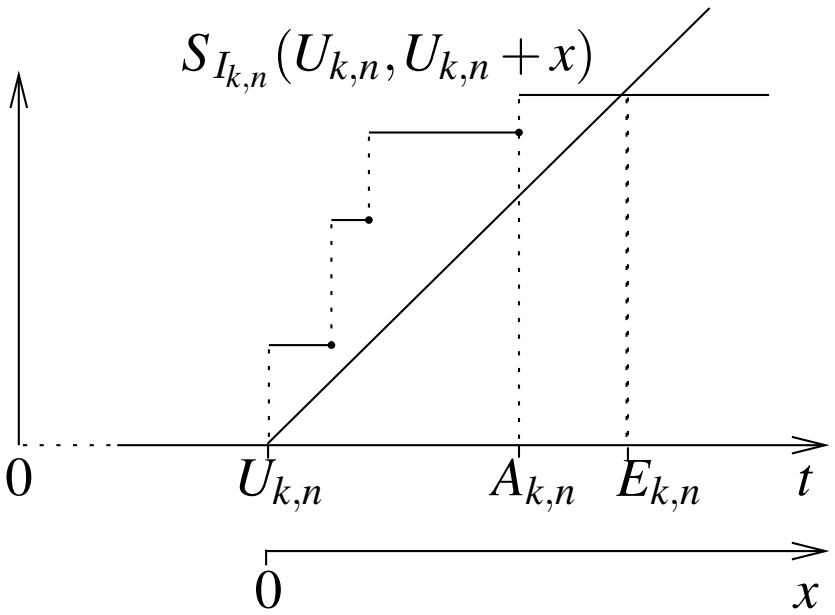


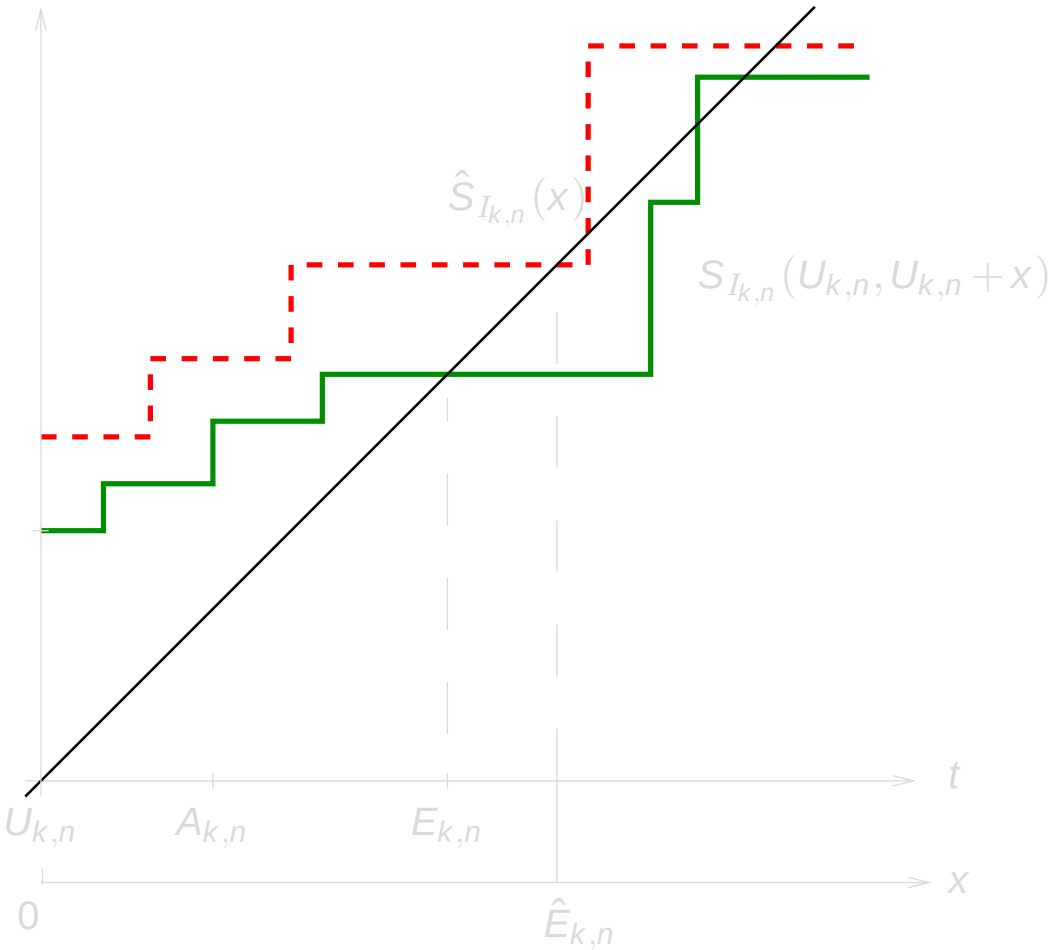
Figure: $E_{k,n} = U_{k,n} + \min\{x > 0 \mid S_{I_{k,n}}(U_{k,n}, U_{k,n}+x) = x\}$

