Adaptive Power Management for Real-Time Event Streams

Kai Huang[1] Luca Santinelli[2] Jian-Jia Chen[1]
Lothar Thiele[1] Giorgio C. Buttazzo[2]

[1]Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland

[2]Scuola Superiore Sant'Anna of Pisa, Italy

January 19, 2010

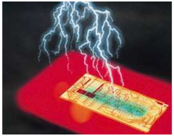
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

Problem Context

Power Dissipation

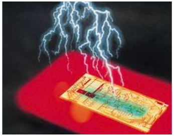
- Dynamic power consumption
- Static power consumption (leakage)



Problem Context

Power Dissipation

- Dynamic power consumption
- Static power consumption (leakage)



Energy Saving

- Dynamic Voltage Scaling
- Dynamic Power Management

Motivation

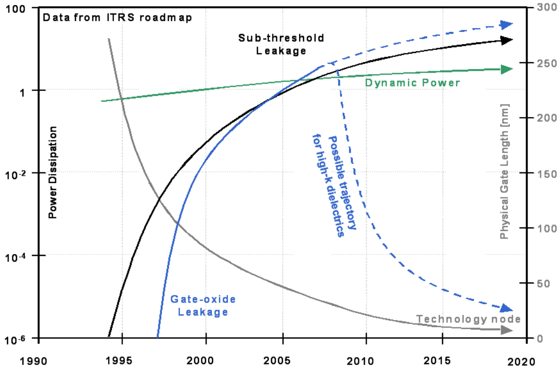


Figure1: ITRS Technology Roadmap: Power Trends.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

Motivation

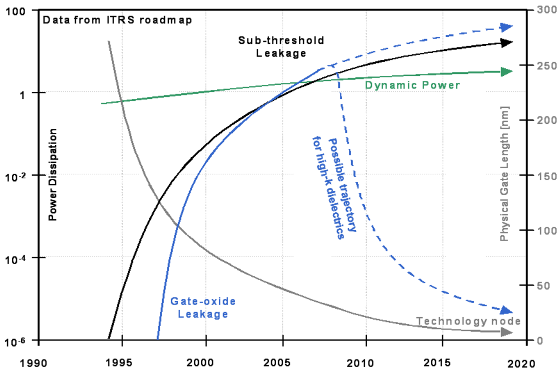


Figure1: ITRS Technology Roadmap: Power Trends.

The leakage power is comparable to or even more than the dynamic power dissipation

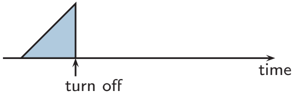
⇒ *Reducing the leakage power is crucial*

Challenges

- When to turn off
  - mode switch overhead

  $\Rightarrow$ Break Even Time

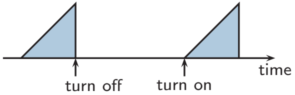
- When to turn off
  - mode switch overhead
  - ⇒ Break Even Time



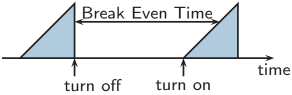
Challenges

- When to turn off
  - ▷ mode switch overhead

⇒ Break Even Time

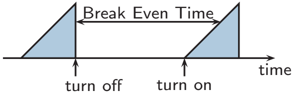


- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time



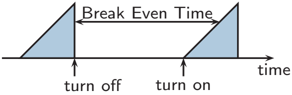
Challenges

- When to turn off
  - mode switch overhead
  - ⇒ Break Even Time



- When to turn on
  - as late as possible
  - cope with future burstiness

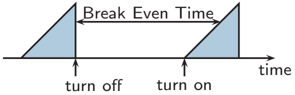
- When to turn off
  - mode switch overhead
  - ⇒ Break Even Time

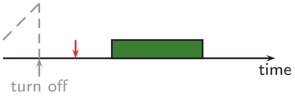


- When to turn on
  - as late as possible
  - cope with future burstiness

Challenges

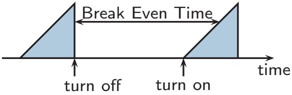
- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness



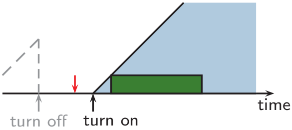


Challenges

- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time

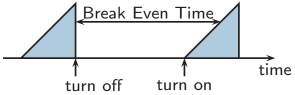


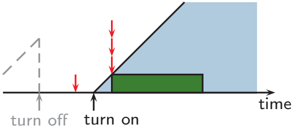
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness



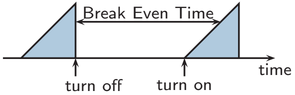
Challenges

- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness

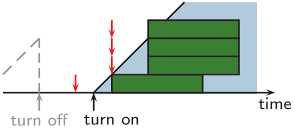




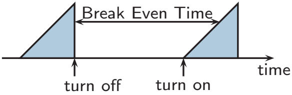
- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time



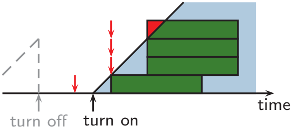
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness



- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time



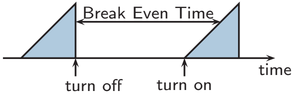
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness

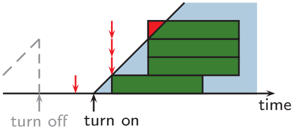


Challenges

- When to turn off
  - ▷ mode switch overhead
  - ⇒ Break Even Time
- When to turn on
  - ▷ as late as possible
  - ▷ cope with future burstiness

⇒ more complex for multiple streams





- Extend our online algorithms in RTSS'09 which adaptively control the power mode of a system (device) by
  - predicting the next moment for mode switch by considering both historical and future event arrivals
  - procrastinating the buffered and future events as late as possible without violating the timing and backlog constraints
- Propose method to cope with multiple streams with
  - preemptive fixed-priority scheduling
  - preemptive earliest-deadline-first scheduling
- Apply to streams with different characteristics to demonstrate the effectiveness

Outline

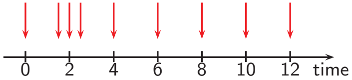
1. Introduction
2. Underlying Mathematical Model
3. Our Algorithms
4. Experimental Results

Model of Event Arrivals

- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....

Model of Event Arrivals

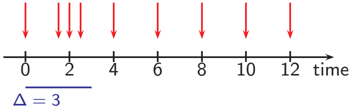
- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



Time domain

Model of Event Arrivals

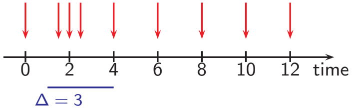
- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - . . . .
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



Time domain

Model of Event Arrivals

- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s

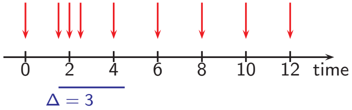


Time domain

Model of Event Arrivals

- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves

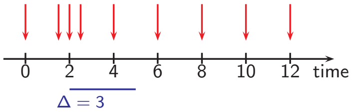
  ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



Time domain

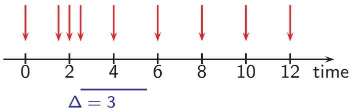
Model of Event Arrivals

- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - . . . .
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



Time domain

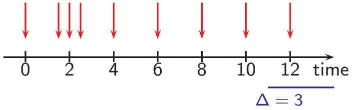
- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



Time domain

Model of Event Arrivals

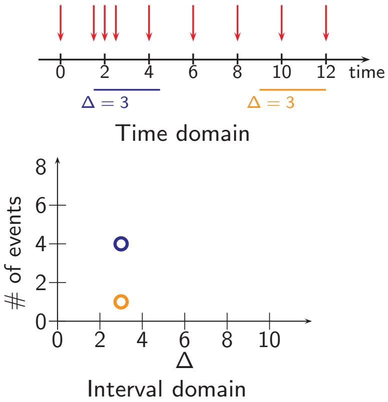
- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



