ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Modular Performance Analysis with Real-Time Calculus

Wolfgang Haid, Simon Perathoner, Nikolay Stoimenov, Lothar Thiele

ARTIST2 PhD Course on Automated Formal Methods for Embedded Systems
DTU - Lyngby, Denmark - June 11, 2007

Presentation overview

1 Introduction to System Level Performance Analysis

(Simon Perathoner)

2 Modular Performance Analysis (MPA)

(Nikolay Stoimenov)

3 Real-Time Calculus (RTC)

(Wolfgang Haid)

4 Extensions to basic model

(Wolfgang Haid)

5 Real-Time Interfaces (RTI)

(Nikolay Stoimenov)

6 Comparison with other approaches

(Simon Perathoner)

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Modular Performance Analysis with Real-Time Calculus

1. Introduction to System Level Performance Analysis

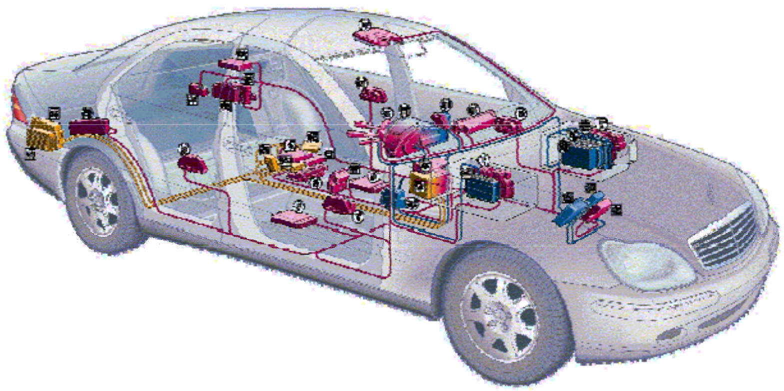
Simon Perathoner

ARTIST2 PhD Course on Automated Formal Methods for Embedded Systems
DTU - Lyngby, Denmark - June 11, 2007

Embedded Real-Time Systems

- Special-purpose information processing systems
- Embedded into larger products
- Must meet real-time constraints

Trends in Embedded System Design

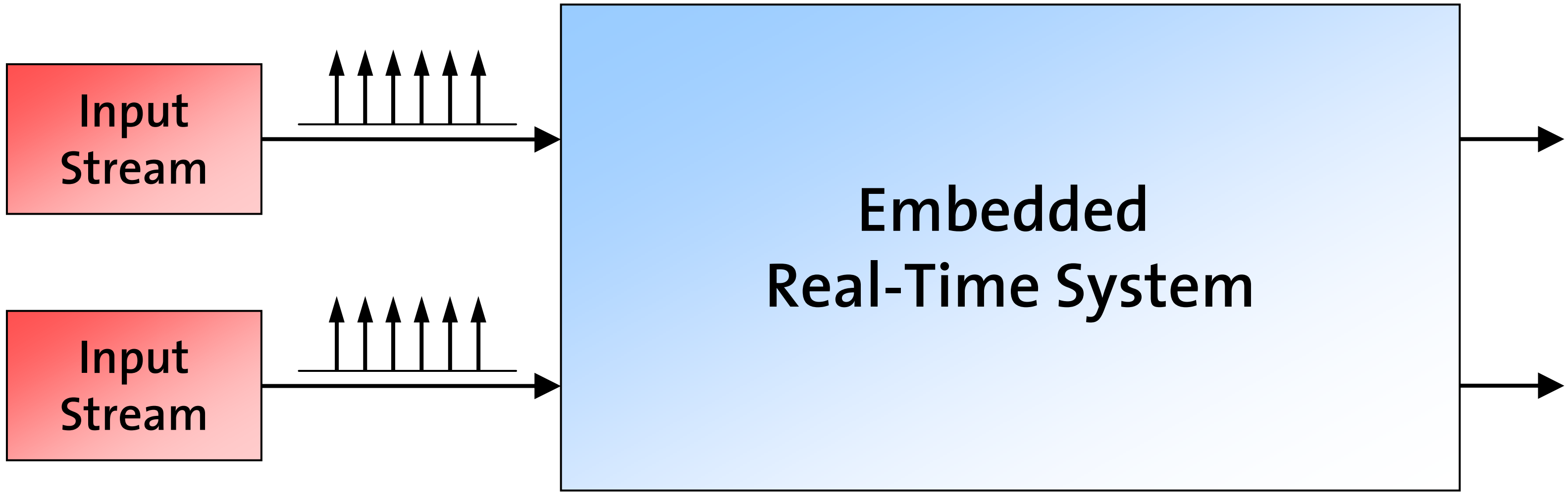


Architectures are increasingly:

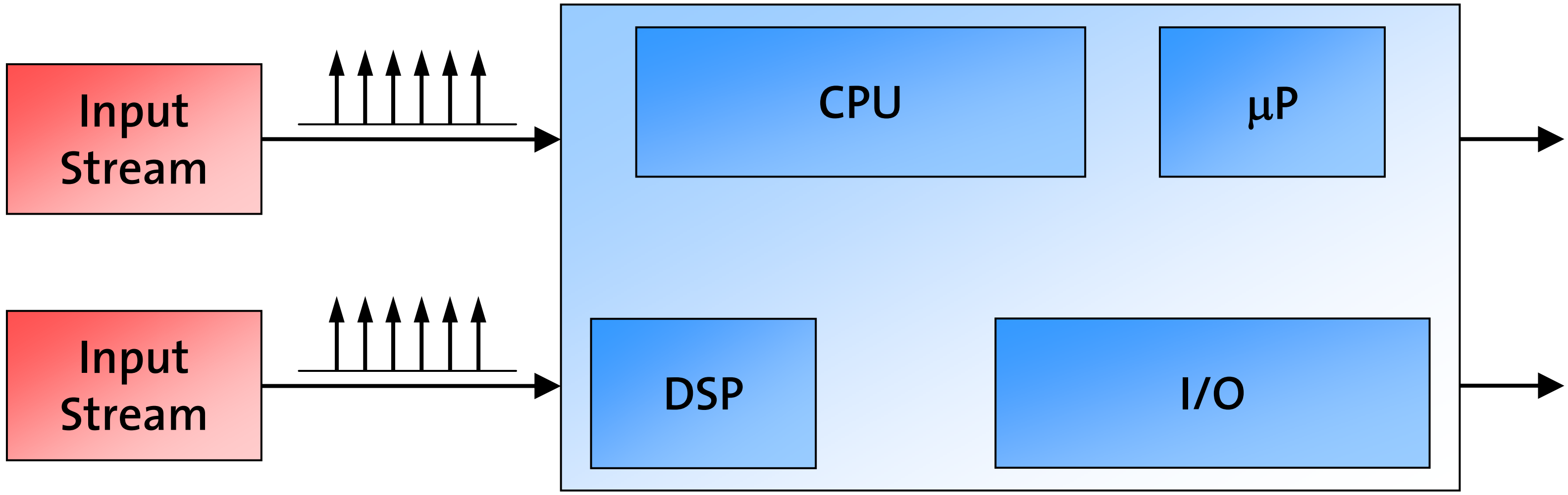
- parallel
- distributed
- heterogeneous

⇨ Analysis and prediction of system behavior is complex!

System Level Performance Analysis

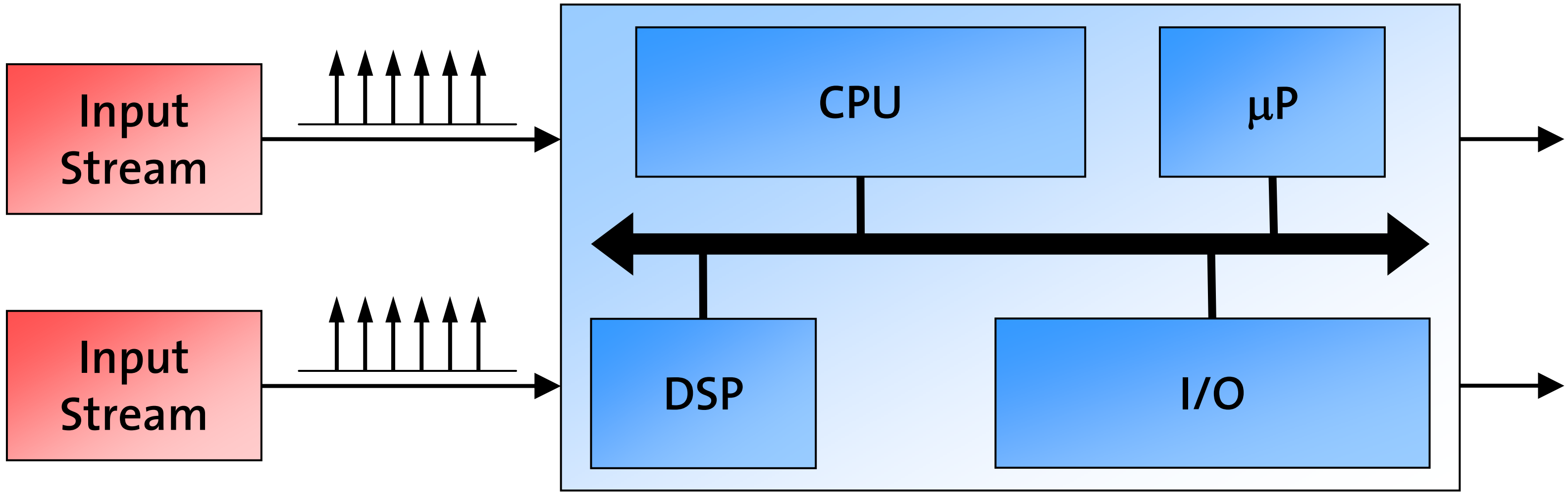


System level performance Analysis



Computational Resources ...

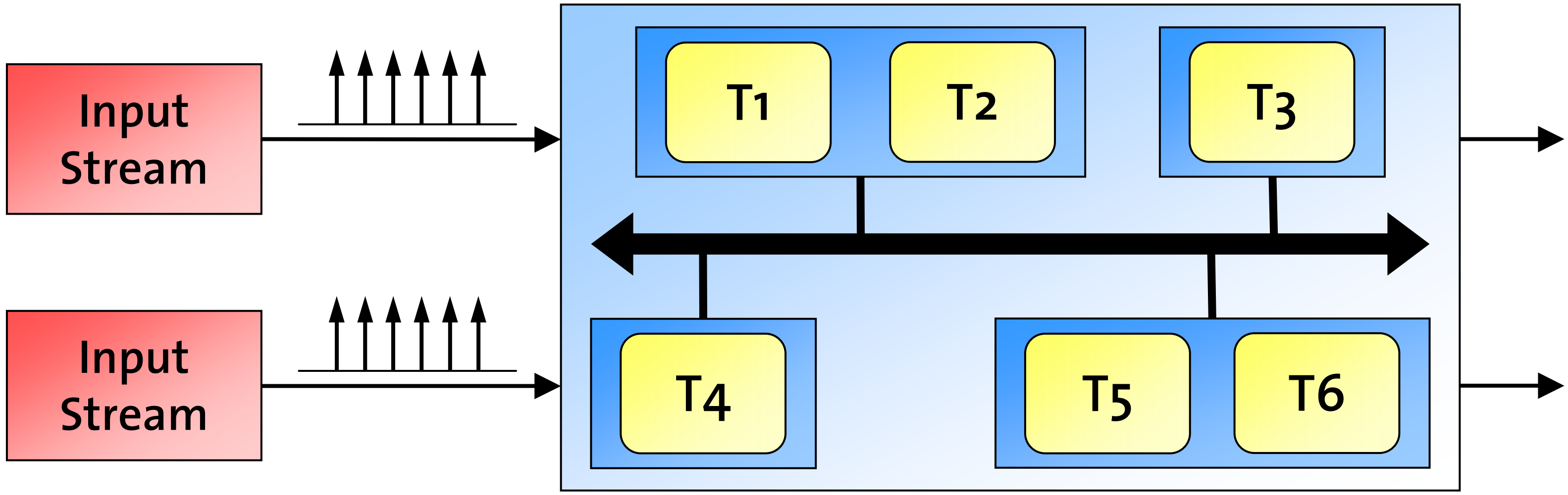
System Level Performance Analysis



Computational Resources ...

... Communication Resources ...