Bound on response times (1/2)

Problem: one cannot evaluate all $R_{k,n}$'s !

A solution: bound the amount of higher priority work - Majorizing

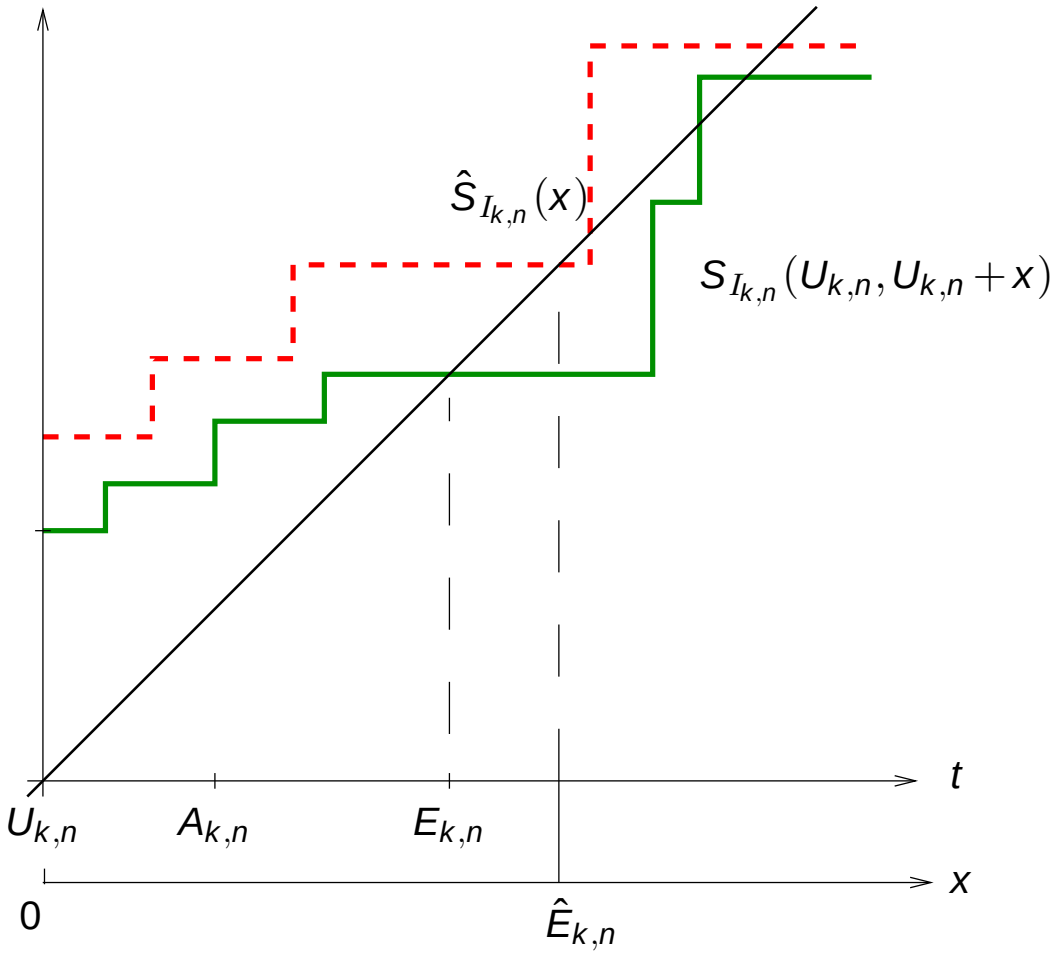
WAF: $S_{I_{k,n}}(u, u+t) \leqslant \hat{S}_{I_{k,n}}(t) \; \forall u \geqslant 0, t \geqslant 0$