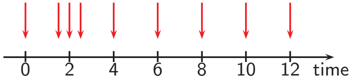
Time domain

Model of Event Arrivals

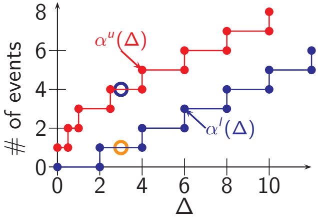
- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - ....
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - . . . .
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s



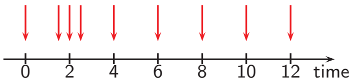
Time domain



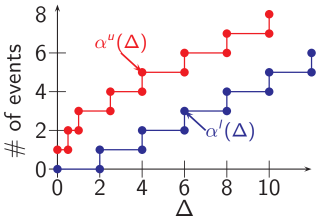
Interval domain

Model of Event Arrivals

- Traditionally
  - Periodic Real-Time Tasks
  - Sporadic Real-Time Tasks
  - . . . .
- Nowadays
  - Arrival curves
  - ⇒ Maximum/minimum arriving demand in *any interval* of length, e.g. 3 s
  - Arrival curves generalize traditional event models and are suitable to represent complex characteristics of event streams



Time domain

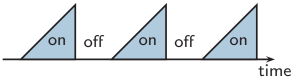


Interval domain

- Service curves
  - ⇒ Maximum/minimum available service in *any interval* of length Δ for the whole time span

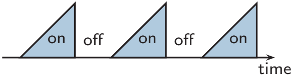
Model of Resource Service

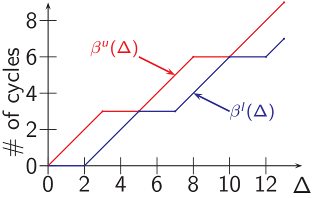
- Service curves
  - ⇒ Maximum/minimum available service in *any interval* of length Δ for the whole time span



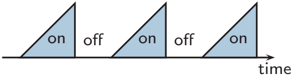
Model of Resource Service

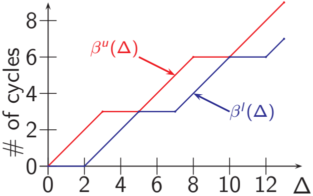
- Service curves
  - ⇒ Maximum/minimum available service in *any interval* of length Δ for the whole time span





- Service curves
  - ⇒ Maximum/minimum available service in *any interval* of length Δ for the whole time span





  - ▷ Service curves generalize different resource models

- Delay and Backlog Analysis

  Given:

  - α is the stream arrival curve
  - β is the service guarantee

  $\Rightarrow$ maximum delay D

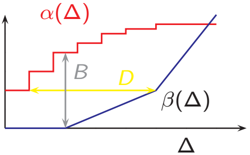
  $\Rightarrow$ maximum backlog B

- Delay and Backlog Analysis

  Given:
  - α is the stream arrival curve
  - β is the service guarantee

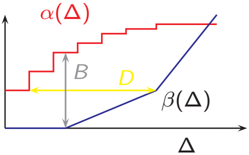
  $\Rightarrow$ maximum delay D

  $\Rightarrow$ maximum backlog B



- Delay and Backlog Analysis

  Given:
  - α is the stream arrival curve
  - β is the service guarantee
  - $\Rightarrow$ maximum delay D
  - $\Rightarrow$ maximum backlog B



- Scheduling Analysis

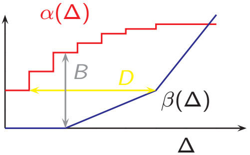
  Suppose:
  - α is the stream arrival curve
  - D is the deadline of the stream
  - $\beta^A = \alpha(\Delta - D)$ is service demand of α
  - β^G is the service guarantee
  - $\Rightarrow$ the event stream is schedulable iff

    $\beta(\Delta) \geq \beta^A = \alpha(\Delta - D), \qquad \forall \Delta \geq 0$

Analysis

- Delay and Backlog Analysis

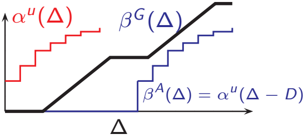
  Given:
  - ▷ α is the stream arrival curve
  - ▷ β is the service guarantee
  - ⇒ maximum delay D
  - ⇒ maximum backlog B



- Scheduling Analysis

  Suppose:
  - ▷ α is the stream arrival curve
  - ▷ D is the deadline of the stream
  - ▷ $\beta^A = \alpha(\Delta - D)$ is service demand of α
  - ▷ β^G is the service guarantee
  - ⇒ the event stream is schedulable iff

$$\beta(\Delta) \geq \beta^A = \alpha(\Delta - D), \qquad \forall \Delta \geq 0$$

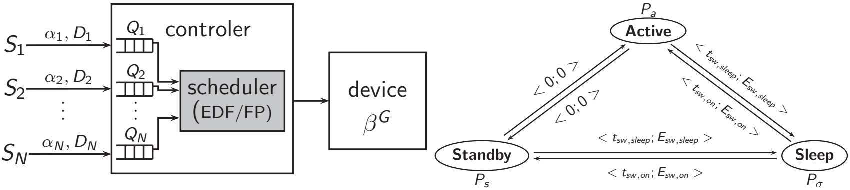


ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

Outline

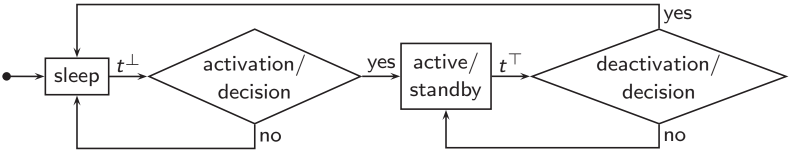
1. Introduction
2. Underlying Mathematical Model
3. Our Algorithms
4. Experimental Results



- System model
  - a device is managed by a controller
  - distributed backlogs to buffer events for each stream
  - events of different streams scheduled with EDF/FP policies
- Power model
  - *active* mode with power consumption P_a
  - *standby* mode with leakage power consumption P_s
  - *sleep* mode with power consumption P_δ
  - mode switch overhead: Break even time T_{BET}

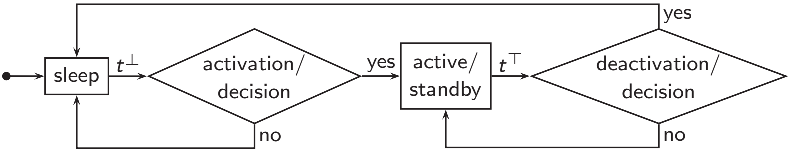
The Control Flow of Our Approach

- Control flow



The Control Flow of Our Approach

- Control flow

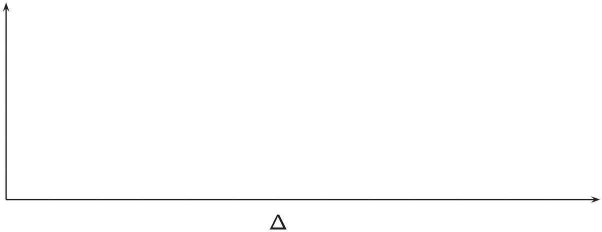


- Activation & Deactivation scheduling decisions
  - History Aware Deactivation (HAD) algorithm
  - Worst Case Greedy (WCG) activation algorithm
  - Event Driven Greedy (EDG) activation algorithm

Basic Routines

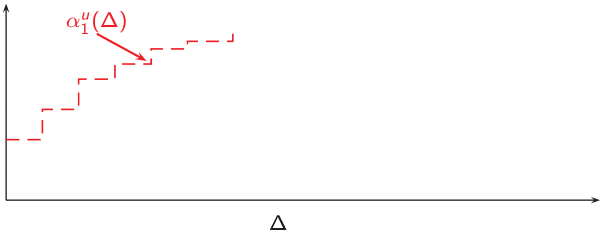
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