System Level Performance Analysis

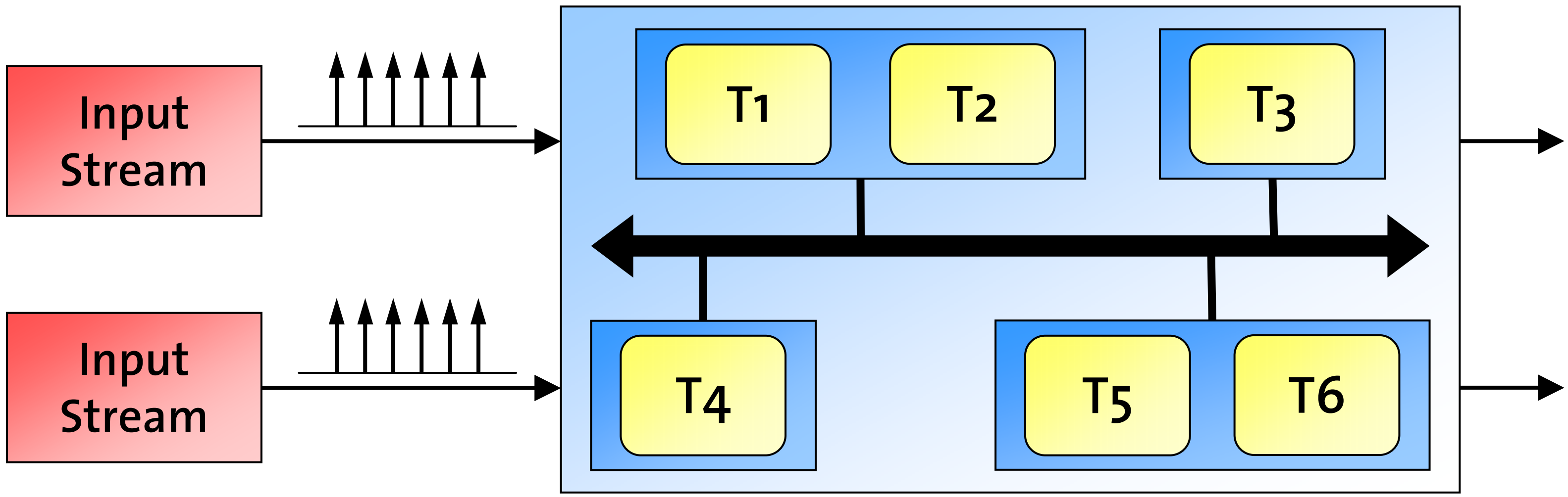


Computational Resources ...

... Communication Resources ...

... Tasks (HW/SW Components)

System Level Performance Analysis



Memory Requirements?

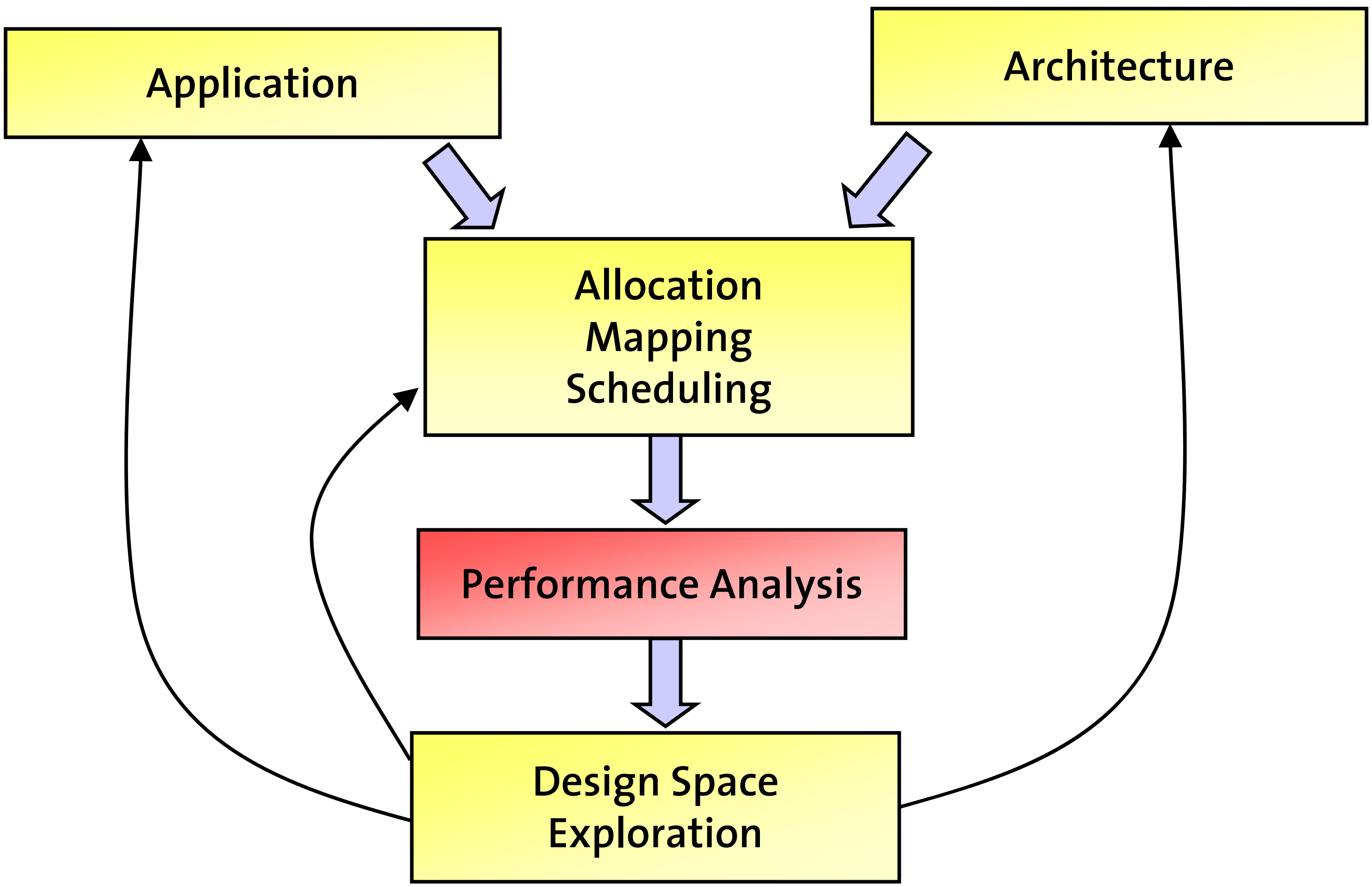
Timing Properties?

Bottleneck?

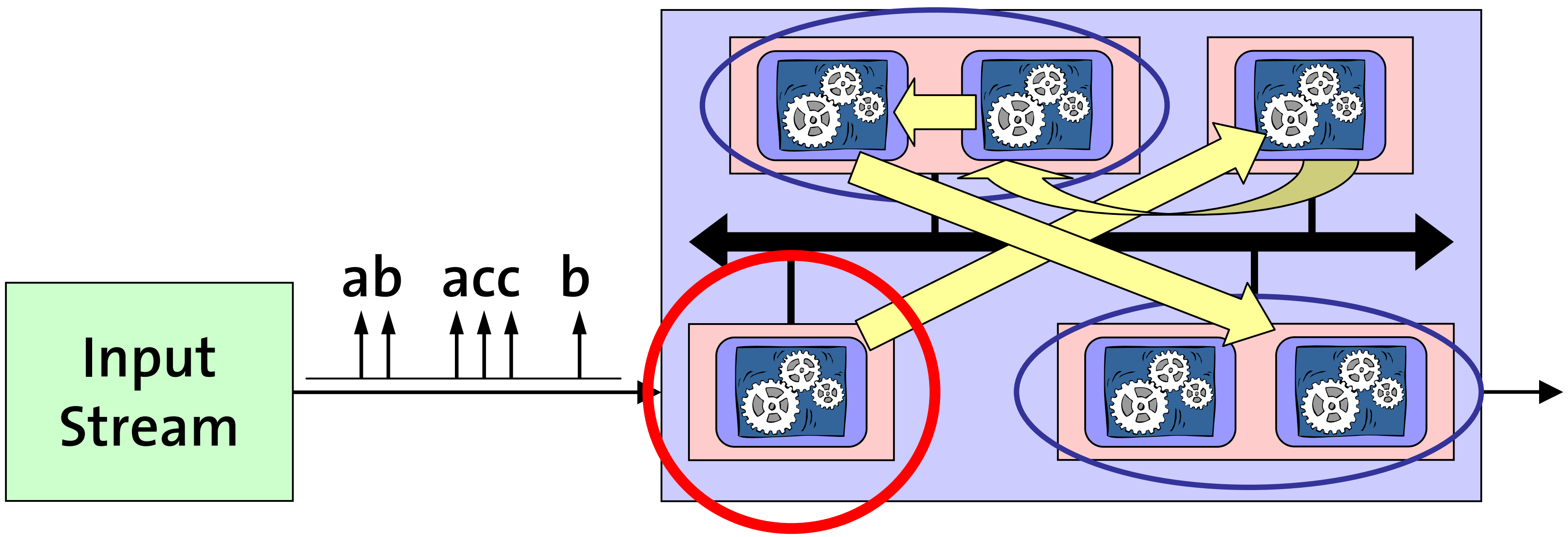
Processor Speeds?

Bus Utilization?

Role in the design process



Challenges of Performance Analysis



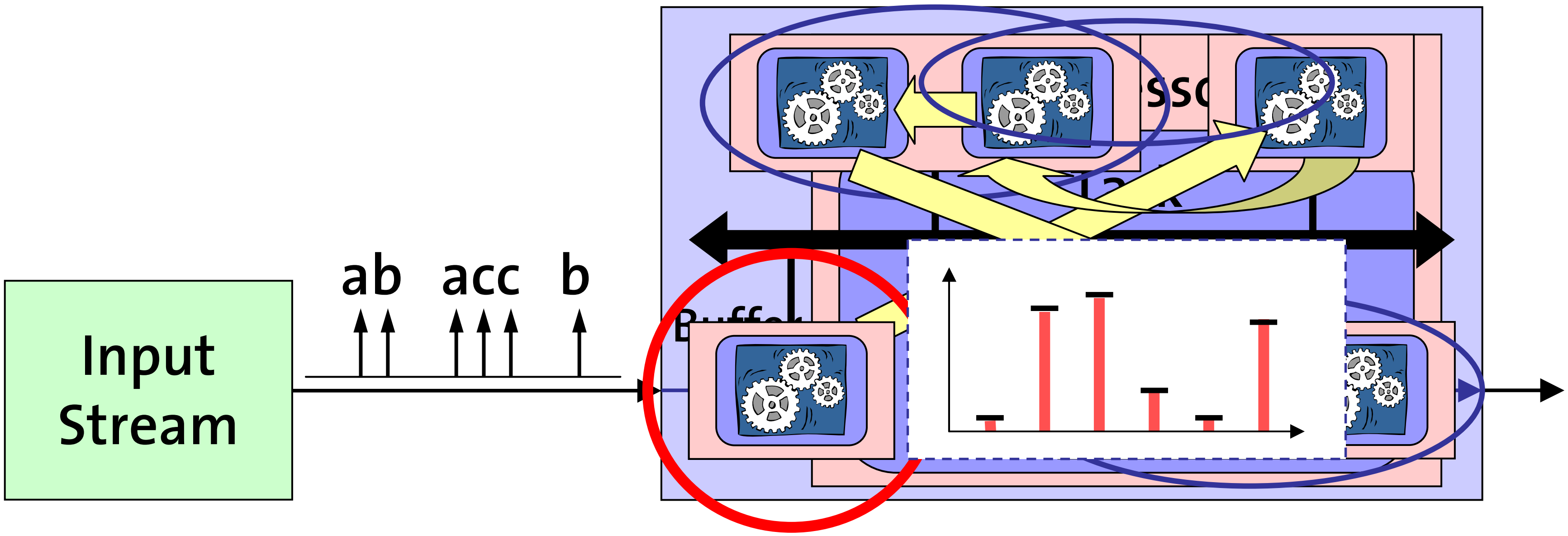
Task Communication

Resource sharing (Scheduling)

Complex Input:

- Timing (jitter, bursts, ...)
- Different Event Types

Challenges of Performance Analysis



Task Communication

Resource sharing (Scheduling)

Complex Input:

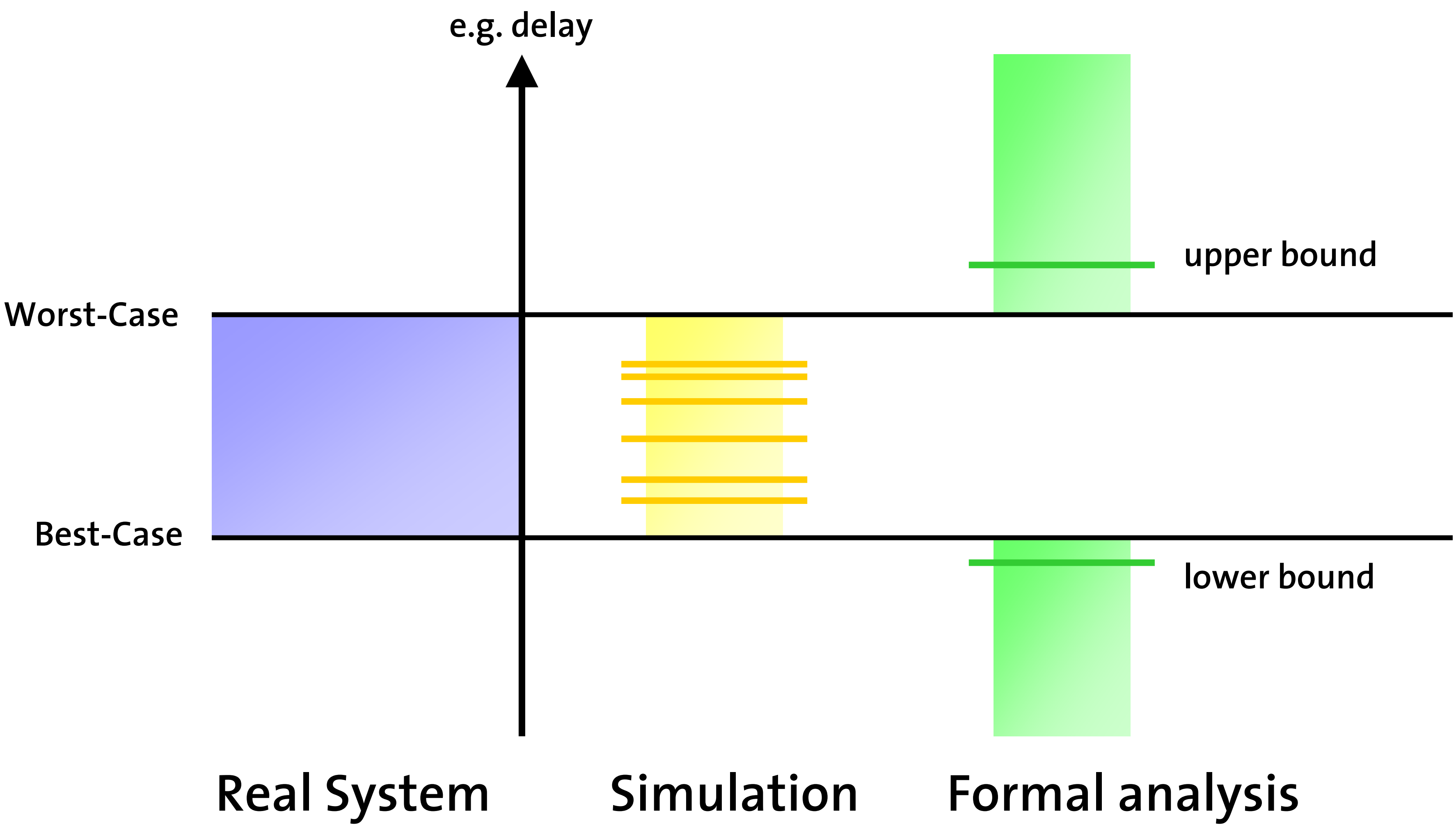
- Timing (jitter, bursts, ...)
- Different Event Types

Variable Resource Availability

Variable Execution Demand

- Input (different event types)
- Internal State (Program, Cache, ...)

Formal Analysis vs. Simulation



Requirements for a formal PA method

- Correctness
- Accuracy
- Embedding into the design process
- Modularity
- Short analysis time

Modular Performance Analysis
- Models, Methods and Scenarios -

© Nikolay Stoimenov

ETH Zurich, Switzerland

Outline

- **Modular Performance Analysis**
- MPA Case Study

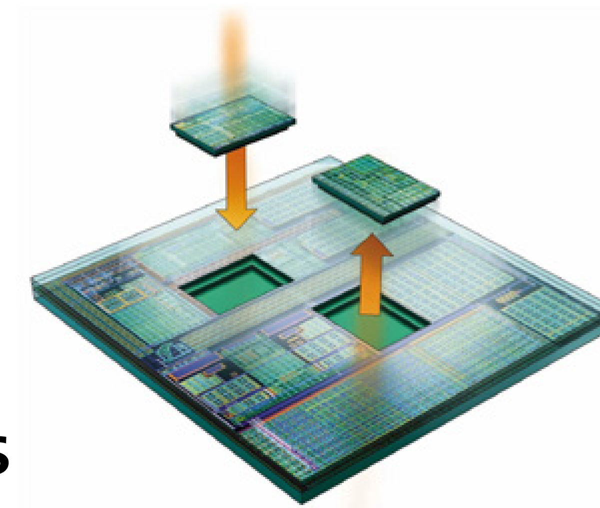
Analysis and Design

Embedded System =

Computation + Resource Interaction

Analysis:

Infer system properties from subsystem properties.



Design:

Build a system from subsystems while meeting requirements.

Challenges

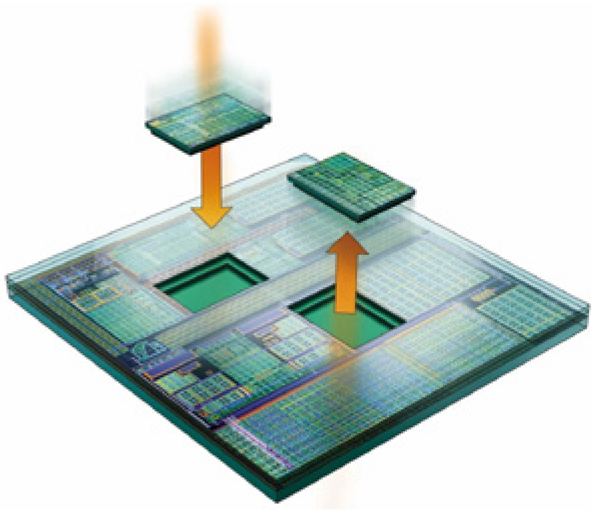
Make Analysis and Synthesis Compositional

Stepwise Refinement:

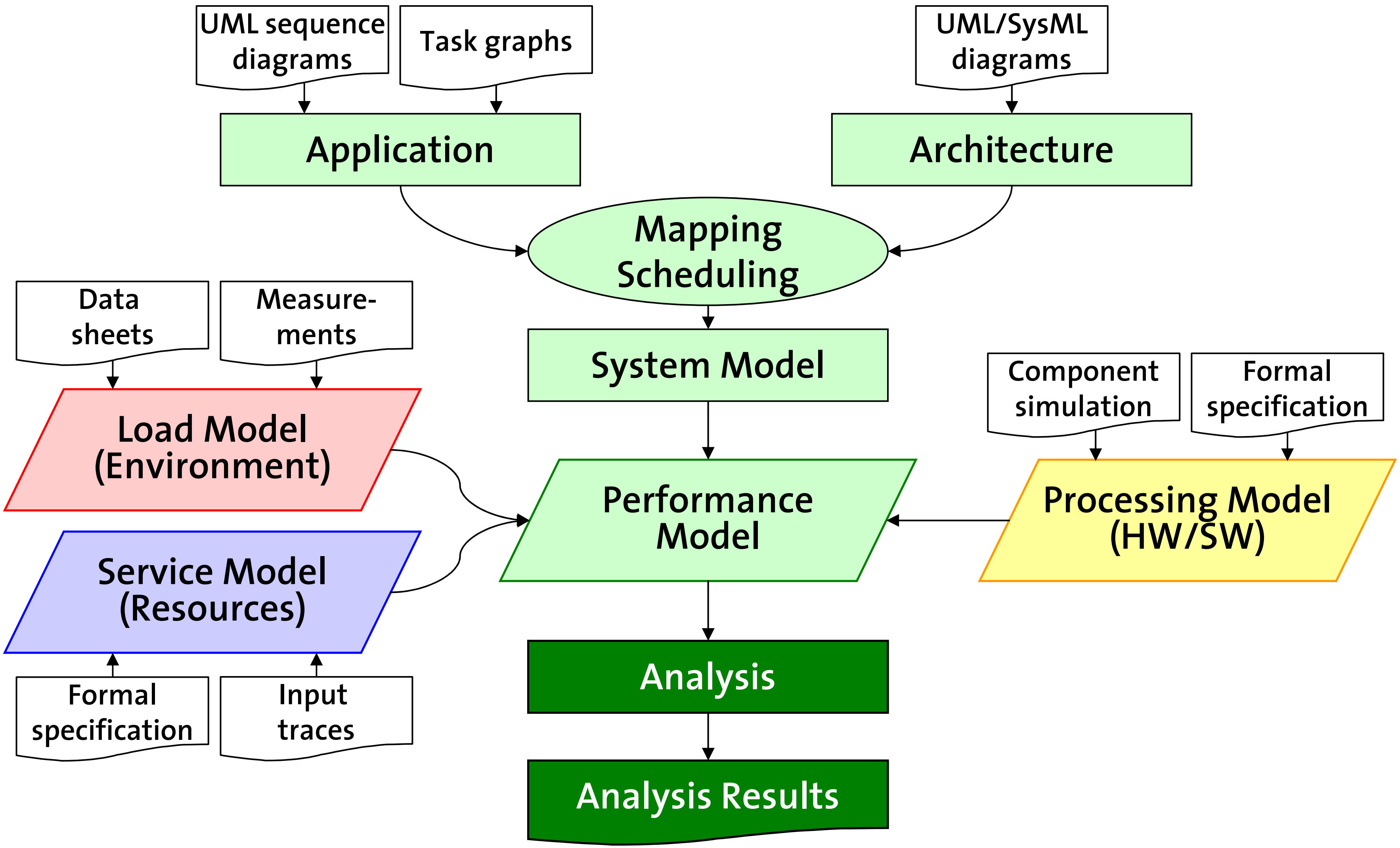
a. compose subsystems
b. refine subsystems

Adaptivity:

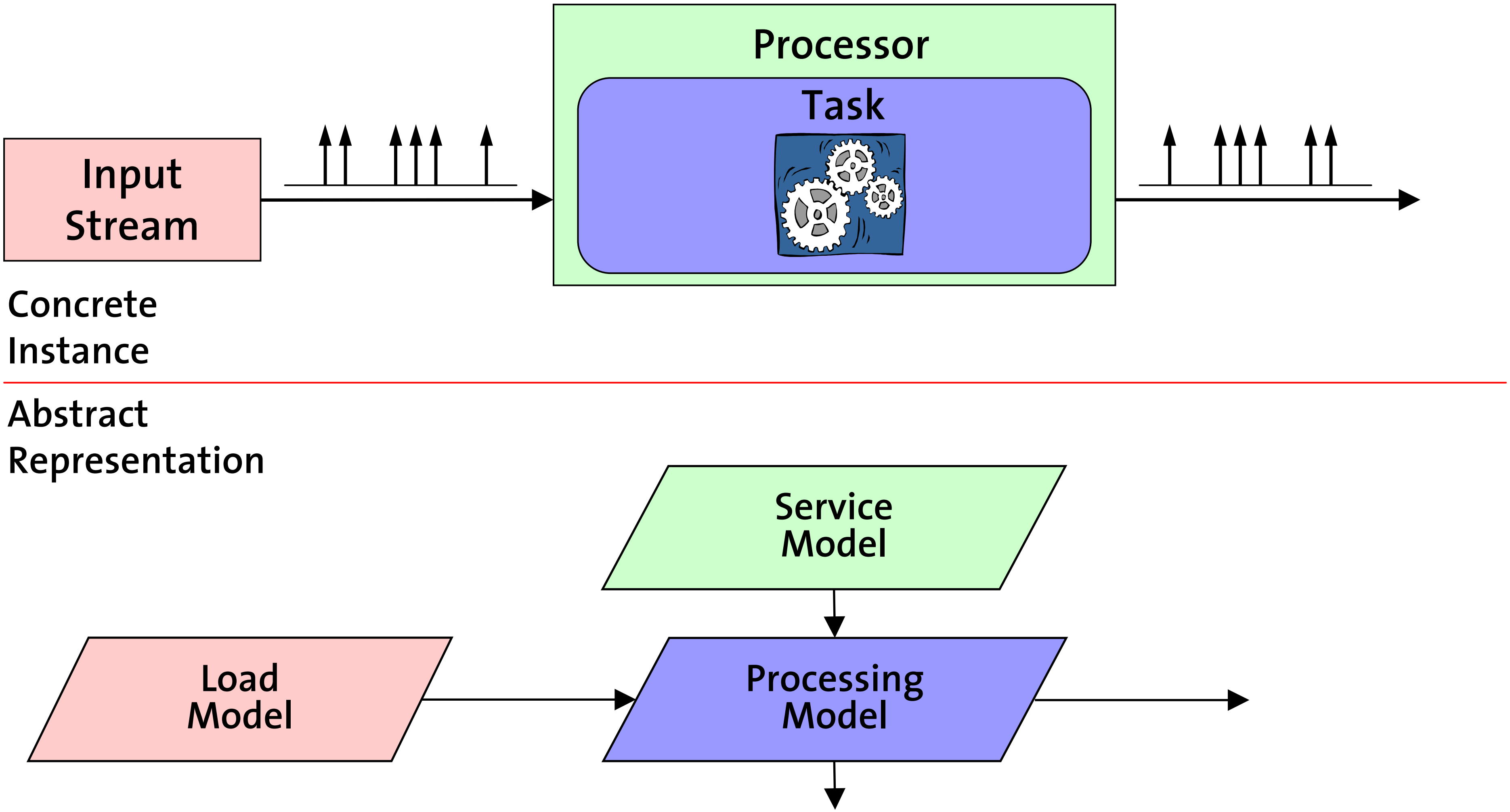
a. changes in environment
b. changes of requirements