Bound on response times (1/2)

Problem: one cannot evaluate all $R_{k,n}$'s !

A solution: bound the amount of higher priority work - Majorizing WAF: $S_{I_{k,n}}(u, u+t) \leqslant \hat{S}_{I_{k,n}}(t) \, \forall \, u \geqslant 0, t \geqslant 0$



Bound on response times (2/2)

Issues

- The bound $\hat{S}_{I_{k,n}}(t)$ depends on $A_{k,n}$
- Lemma 4 in [Mig99]: a bound can be found by analysing all possible arrival times in the longest CPU busy period $\Longrightarrow$ not usable as is
- In practice: determine "by hand" for a particular policy the set of arrival times that lead to changes in $\hat{S}_{I_{k,n}}(t)$ $\Longrightarrow$ not an algorithm

Our contribution: "generic" schedulability analysis

- An algorithm that determines from the priority function
  - the set of arrival times
  - the functions $\hat{S}_{I_{k,n}}(t)$ for which to compute the first fixed-point

Bound on response times (2/2)

Issues

- The bound $\hat{S}_{I_{k,n}}(t)$ depends on $A_{k,n}$
- Lemma 4 in [Mig99]: a bound can be found by analysing all possible arrival times in the longest CPU busy period $\Longrightarrow$ not usable as is
- In practice: determine "by hand" for a particular policy the set of arrival times that lead to changes in $\hat{S}_{I_{k,n}}(t)$ $\Longrightarrow$ not an algorithm

Our contribution: "generic" schedulability analysis

- An algorithm that determines from the priority function
  - the set of arrival times
  - the functions $\hat{S}_{I_{k,n}}(t)$ for which to compute the first fixed-point

Search techniques and Experiments

Exploration of the solution space

Issues

- Size of the search space is infinite
- Schedulability analysis can be time-consuming if the system is highly loaded
- Evaluation of other criteria through simulation or measurements is time-consuming

Tested :

- Exhaustive search with a chosen “granularity”
- Iterated hill-climbing with tabu list (neighborhood method)

Possible : all optimization techniques

Genetic Algorithm, Genetic Programming, Simulated Annealing...

Exploration of the solution space

Issues

- Size of the search space is infinite
- Schedulability analysis can be time-consuming if the system is highly loaded
- Evaluation of other criteria through simulation or measurements is time-consuming

Tested :

- Exhaustive search with a chosen “granularity”
- Iterated hill-climbing with tabu list (neighborhood method)

Possible : all optimization techniques

Genetic Algorithm, Genetic Programming, Simulated Annealing...

Exploration of the solution space

Issues

- Size of the search space is infinite
- Schedulability analysis can be time-consuming if the system is highly loaded
- Evaluation of other criteria through simulation or measurements is time-consuming

Tested :

- Exhaustive search with a chosen “granularity”
- Iterated hill-climbing with tabu list (neighborhood method)

Possible : all optimization techniques

Genetic Algorithm, Genetic Programming, Simulated Annealing...