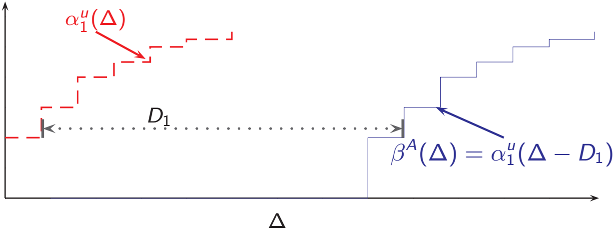


Basic Routines

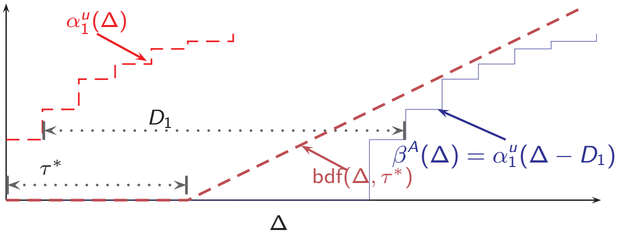
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



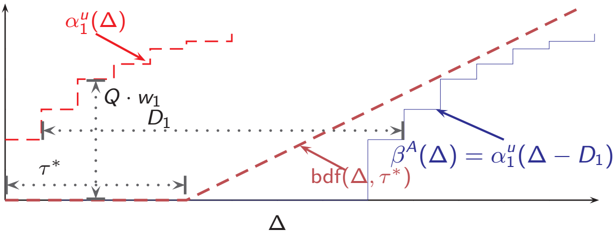
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

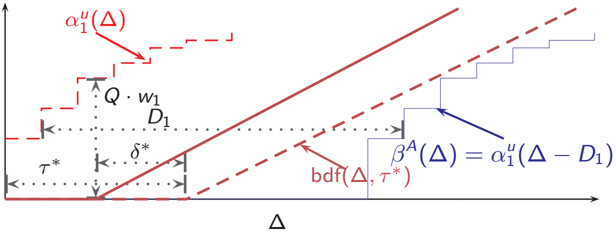


- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

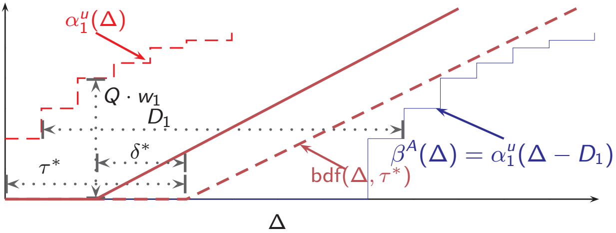


Basic Routines

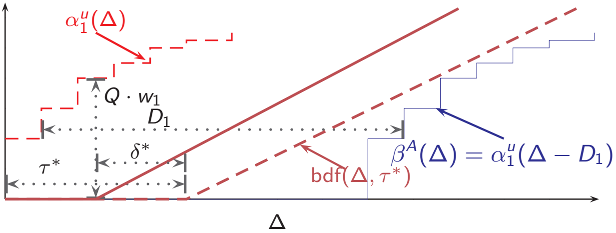
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

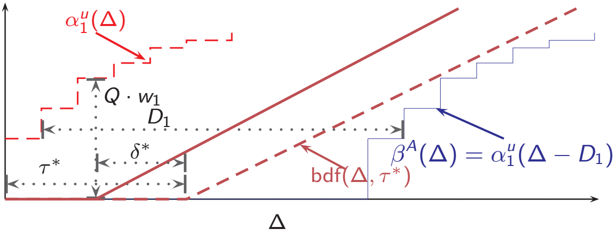


- Bounded delay function: **bdf**(Δ, τ)

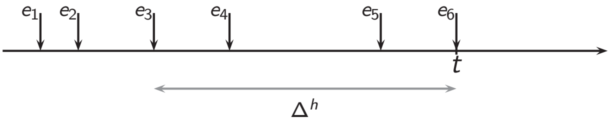


- History aware arrival curve: $\alpha^u(\Delta, t)$

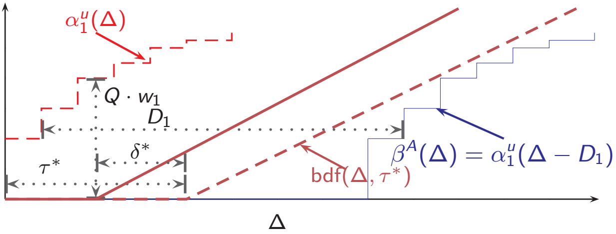
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



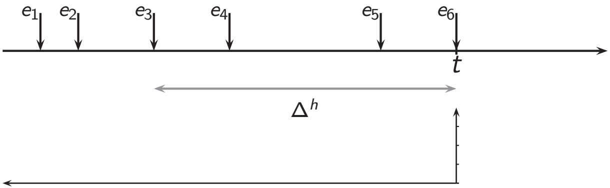
- History aware arrival curve: $\alpha^u(\Delta, t)$



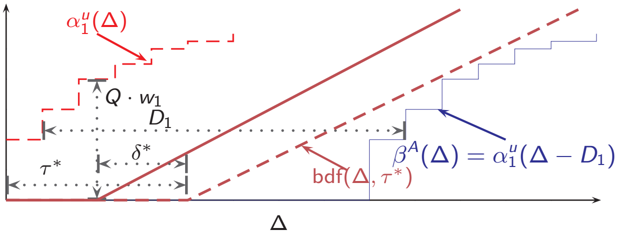
- Bounded delay function: $\textbf{bdf}(\Delta, \tau)$



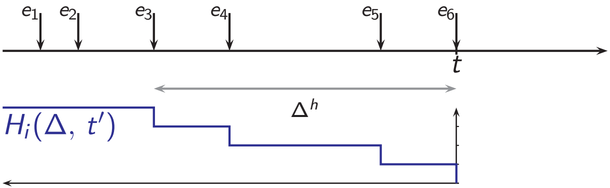
- History aware arrival curve: $\alpha^u(\Delta, t)$



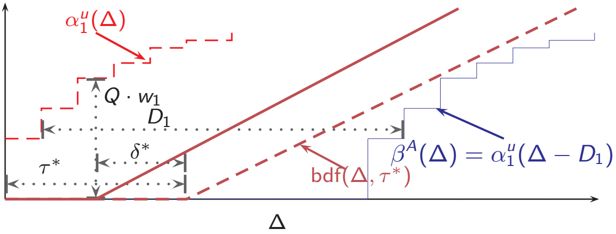
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



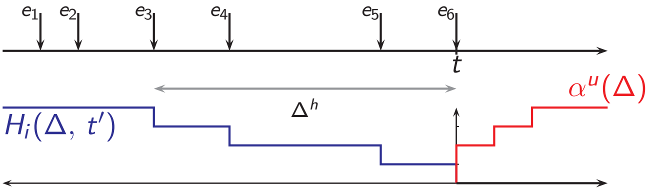
- History aware arrival curve: $\alpha^u(\Delta, t)$



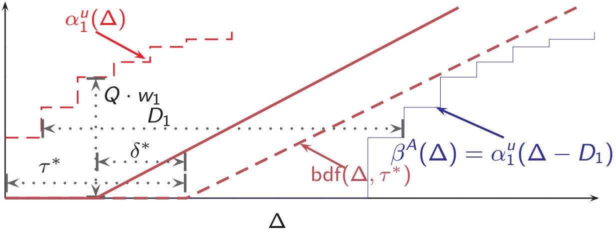
- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



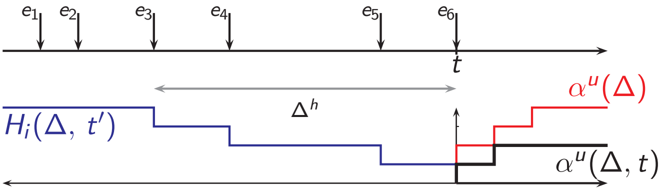
- History aware arrival curve: $\alpha^u(\Delta, t)$



- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$



- History aware arrival curve: $\alpha^u(\Delta, t)$



- Bounded delay function: $\mathbf{bdf}(\Delta, \tau)$

$$\mathbf{bdf}(\Delta, \tau) = \max\{0, (\Delta - \tau)\}, \forall \Delta \geq 0$$
$$\tau^* = \max\{\tau : \mathbf{bdf}(\Delta, \tau) \geq \beta^A(\Delta), \forall \Delta \geq 0\}$$
$$\delta^* = \max\left\{0, \min\{\delta : \alpha_i^u(\Delta) - \mathbf{bdf}(\Delta, \tau^* - \delta) \leq Q_i \cdot w_i, \forall \Delta\}\right\}$$

- History aware arrival curve: $\alpha^u(\Delta, t)$

$$H_i(\Delta, t') = \begin{cases} R_i(t') - R_i(t - \Delta), & \text{if } \Delta \leq \Delta^h; \\ R_i(t') - R_i(t' - \Delta^h), & \text{otherwise.} \end{cases}$$

$$\alpha_i^u(\Delta, t') \leq \inf_{\lambda \geq 0} \left\{\alpha_i^u(\Delta + \lambda) - H_i(\lambda, t')\right\}$$

Activation & Deactivation Algorithms

- History Aware Deactivation (HAD) algorithm
  - idea: turn off when the sleep interval can be larger than the break even time
- Worst Case Greedy (WCG) activation algorithm
  - idea: reevaluate when at the previous predication time
- Event Driven Greedy (EDG) activation algorithm
  - idea: reevaluate upon every event arrival
- Details refer to RTSS'09

Activation & Deactivation Algorithms

- History Aware Deactivation (HAD) algorithm
  - idea: turn off when the sleep interval can be larger than the break even time
- Worst Case Greedy (WCG) activation algorithm
  - idea: reevaluate when at the previous predication time
- Event Driven Greedy (EDG) activation algorithm
  - idea: reevaluate upon every event arrival
- Details refer to RTSS'09

⇒ Key: How to compute a *valid but tight* service demand β^A

Computing Flow for Multiple-Stream Scenario

history-aware future arrival: α_i^u

↓

backlogged-aware and backlog-constrained demands: $\beta_i^\flat, \beta_i^\dagger$

↓

individual stream service demand β_i^*

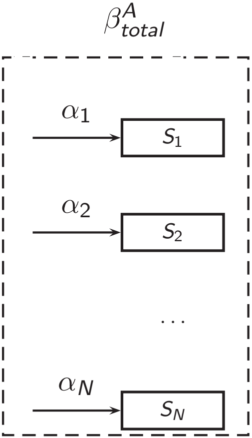
↓

total service demand β_{total}^A

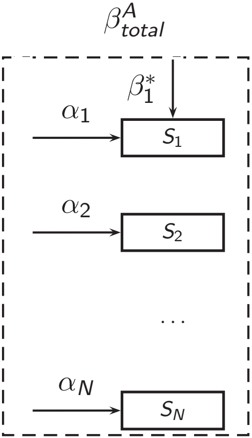
↓

bounded delay τ_{total} from β_{total}^A

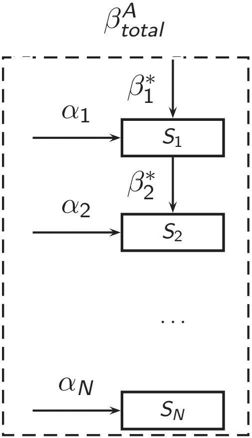
Preemptive Fixed-Priority Scheduling



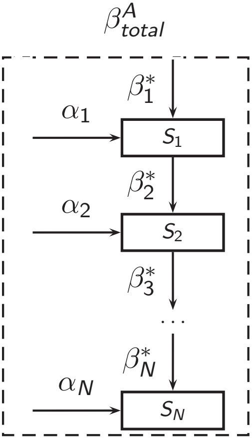
Preemptive Fixed-Priority Scheduling



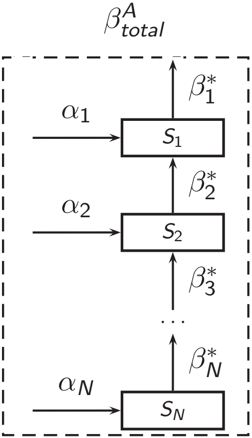
Preemptive Fixed-Priority Scheduling



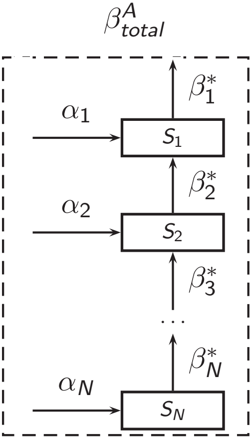
Preemptive Fixed-Priority Scheduling



Preemptive Fixed-Priority Scheduling

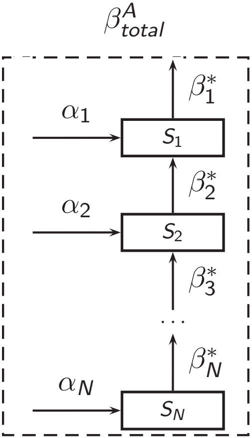


Preemptive Fixed-Priority Scheduling



$$\beta^*_N(\Delta, t') = \max\{\beta^\flat_N(\Delta, t'), \beta^\dagger_N(\Delta, t')\}$$

Preemptive Fixed-Priority Scheduling

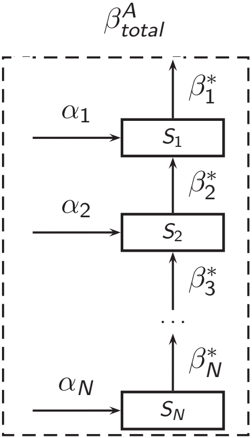


$$\beta^*_{k-1}(\Delta) = \max\left\{\beta^\sharp_{k-1}(\Delta),\, \beta^\flat_{k-1}(\Delta,\, t'),\, \beta^\dagger_{k-1}(\Delta,\, t')\right\}$$

$$\beta^\sharp_{k-1}(\Delta) \geq \inf\left\{\beta : \beta^*_k(\Delta,\, t') = \sup_{0\leq\lambda\leq\Delta}\left\{\beta(\lambda) - \alpha^u_{k-1}(\lambda,\, t')\right\}\right\}$$

$$\beta^*_N(\Delta,\, t') = \max\{\beta^\flat_N(\Delta,\, t'),\, \beta^\dagger_N(\Delta,\, t')\}$$

Preemptive Fixed-Priority Scheduling



$$\beta^A_{total}(\Delta) = \beta^*_1(\Delta)$$

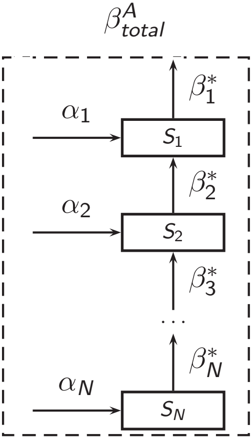
$$\beta^*_{k-1}(\Delta) = \max\left\{\beta^\sharp_{k-1}(\Delta),\, \beta^\flat_{k-1}(\Delta,\, t'),\, \beta^\dagger_{k-1}(\Delta,\, t')\right\}$$

$$\beta^\sharp_{k-1}(\Delta) \geq \inf\left\{\beta : \beta^*_k(\Delta,\, t') = \sup_{0\leq\lambda\leq\Delta}\{\beta(\lambda) - \alpha^u_{k-1}(\lambda,\, t')\}\right\}$$

$$\beta^*_N(\Delta,\, t') = \max\{\beta^\flat_N(\Delta,\, t'),\, \beta^\dagger_N(\Delta,\, t')\}$$