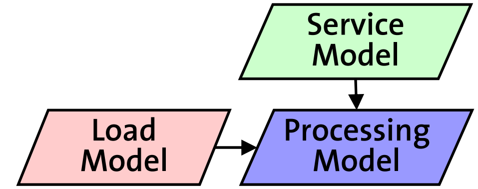
Modular Performance Analysis



Abstract Models for Performance Analysis

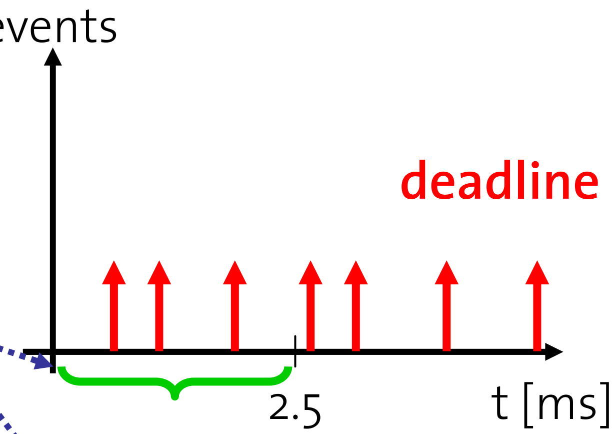


Load Model (Environment)



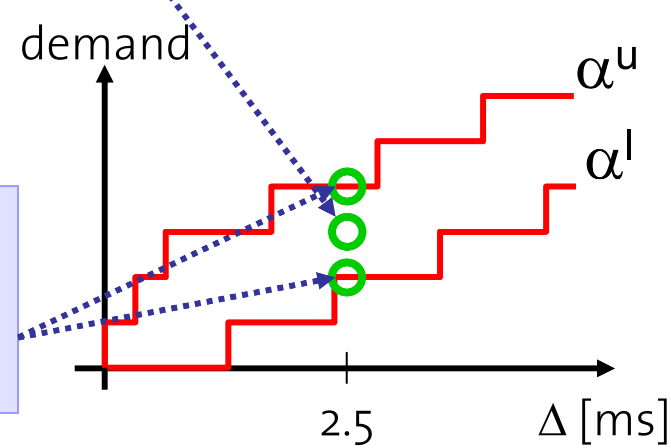
Event Stream

number of events in
in t=[0 .. 2.5] ms

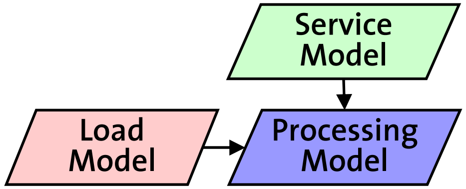


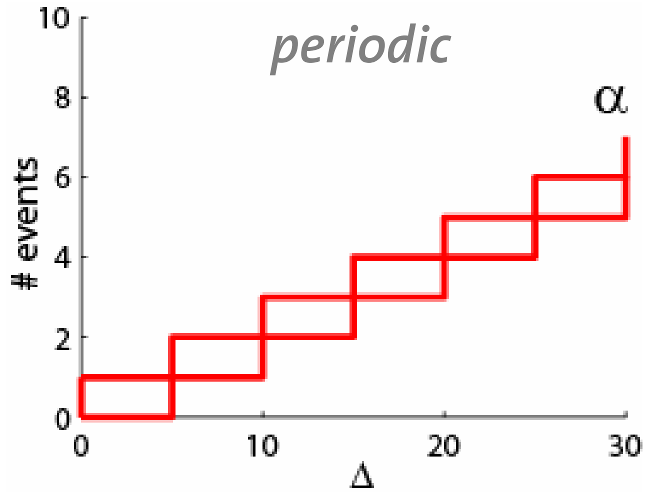
Arrival Curve α & Delay d

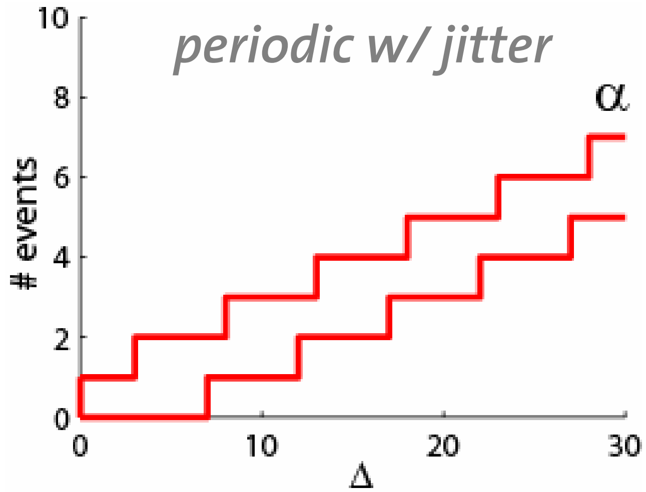
maximum / minimum
arriving demand in *any*
interval of length 2.5 ms

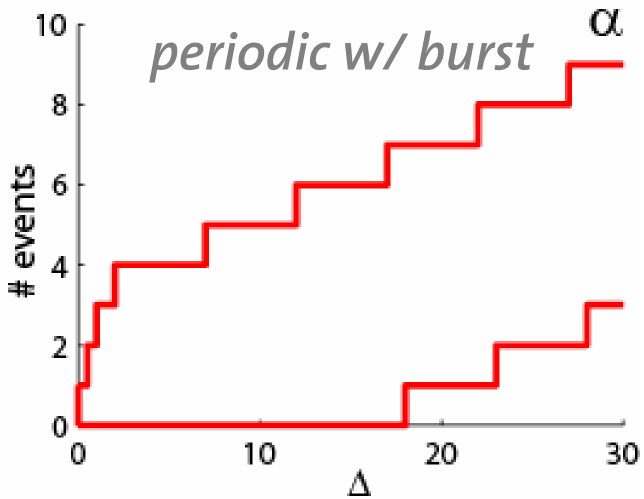


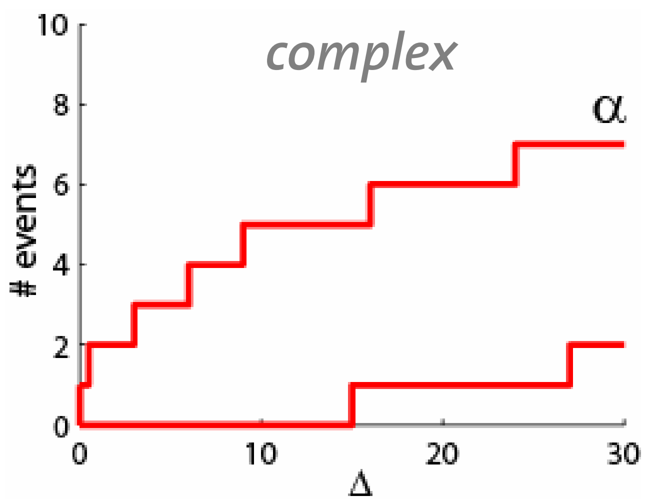
Load Model - Examples



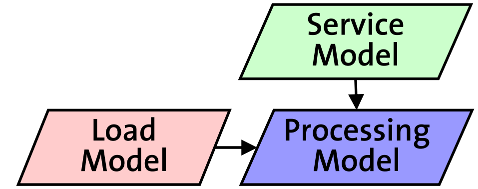






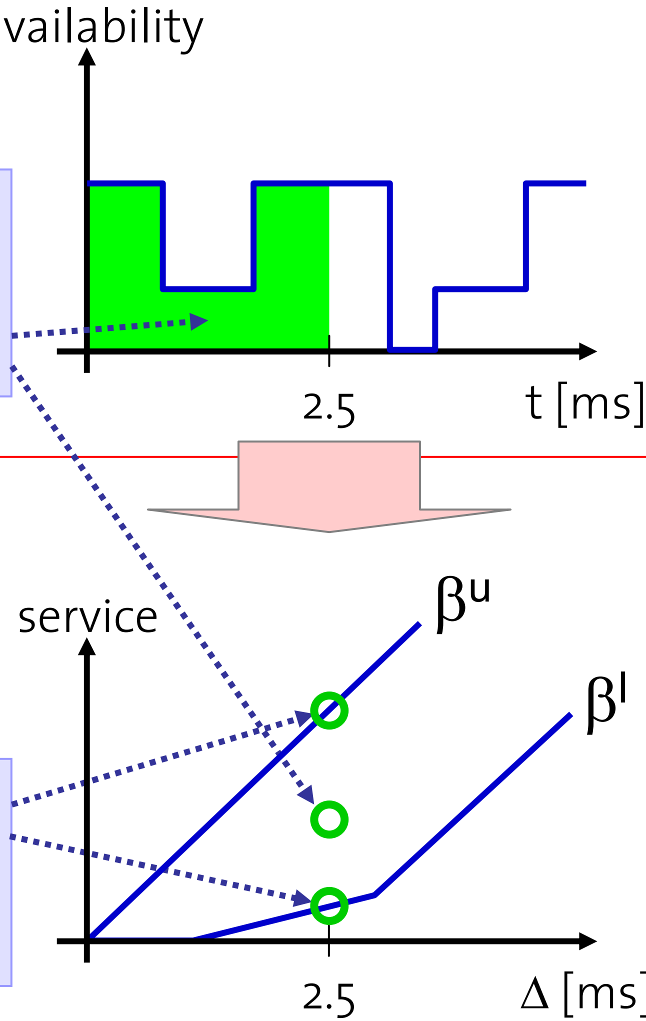


Service Model (Resources)



Resource Availability

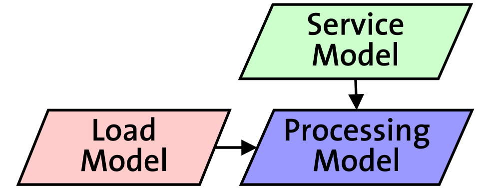
available service
in t=[0 .. 2.5] ms

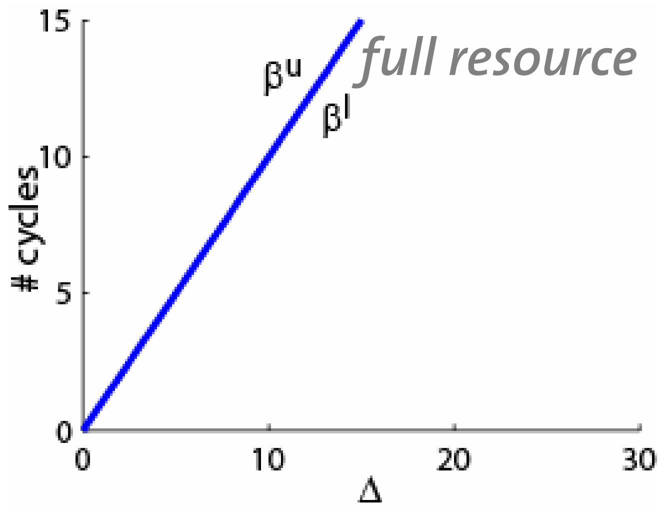


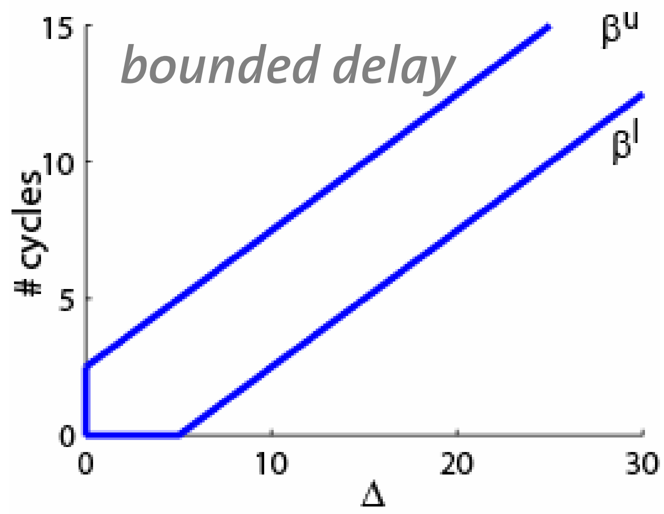
Service Curves [β^l, β^u]