Experiments (1/2)

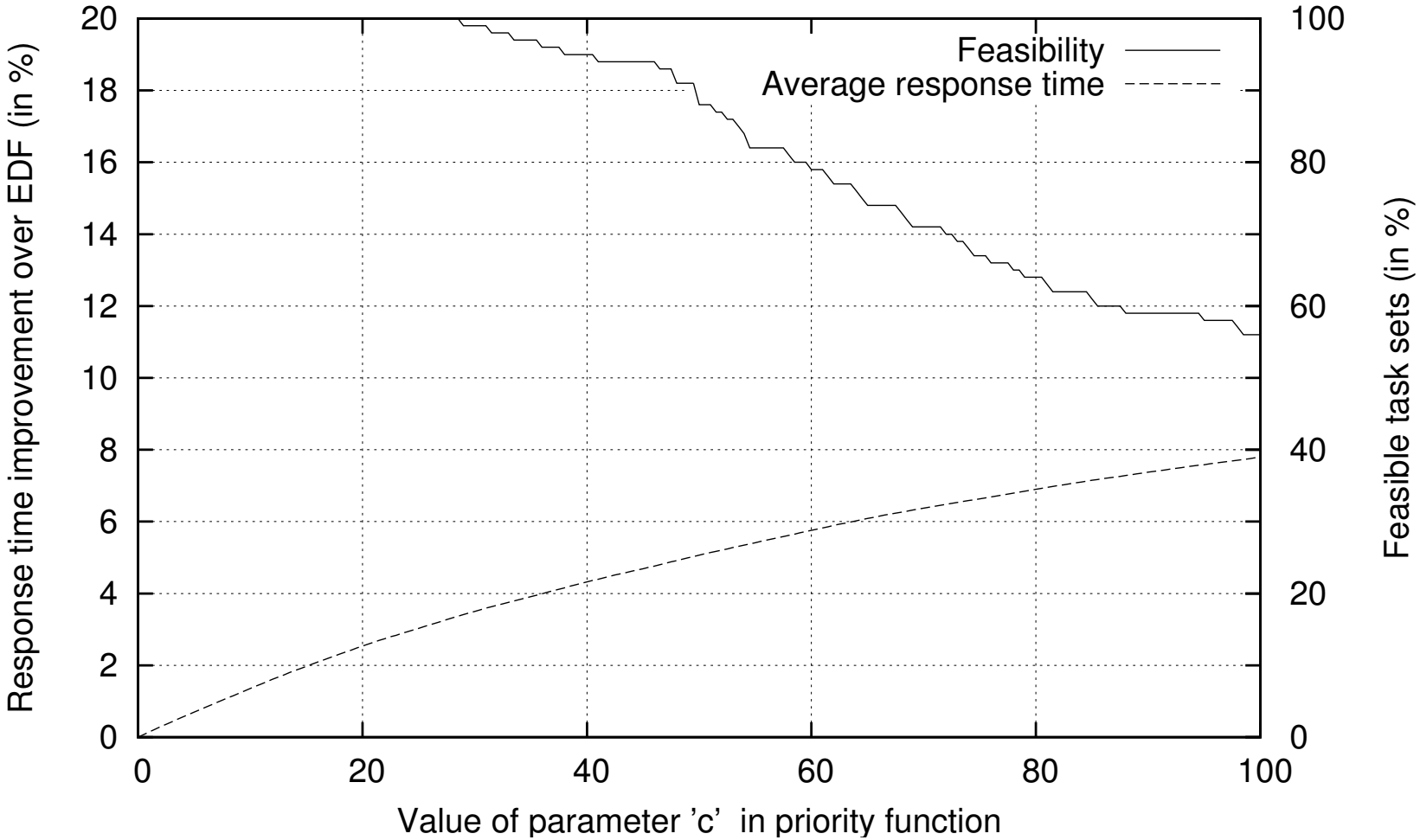


Figure: Performance of ATD policies defined by $\overrightarrow{\mathcal{P}} = (A_{k,n} + c \cdot C_{k,n} + \overline{D}_k, k, n)$ when the value of c increases - Each point = 1500 random experiments with 20 tasks - $<$ 1% of task sets feasible with SJF.