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento



$$\beta^A_{total}(\Delta) = \beta^*_1(\Delta)$$

$$\beta^*_{k-1}(\Delta) = \max\left\{\beta^\sharp_{k-1}(\Delta),\ \beta^\flat_{k-1}(\Delta,\ t'),\ \beta^\dagger_{k-1}(\Delta,\ t')\right\}$$

$$\beta^\sharp_{k-1}(\Delta) \geq \inf\left\{\beta : \beta^*_k(\Delta,\ t') = \sup_{0\leq\lambda\leq\Delta}\{\beta(\lambda) - \alpha^u_{k-1}(\lambda,\ t')\}\right\}$$

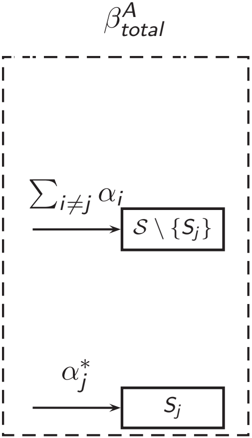
$$\beta^*_N(\Delta,\ t') = \max\{\beta^\flat_N(\Delta,\ t'),\ \beta^\dagger_N(\Delta,\ t')\}$$

$$\tau_{total} = \max\left\{\tau : \mathbf{bdf}(\Delta, \tau) \geq \beta^*_1(\Delta),\ \forall\Delta \geq 0\right\}$$

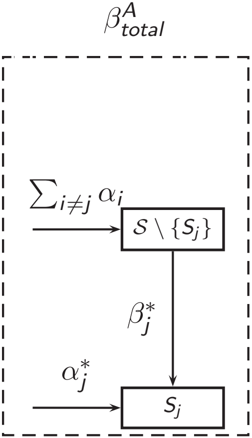
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

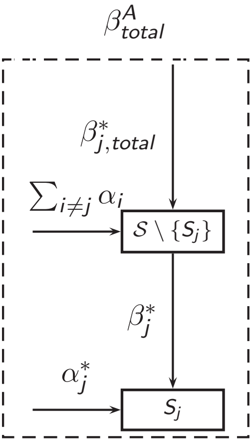
Preemptive Earliest-Deadline-First Scheduling

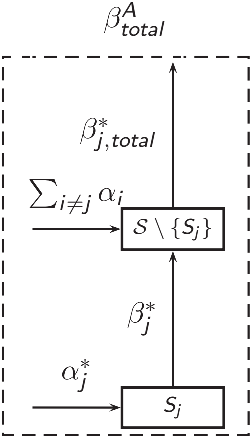


Preemptive Earliest-Deadline-First Scheduling

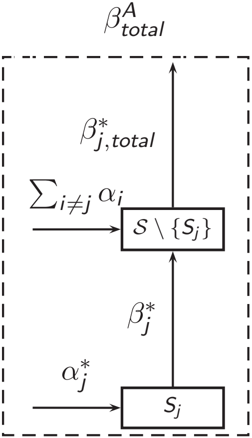


Preemptive Earliest-Deadline-First Scheduling

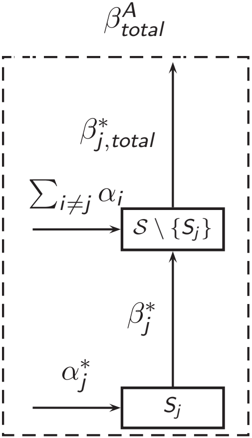



β_{total}^{A}
$\beta_{j,total}^{*}$
$\sum_{i \neq j} \alpha_i$
$\mathcal{S} \setminus \{S_j\}$
β_j^*
α_j^*
S_j

Preemptive Earliest-Deadline-First Scheduling



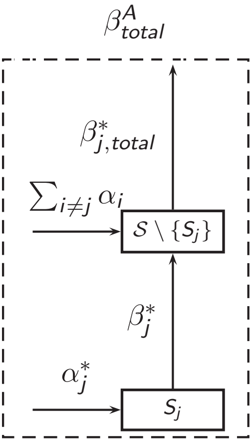
$$\beta^*_j(\Delta,\, t') = \max\{\beta^\flat_j(\Delta,\, t'),\, \beta^\dagger_j(\Delta,\, t')\}$$



$$\beta^{*}_{j,total}(\Delta) = \max\left\{\beta^{\sharp}_{j}(\Delta), \sum_{i \neq j}^{N} \beta^{\flat}_{i}(\Delta, t')\right\}$$

$$\beta^{\sharp}_{j}(\Delta) \geq \inf\left\{\beta : \beta^{*}_{j}(\Delta, t') = \sup_{0 \leq \lambda \leq \Delta}\{\beta(\lambda) - \sum_{i \neq j}^{N} \alpha^{u}_{i}(\lambda, t')\}\right\}$$

$$\beta^{*}_{j}(\Delta, t') = \max\{\beta^{\flat}_{j}(\Delta, t'), \beta^{\dagger}_{j}(\Delta, t')\}$$

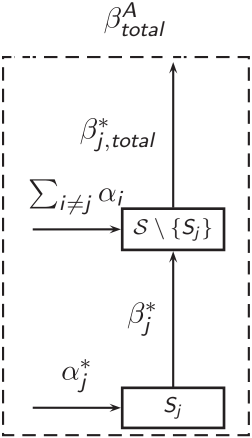


$$\beta^A_{total}(\Delta) = \max_{i\in N}\{\beta^*_{i,total}(\Delta)\}$$

$$\beta^*_{j,total}(\Delta) = \max\left\{\beta^\sharp_j(\Delta), \sum_{i\neq j}^{N}\beta^\flat_i(\Delta, t')\right\}$$

$$\beta^\sharp_j(\Delta) \geq \inf\left\{\beta : \beta^*_j(\Delta, t') = \sup_{0\leq\lambda\leq\Delta}\{\beta(\lambda) - \sum_{i\neq j}^{N}\alpha^u_i(\lambda, t')\}\right\}$$

$$\beta^*_j(\Delta, t') = \max\{\beta^\flat_j(\Delta, t'), \beta^\dagger_j(\Delta, t')\}$$



$$\beta^A_{total}(\Delta) = \max_{i\in N}\{\beta^*_{i,total}(\Delta)\}$$

$$\beta^*_{j,total}(\Delta) = \max\left\{\beta^\sharp_j(\Delta), \sum_{i\neq j}^{N}\beta^\flat_i(\Delta,\, t')\right\}$$

$$\beta^\sharp_j(\Delta) \geq \inf\left\{\beta : \beta^*_j(\Delta,\, t') = \sup_{0\leq\lambda\leq\Delta}\{\beta(\lambda) - \sum_{i\neq j}^{N}\alpha^u_i(\lambda,\, t')\}\right\}$$

$$\beta^*_j(\Delta,\, t') = \max\{\beta^\flat_j(\Delta,\, t'),\, \beta^\dagger_j(\Delta,\, t')\}$$

$$\tau_{total} = \max\Big\{\tau : \mathbf{bdf}(\Delta,\tau) \geq \max_{i\in N}\{\beta^*_{i,total}(\Delta)\},\ \forall\Delta\geq 0\Big\}$$