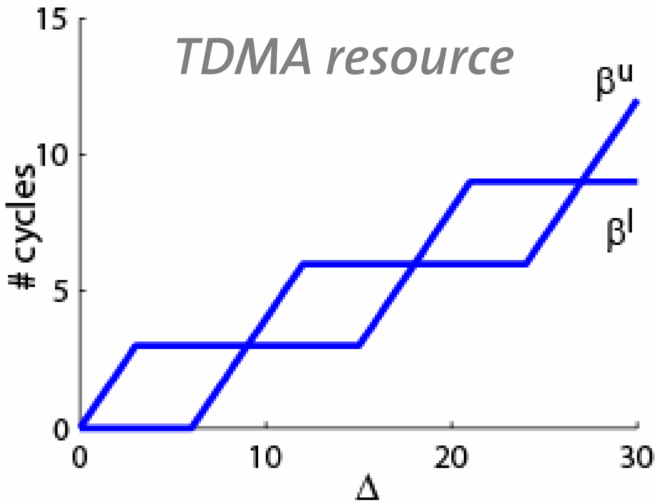
maximum/minimum
available service in *any*
interval of length 2.5 ms

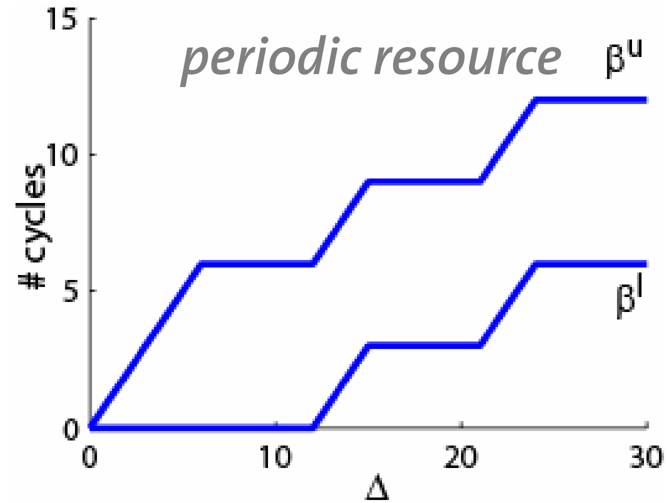
Service Model - Examples

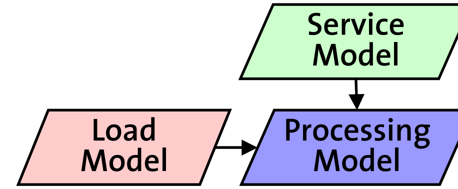








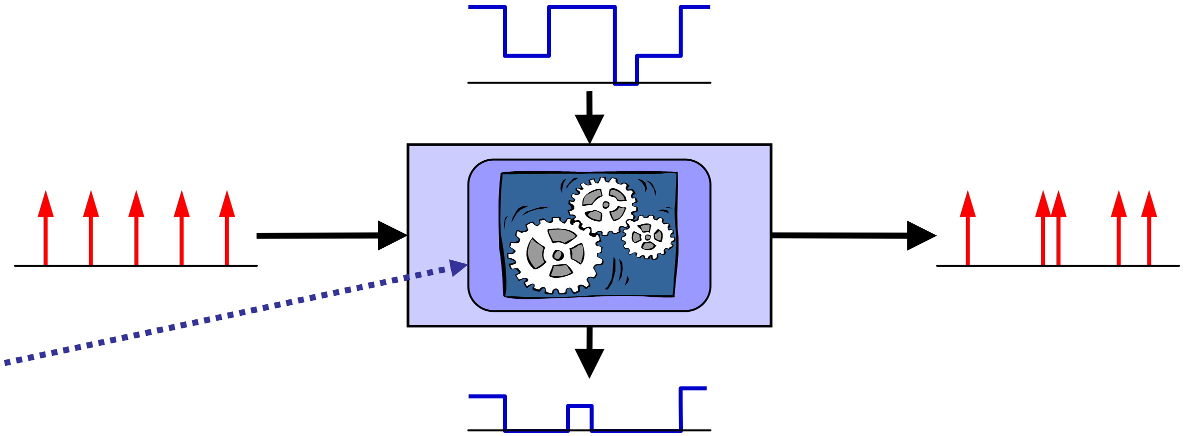




Processing Model (HW/SW)

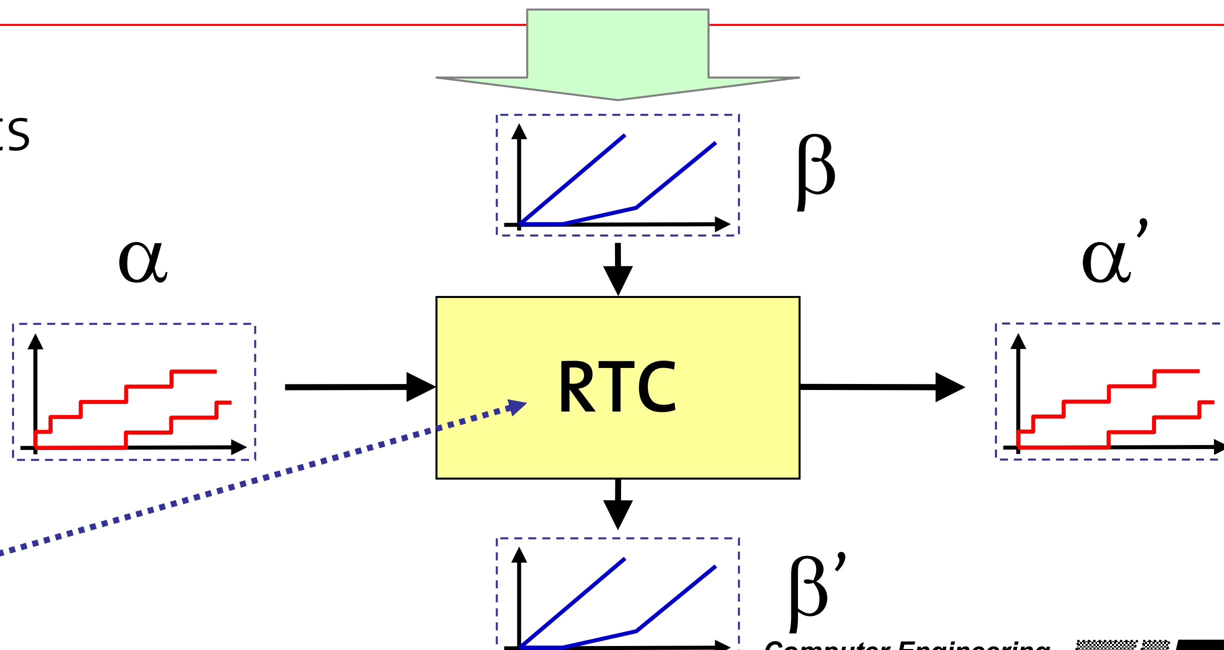
HW/SW Components

Processing semantics and functionality of hardware or software tasks

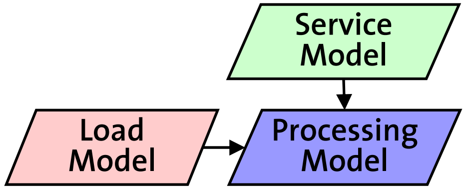


Abstract Components

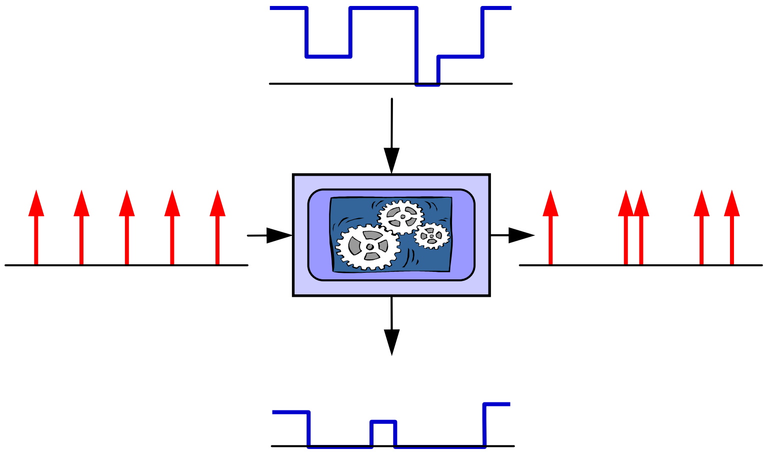
$$\alpha'(\Delta) = f_\alpha(\alpha, \beta)$$
$$\beta'(\Delta) = f_\beta(\alpha, \beta)$$



Processing Model – Examples

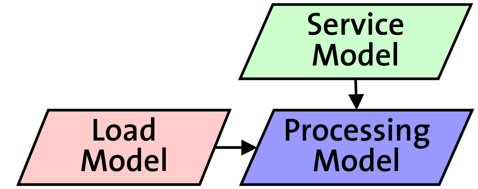


Greedy Processing Component



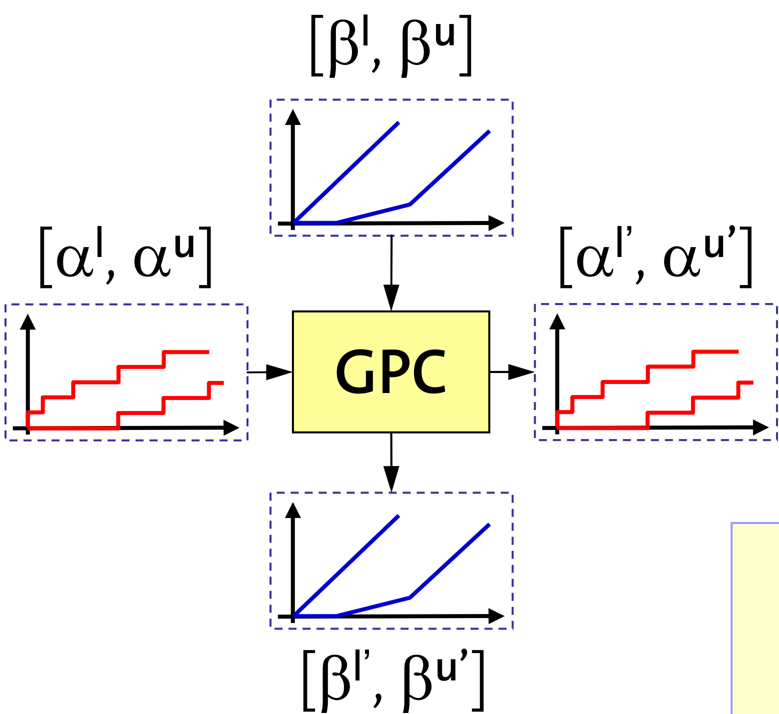
Behavioral Description

- Component is triggered by incoming events.
- A fully preemptable task is instantiated at every event arrival to process the incoming event.
- Active tasks are processed in a greedy fashion in FIFO order.
- Processing is restricted by the availability of resources.