Experiments (2/2)

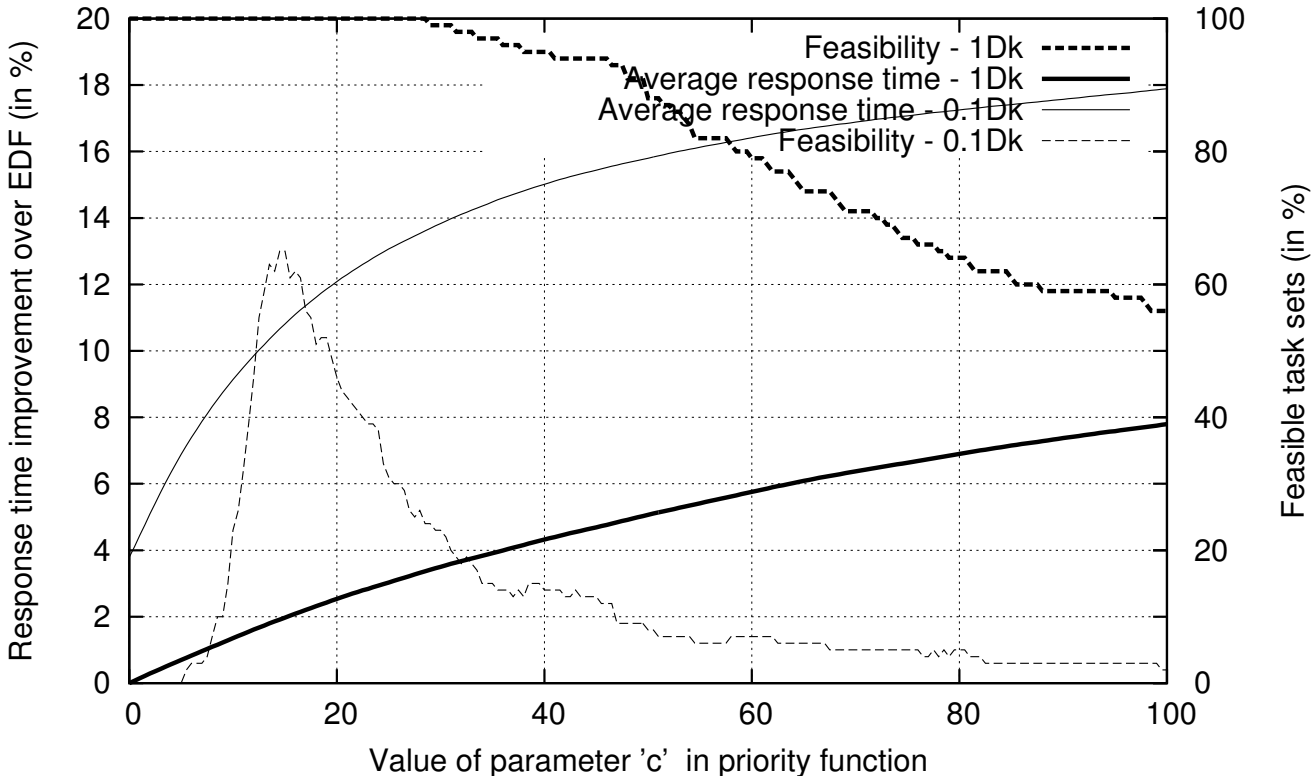


Figure: Same as before + policies defined by $\overrightarrow{\mathcal{P}} = (A_{k,n} + c \cdot C_{k,n} + \frac{1}{10}\overline{D}_k, k, n)$

Much better improvements ($>$ 25%) for a particular task set !