Outline

1. Introduction
2. Underlying Mathematical Model
3. Our Algorithms
4. Experimental Results

Experiment Setup

- Event Stream Setting

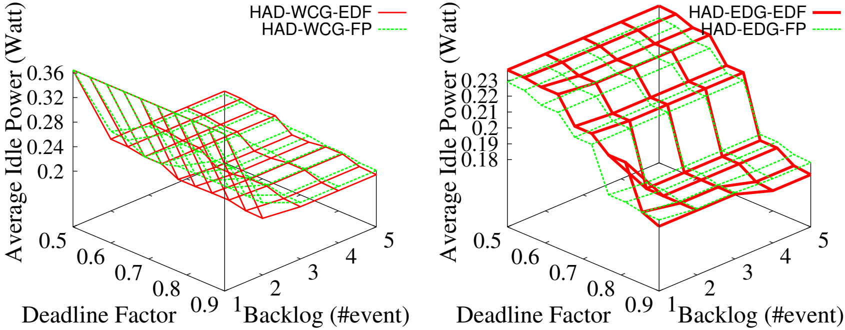
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
p (msec)	198	102	283	354	239	194	148	114	313	119
j (msec)	387	70	269	387	222	260	91	13	302	187
d (msec)	48	45	58	17	65	32	78	-	86	89
c (msec)	12	7	7	11	8	5	13	14	5	6

- Power Profiles for the Device

Device Name	P_a (Watt)	P_s (Watt)	P_σ (Watt)	t_{sw} (sec)	E_{sw} (mJ)
IBM Microdrive	1.3	0.5	0.1	0.012	9.6

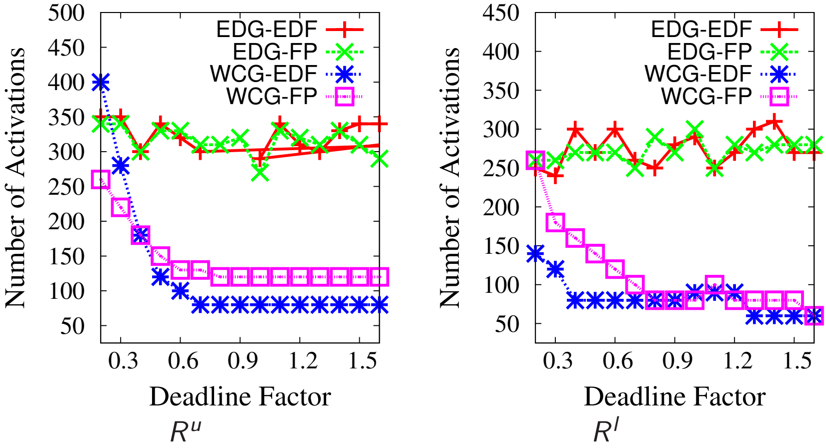
- Schemes to compare: HAD-EDG & HAD-WCG
- Bursting and sparse traces: R^u and R^l
- Implemented using RTC ToolBox
- Simulated on 1.7 GHz processor

Average Idle Power Consumption (Watt)



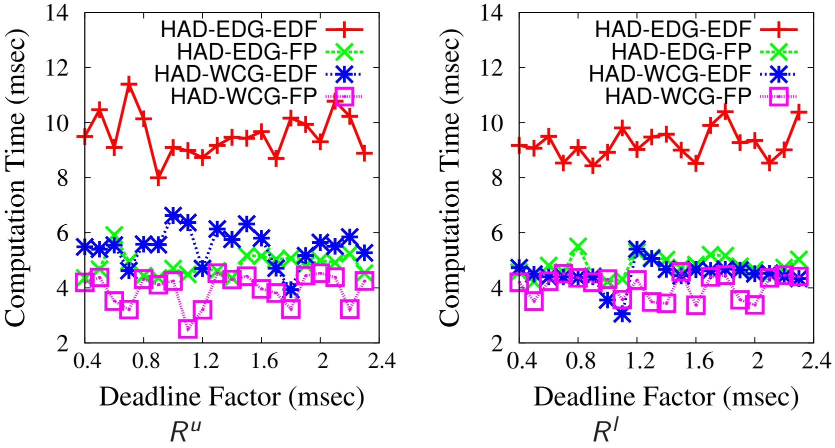
R^u R^l

- Idle power is reduced for both traces R^u and R^l



- EDG activation is varied according to the traces
- WCG activation is affected by the deadline

Computation Expense



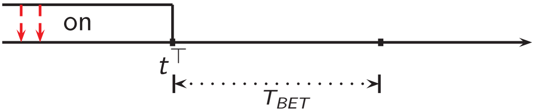
- Both schemes require a small computation time
- The increment for longer relative deadline is small

Conclusion

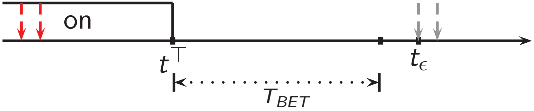
- Extend online algorithms which adaptively control the on/off of a device for multiple event streams with
  - preemptive fixed-priority scheduling
  - preemptive earliest-deadline-first scheduling
- Guarantee hard real-time requirements with respect to both timing and backlog constraints
- Experiments prove the effectiveness of the algorithms

Deactivation Algorithm (HAD)

The principle is to deactivate the device only when energy saving is possible.

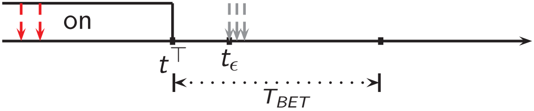


The principle is to deactivate the device only when energy saving is possible.



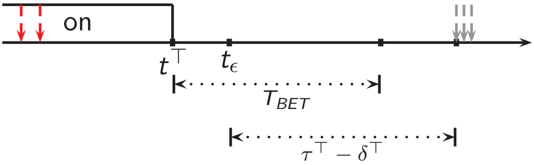
Deactivation Algorithm (HAD)

The principle is to deactivate the device only when energy saving is possible.



Deactivation Algorithm (HAD)

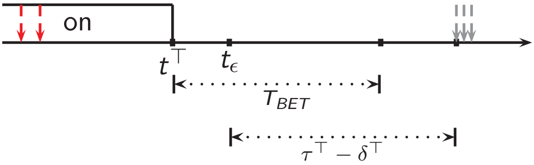
The principle is to deactivate the device only when energy saving is possible.



Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

The principle is to deactivate the device only when energy saving is possible.

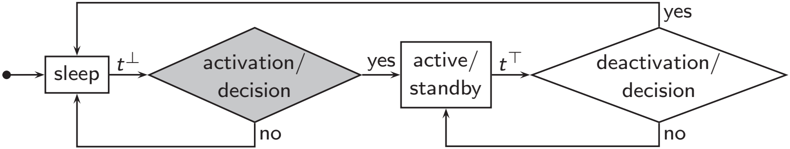


$$t_\epsilon \leftarrow \min_{t > t^\top} t \text{ such that } \bar{\alpha}_1^u(t - t^\top, t^\top) > 0$$

$$\tau^\top = \max\{\tau : \mathbf{bdf}(\Delta, \tau) \geq \alpha_1^u(\Delta - D_1, t_\epsilon)\}$$

$$\delta^\top = \max\left\{0, \min\{\delta : \alpha_1^u(\Delta, t_\epsilon) - \mathbf{bdf}(\Delta, \tau^\top - \delta) \leq Q \cdot w_1, \forall \Delta\}\right\}$$

Activation Algorithms

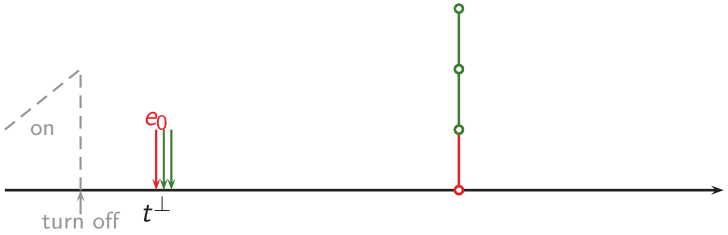


- Worst Case Greedy (WCG) Algorithm
  - Time triggered reevaluation
  - Suitable for bursty event arrival
- Event Driven Greedy (EDG) Algorithm
  - Event triggered reevaluation
  - Suitable for sparse event arrival