Processing Model – Examples

Greedy Processing Component

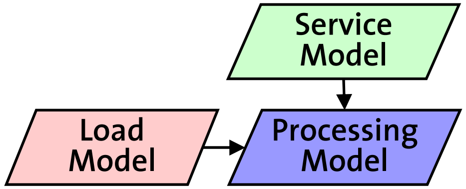


Real-Time Calculus

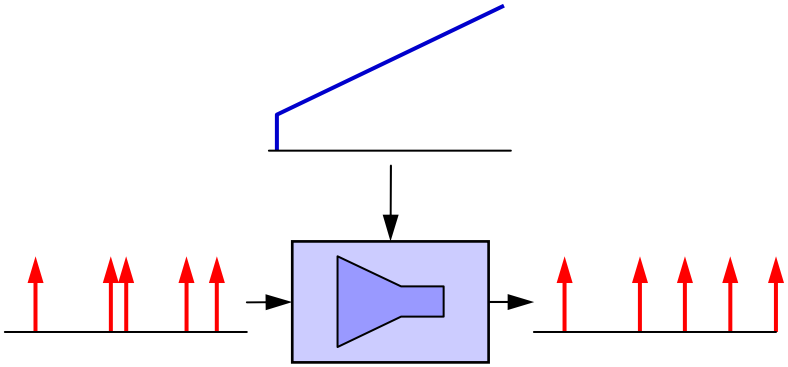
$$
\begin{aligned}
\alpha'^{u} &= \min\{(\alpha^{u} \otimes \beta^{u}) \oslash \beta^{l}, \beta^{u}\} \\
\alpha'^{l} &= \min\{(\alpha^{l} \oslash \beta^{u}) \otimes \beta^{l}, \beta^{l}\} \\
\beta'^{u} &= (\beta^{u} - \alpha^{l}) \,\overline{\oslash}\, 0 \\
\beta'^{l} &= (\beta^{l} - \alpha^{u}) \,\overline{\otimes}\, 0
\end{aligned}
$$

$$
\begin{aligned}
(f \otimes g)(\Delta) &= \inf_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\} \\
(f \oslash g)(\Delta) &= \sup_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\} \\
(f \,\overline{\otimes}\, g)(\Delta) &= \sup_{0 \le \lambda \le \Delta} \{f(\Delta - \lambda) + g(\lambda)\} \\
(f \,\overline{\oslash}\, g)(\Delta) &= \inf_{\lambda \ge 0} \{f(\Delta + \lambda) - g(\lambda)\}
\end{aligned}
$$

Processing Model – Examples



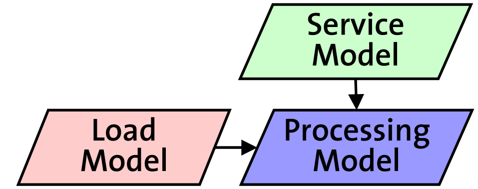
Greedy Shaper Component



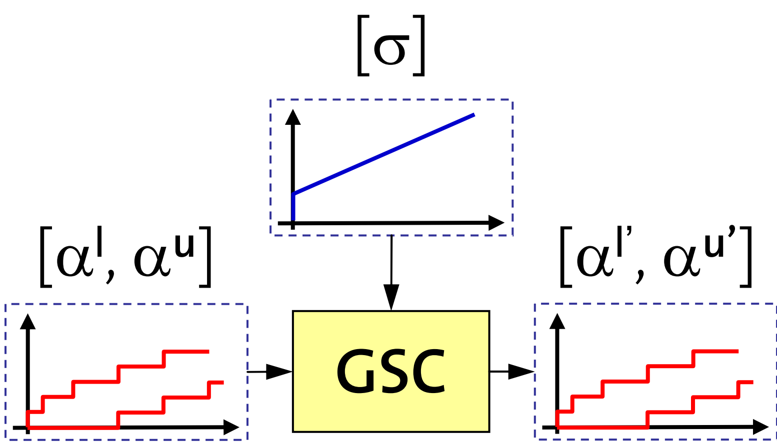
Behavioral Description

- Delays incoming events such that the output conforms to a given traffic specification.
- Guarantees that no events get delayed any longer than necessary.
- Works also with bursty traffic specifications.

Processing Model – Examples



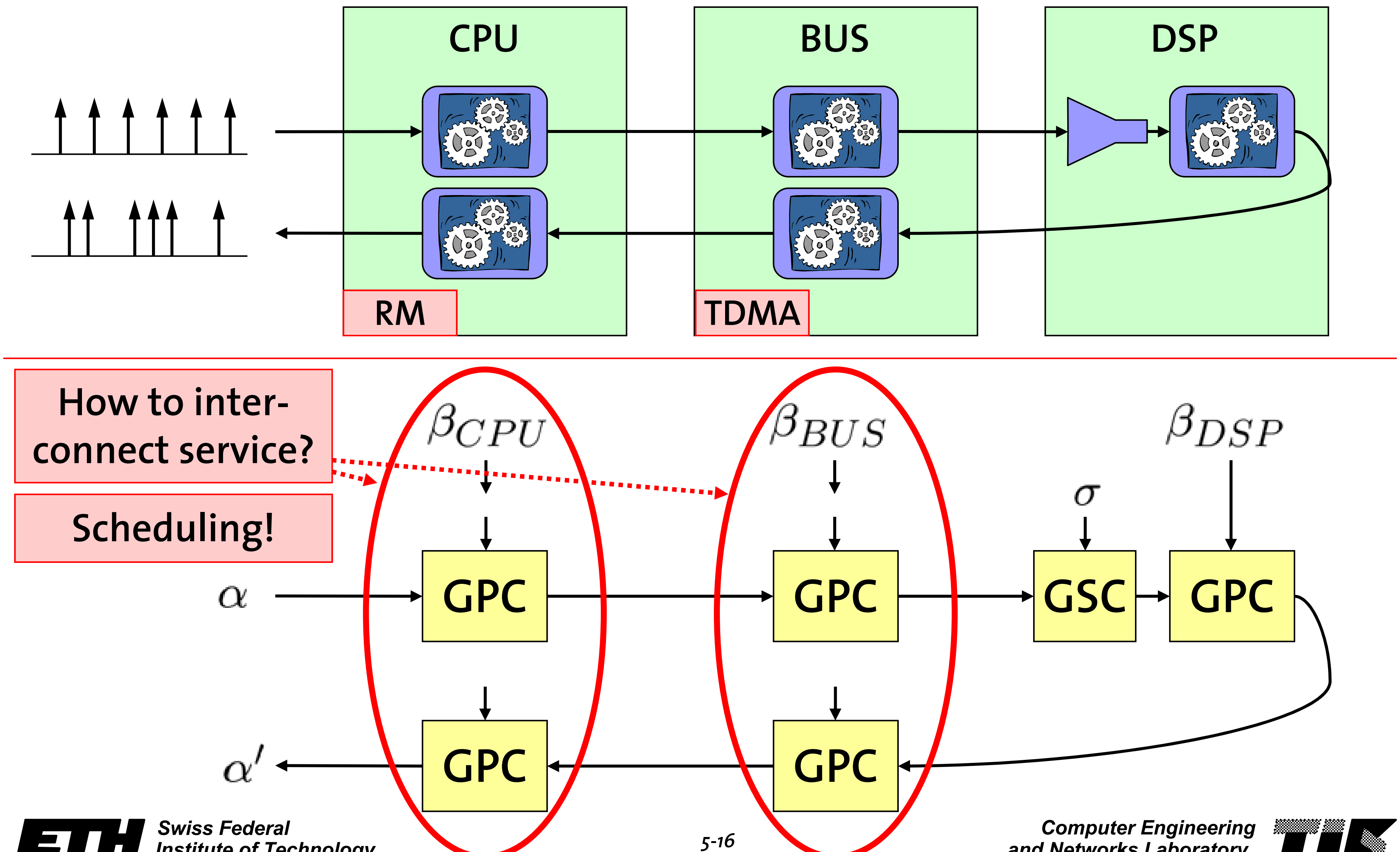
Greedy Shaper Component



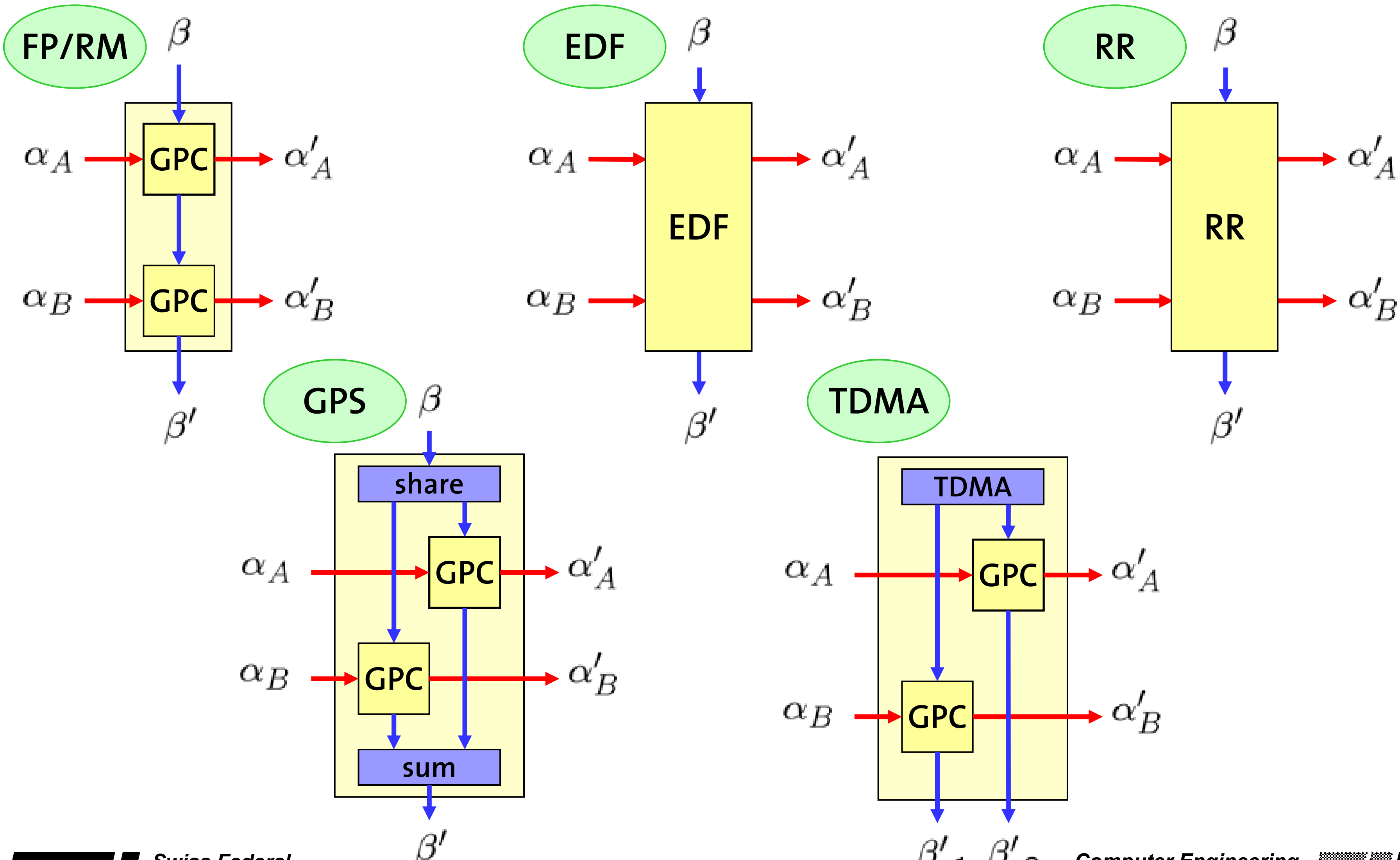
Real-Time Calculus

$$\alpha'^{u} = \alpha^{u} \otimes \sigma$$
$$\alpha'^{l} = \alpha^{l} \otimes (\sigma \overline{\oslash} \sigma)$$