Experiments (2/2)

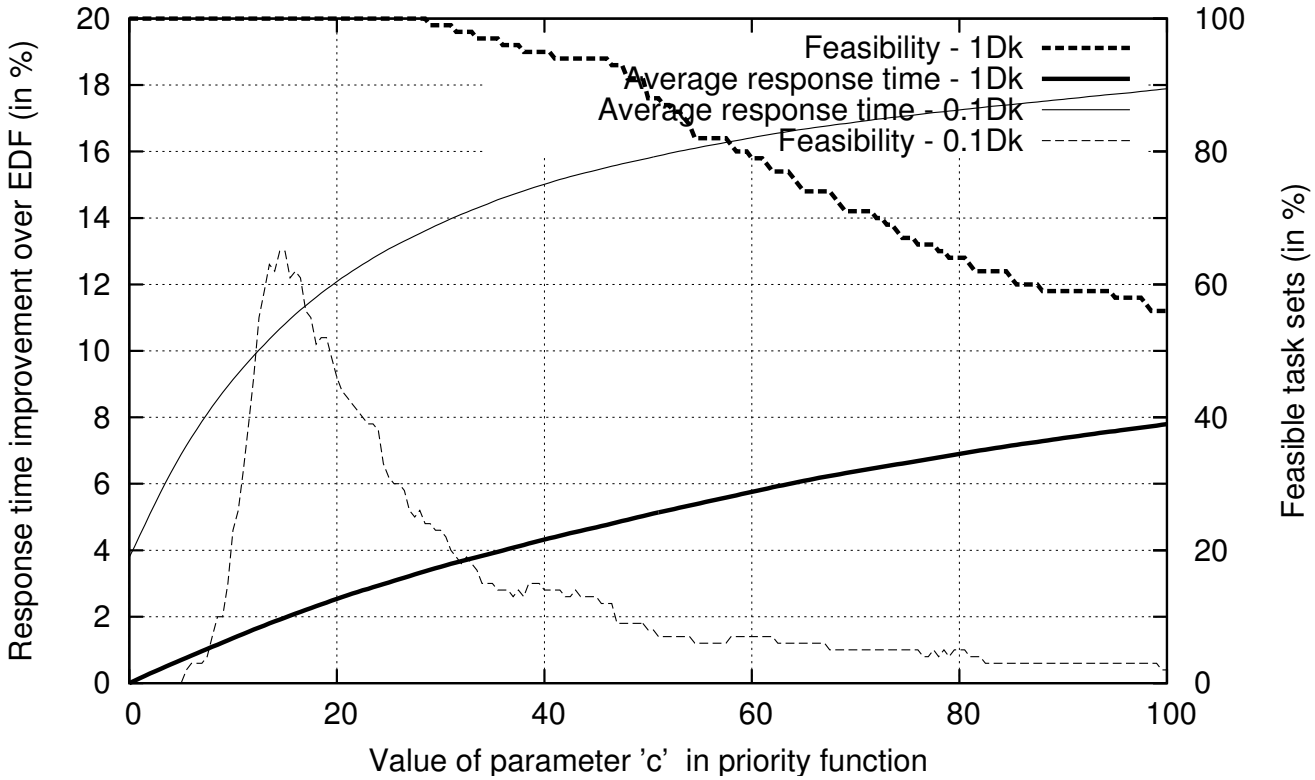


Figure: Same as before + policies defined by $\overrightarrow{\mathcal{P}} = (A_{k,n} + c \cdot C_{k,n} + \frac{1}{10}\overline{D}_k, k, n)$

Much better improvements ($>$ 25%) for a particular task set !