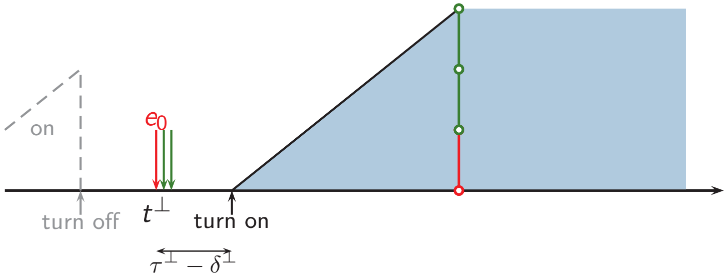
Worst Case Greedy Algorithm for Activation

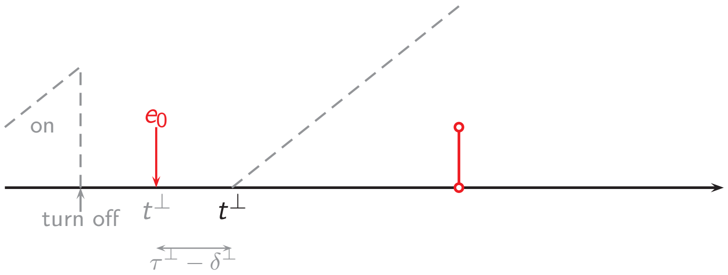
Worst Case Greedy Algorithm for Activation

Worst Case Greedy Algorithm for Activation

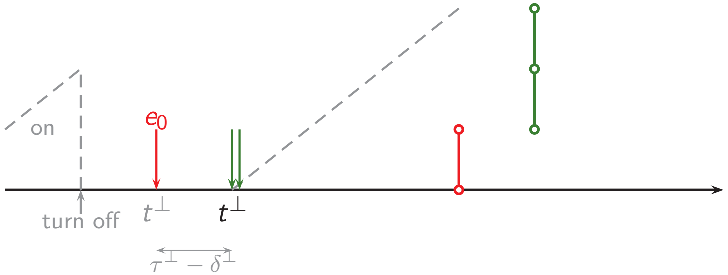


Worst Case Greedy Algorithm for Activation

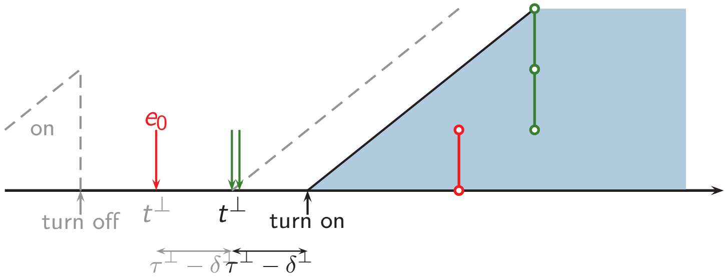



on
e_0
turn off
$t^\perp$
$t^\perp$
$\tau^\perp - \delta^\perp$

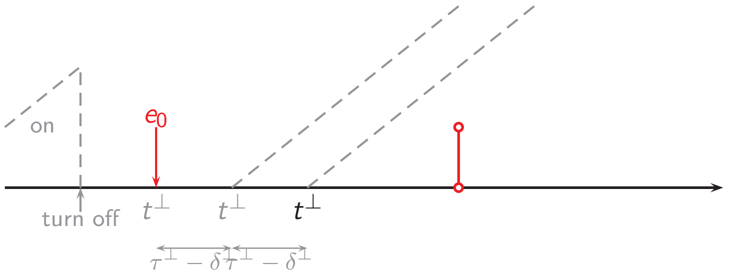
Worst Case Greedy Algorithm for Activation



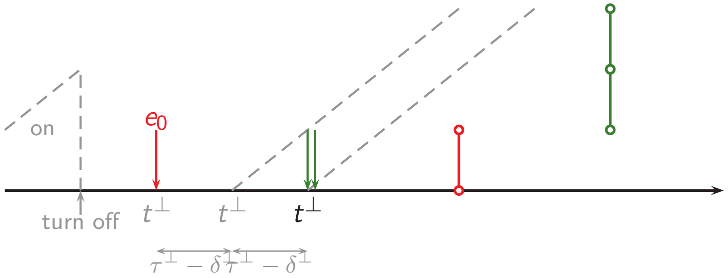
Worst Case Greedy Algorithm for Activation



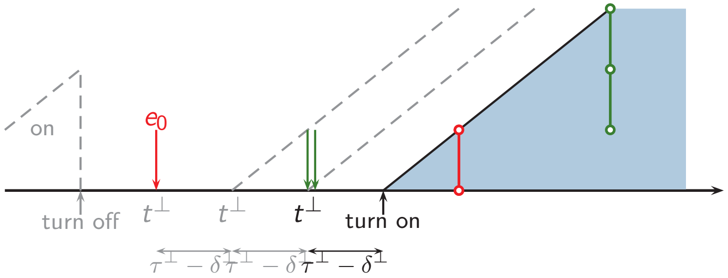
Worst Case Greedy Algorithm for Activation

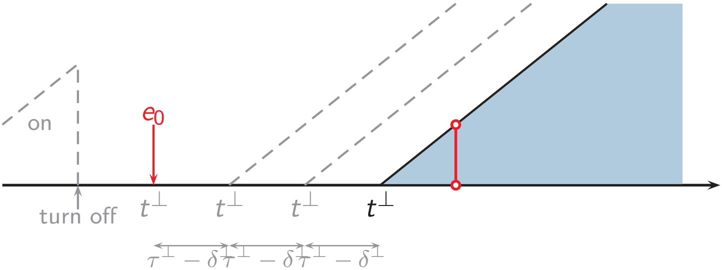


Worst Case Greedy Algorithm for Activation

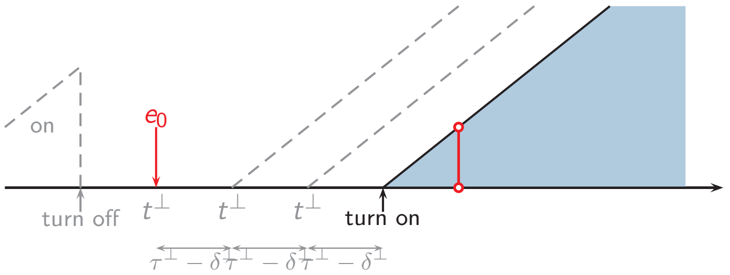


Worst Case Greedy Algorithm for Activation

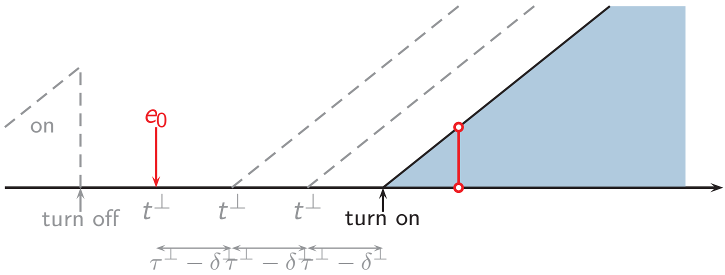



on
e_0
turn off
$t^{\perp}$
$t^{\perp}$
$t^{\perp}$
$t^{\perp}$
$\tau^{\perp} - \delta^{\perp}$
$\tau^{\perp} - \delta^{\perp}$
$\tau^{\perp} - \delta^{\perp}$