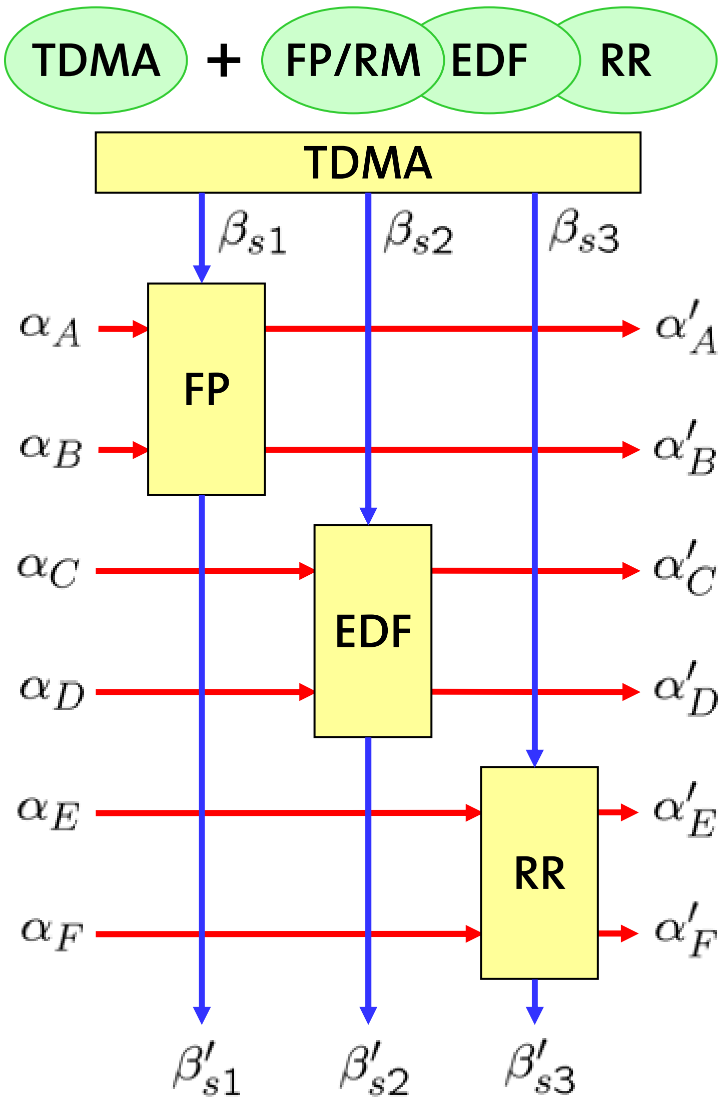
System Composition

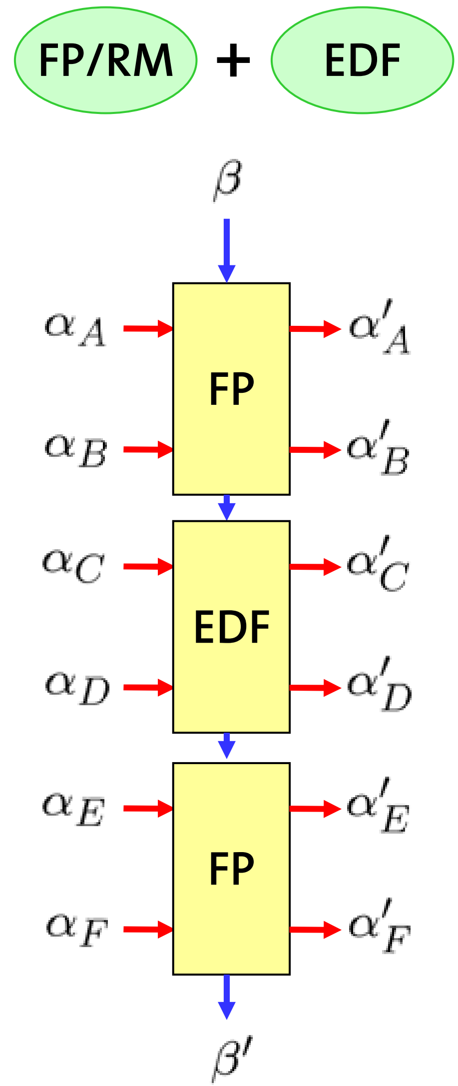


Scheduling and Arbitration



Mixed Hierarchical Scheduling



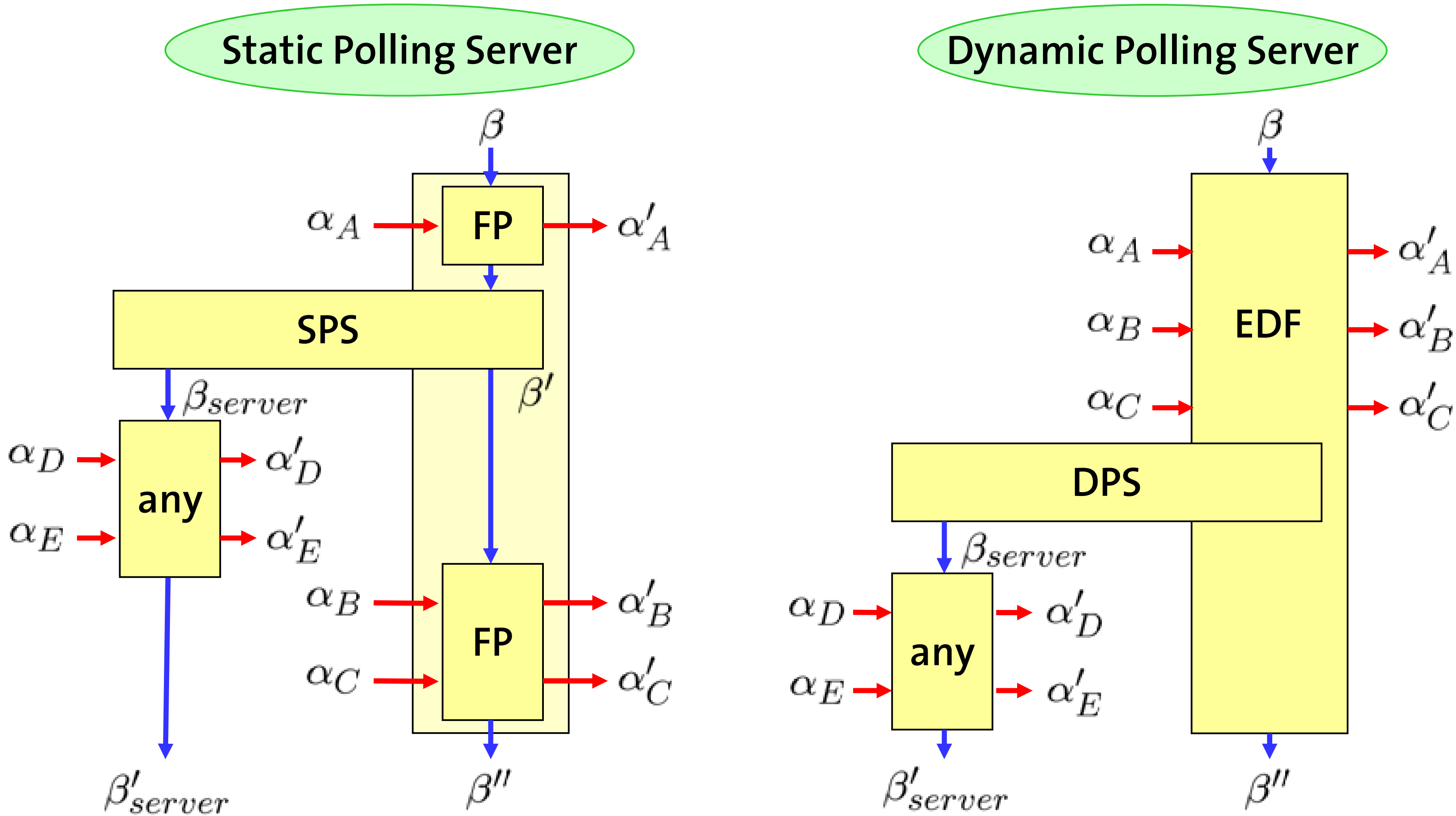


...and many other combinations:

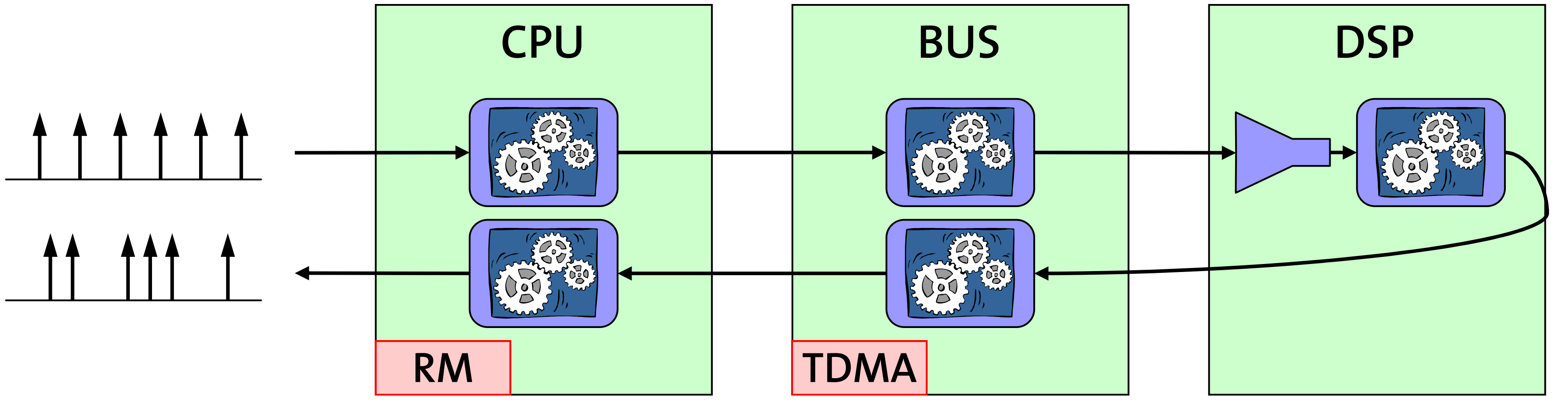
- RR + EDF
- FP/RM + RR
- FP/RM + GPS
- GPS + EDF

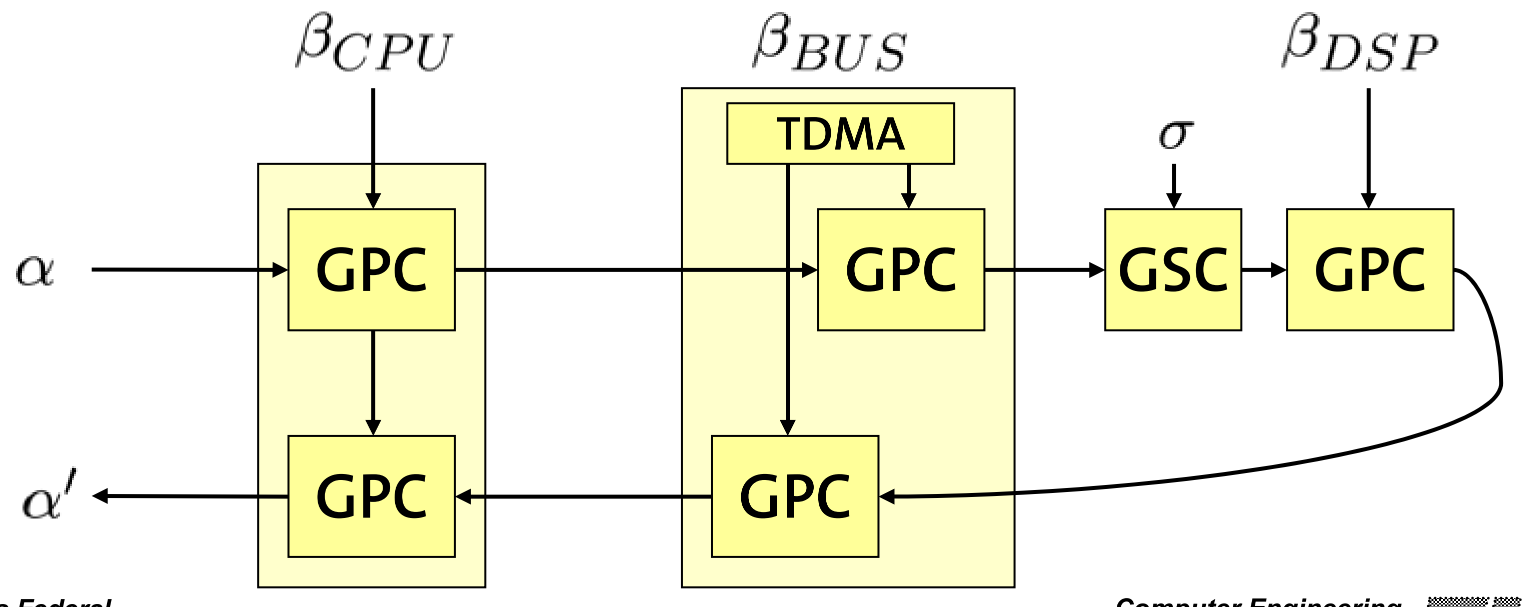
...