Summary

There are “other” on-line policies of interest for real-time computing:

- feasibility can be analyzed (sometimes better than EDF!)
- optimized for one or several criteria
- easily implementable on COTS O.S. or Network
- the choice of the policy can be automatized in the development process

Perspective : Time sharing policies and multi-policy systems

Software tools : downloadable at http://www.loria.fr/~nnavet

- Simulator for time-independent policies : Java applet with chronogram
- Software layer for scheduling with time independent policies on top of Posix1003.1b systems

Summary

There are “other” on-line policies of interest for real-time computing:

- feasibility can be analyzed (sometimes better than EDF!)
- optimized for one or several criteria
- easily implementable on COTS O.S. or Network
- the choice of the policy can be automatized in the development process

Perspective : Time sharing policies and multi-policy systems

Software tools : downloadable at http://www.loria.fr/~nnavet

- Simulator for time-independent policies : Java applet with chronogram
- Software layer for scheduling with time independent policies on top of Posix1003.1b systems

Summary

There are “other” on-line policies of interest for real-time computing:

- feasibility can be analyzed (sometimes better than EDF!)
- optimized for one or several criteria
- easily implementable on COTS O.S. or Network
- the choice of the policy can be automatized in the development process

Perspective : Time sharing policies and multi-policy systems

Software tools : downloadable at http://www.loria.fr/~nnavet

- Simulator for time-independent policies : Java applet with chronogram
- Software layer for scheduling with time independent policies on top of Posix1003.1b systems

Bibliography

The results presented in this talk can be found in [GrNa:05].


[Mig:99] J. Migge, Scheduling of recurrent tasks on one processor: A trajectory based Model, PhD Thesis, University of Nice Sophia-Antipolis, 1999.


[GrNa:05] M. Grenier and N. Navet, Non-Preemptive Real-Time Scheduling with Arrival Time Dependent Policies, submitted to IEEE ETFA'05, to appear as INRIA Research Report, Avril 2005.


[NaMi:03] N. Navet, J. Migge, Fine Tuning the Scheduling of Tasks through a Genetic Algorithm: Application to Posix1003.1b Compliant Systems, IEE Proceedings Software, 150(1):13-24, 2003.


[BaBuGoLi:99] S. Baruah, G. Buttazzo, S. Gorinsky and G. Lipari, Scheduling Periodic Task Systems to Minimize Output Jitter, International Conference on Real-Time Computing Systems and Applications, pp 62-69, 1999.


[BaRiViCr:04] P. Balbastre, I. Ripoll, J. Vidal, A. Crespo, A Task Model to Reduce Control Delays, Real-Time Systems, 27:215-236, 2004.


[GoRi:04] J. Goossens, P. Richard, Performance Optimization for Hard Real-Time Fixed Priority Tasks, International Conference on Real-Time Systems, Paris, Avril, 2004.


[MaViFuFo:05] P. Marti, R. Villà, J. Fuertes and G. Fohler, Networked Control Systems Overview, The industrial Information Technology Handbook, CRC Press, 2005.


[AlGoSi:02] K. Altisen, G. Gössler, J. Sifakis, Scheduler Modeling based on the Controller Synthesis Paradigm, Real-Time Systems, 23:55-84, 2002.


[ArCeEk00] K.E. Arzen, A. Cervin, J. Ecker, An Introduction to Control and Scheduling Co-Design, Conference on Control and Decision, Sydney, Australia, 2000.

Appendix : computation of response time bounds for periodic tasks scheduled under ATD policies

- Set of significant arrival dates for ATD policies ($p_k = \Gamma_{k,n}[1] - A_{k,n}$)

  $$a \in \{t = \lceil n \times T_k + p_k - p_i \rceil \mid t > 0, t < \mathcal{L} - C_k, i = 1..m, n \in \mathbb{N}\}.$$

- $\forall a$ compute first fixed-point of
  - Workload of higher priority tasks:

$$W_i(a,t) = \sum_{k \neq i} \underbrace{\left(\min\left(\left\lfloor \frac{t}{T_k} \right\rfloor + 1, \left\lfloor \frac{a + p_i - p_k}{T_k} \right\rfloor + 1 \right) \right)^+}_{\text{maximum number of instances in interval } t} . C_k \ ,$$

  - + $\mathcal{B}_i(a)$ the blocking factor (lower priority task already in execution at time a)

$$\mathcal{B}_i(a) \quad = \quad \max_{i=1..m} \{C_i \mid p_i > a + p_k\}$$