Worst Case Greedy Algorithm for Activation



Worst Case Greedy Algorithm for Activation

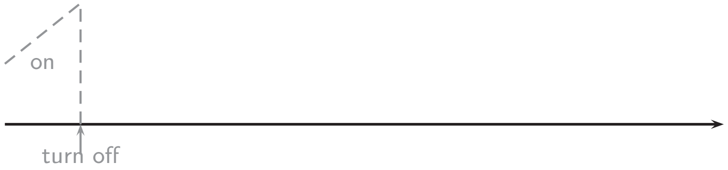


$$\beta^A(\Delta) = \alpha_1^u(\Delta - D_1,\ t^\perp) + w_1 \cdot B_1(\Delta,\ t^\perp)$$

$$\tau^\perp = \max\{\tau : \mathbf{bdf}(\Delta, \tau) \geq \beta^A(\Delta)\}$$

$$\delta^\perp = \max\left\{0,\ \min\{\delta : \alpha_1^u(\Delta, t^\perp) - \mathbf{bdf}(\Delta, \tau^\perp - \delta) \leq \left(Q - |\mathbf{E}(t^\perp)|\right) \cdot w_1,\ \forall\Delta\}\right\}$$

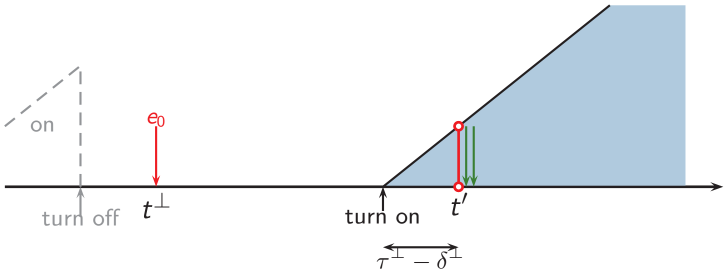
Event Driven Greedy Algorithm for Activation



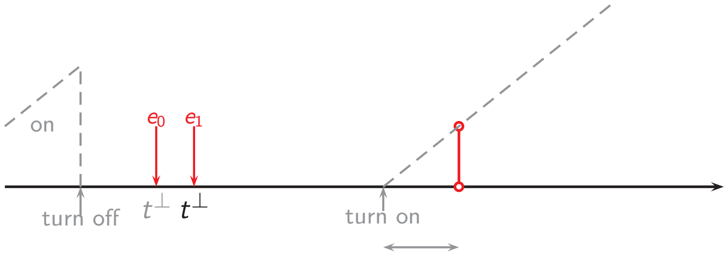
Event Driven Greedy Algorithm for Activation

Event Driven Greedy Algorithm for Activation

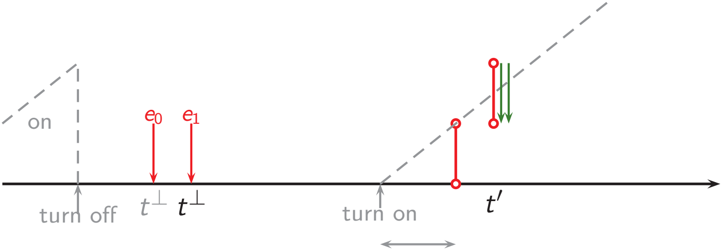
Event Driven Greedy Algorithm for Activation



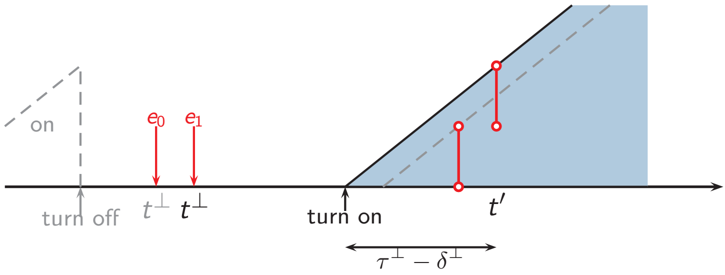
Event Driven Greedy Algorithm for Activation



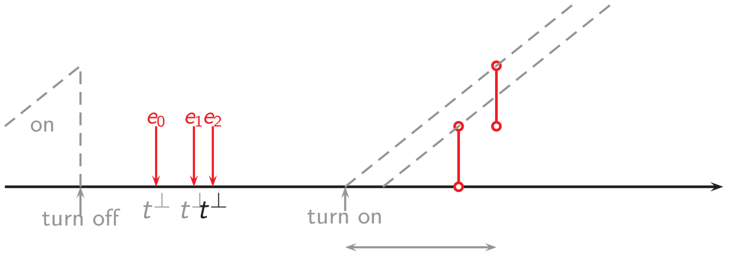
Event Driven Greedy Algorithm for Activation



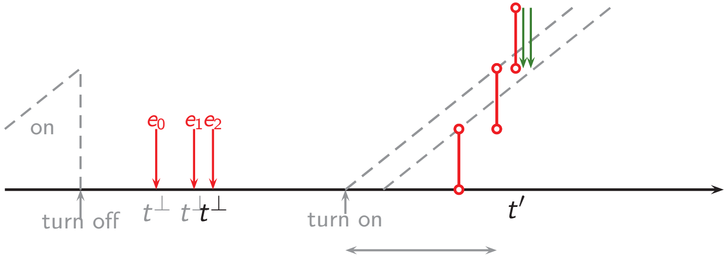
Event Driven Greedy Algorithm for Activation



Event Driven Greedy Algorithm for Activation



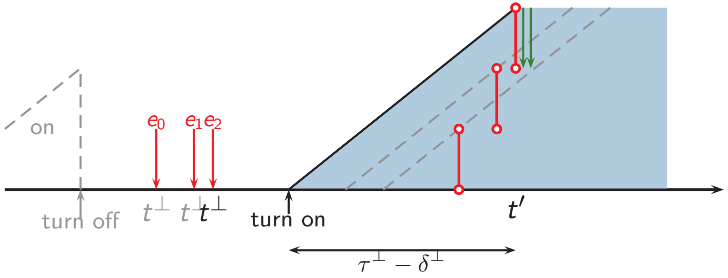
Event Driven Greedy Algorithm for Activation



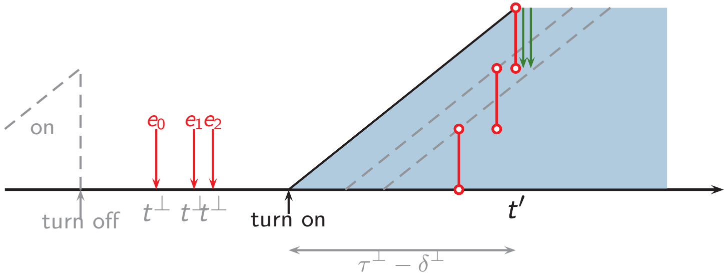
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Scuola Superiore Sant'Anna
di Studi Universitari e di Perfezionamento

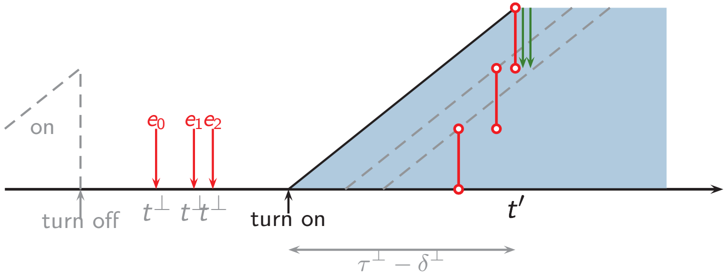
Event Driven Greedy Algorithm for Activation



Event Driven Greedy Algorithm for Activation



Event Driven Greedy Algorithm for Activation



$$\beta^A(\Delta) = \alpha_1^u(\Delta - D_1,\ t') + w_1 \cdot B_1'(\Delta,\ t')$$

$$\tau^\perp = \max\{\tau : \mathbf{bdf}(\Delta, \tau) \geq \beta^A(\Delta)\}$$

$$\delta^\perp = \max\Big\{0,\ \min\{\delta : \alpha_1^u(\Delta, t') - \mathbf{bdf}(\Delta, \tau^\perp - \delta) \leq \left(Q - |\mathbf{E}(t^\perp)| - \bar{\alpha}_1^l(\epsilon)\right) \cdot w_1,\ \forall\Delta\}\Big\}$$