Hierarchical Scheduling with Servers

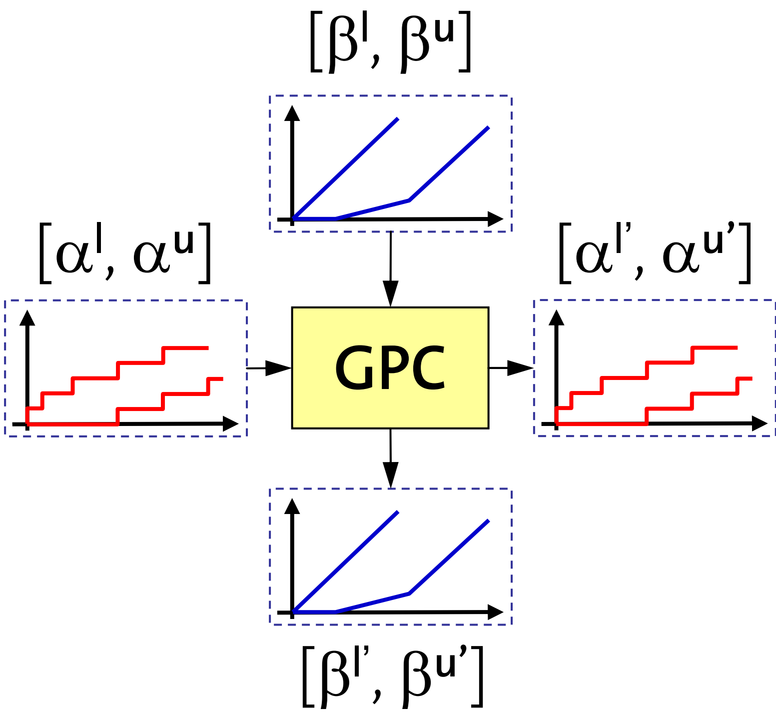


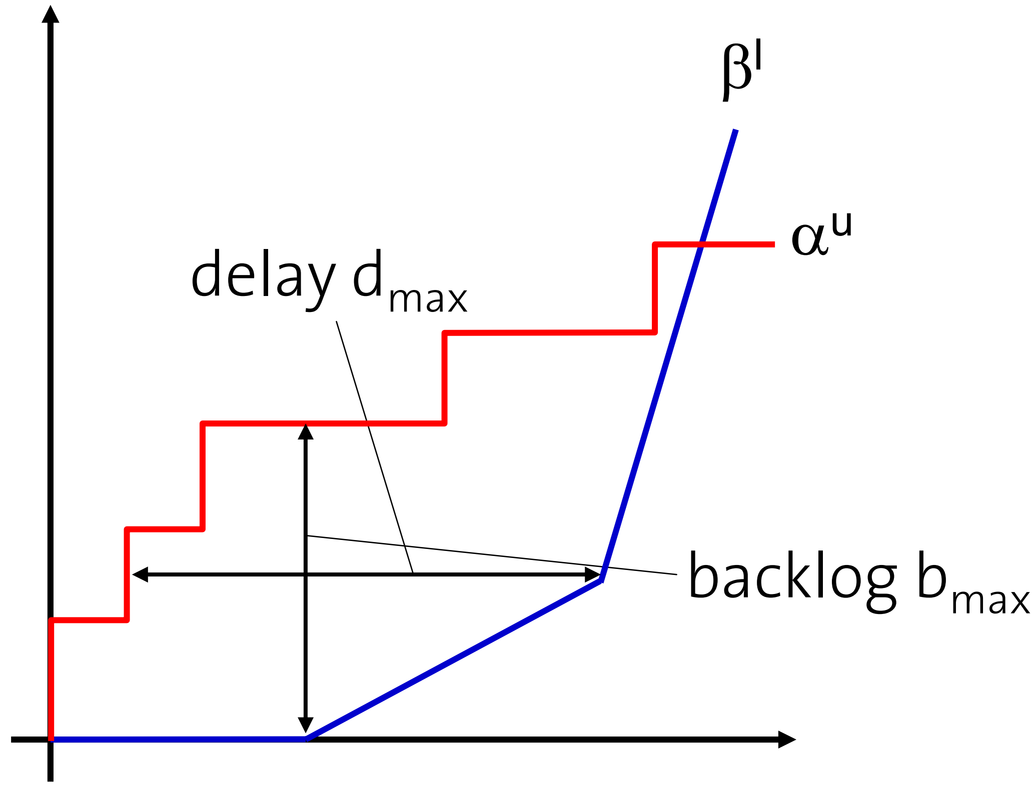
Complete System Composition





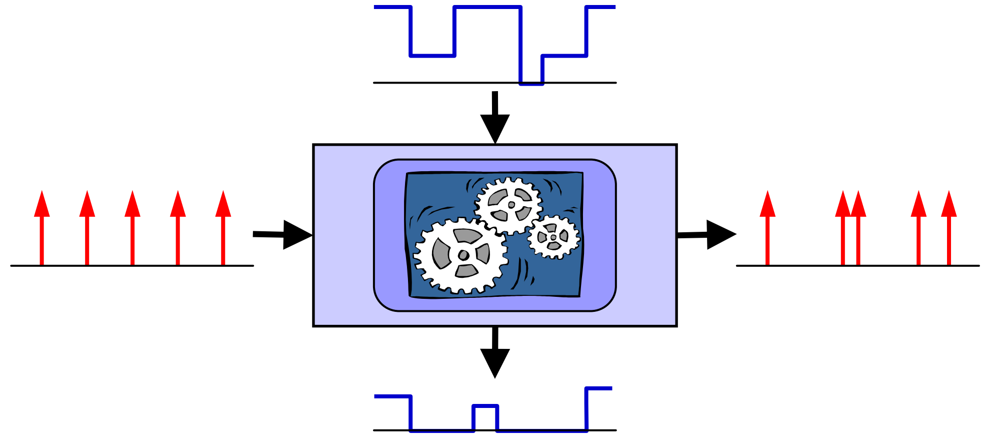
Analysis: Delay and Backlog





Extending the Framework

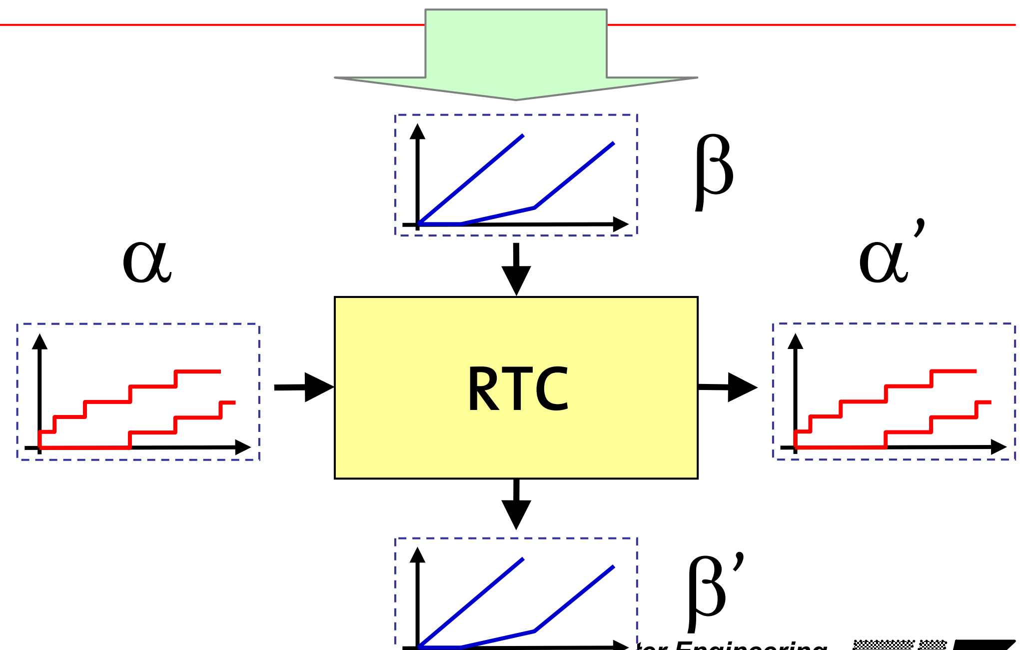
- New HW behavior
- New SW behavior
- New scheduling scheme
- ...



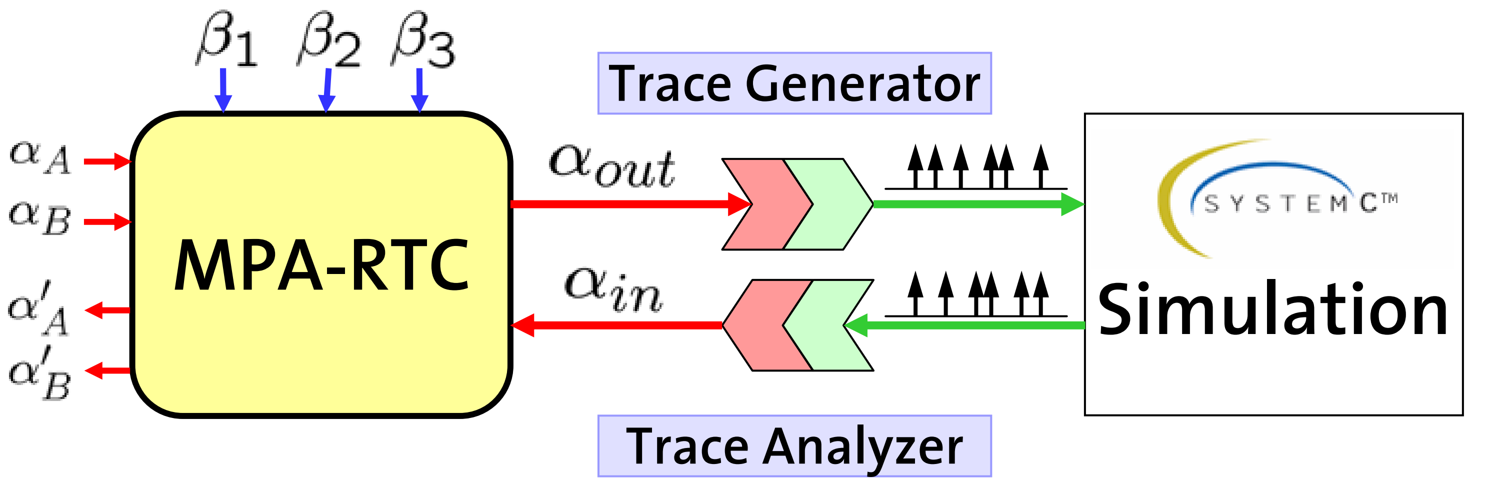
- Find new relations:

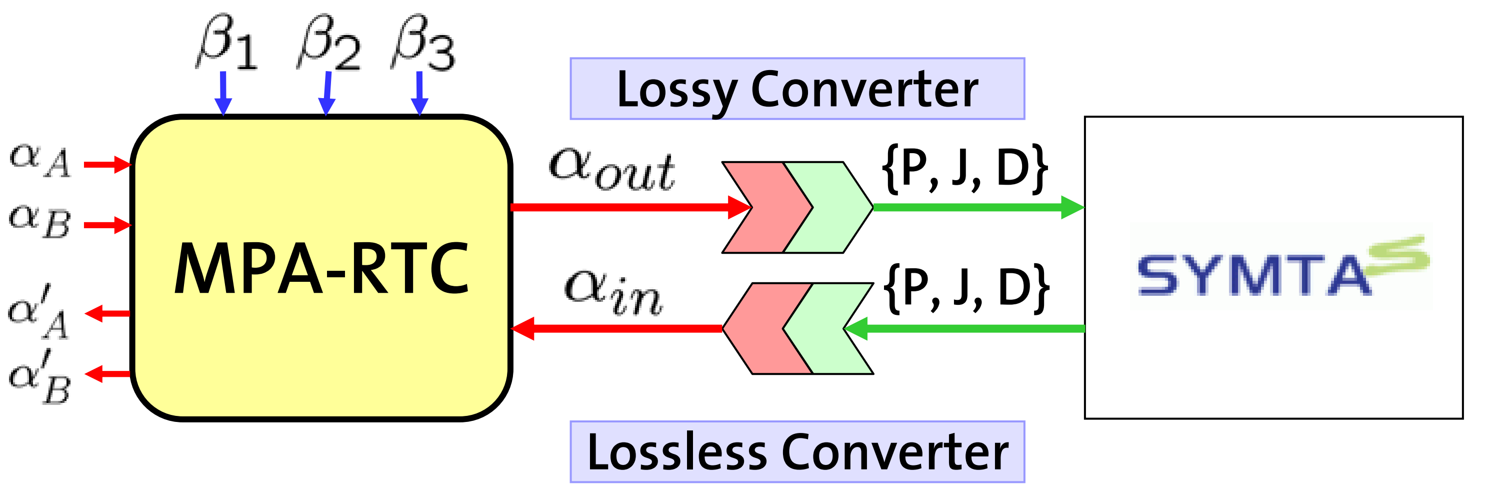
$$\alpha'(\Delta) = f_\alpha(\alpha, \beta)$$
$$\beta'(\Delta) = f_\beta(\alpha, \beta)$$

This is the hard part...!



Embedding with other Frameworks

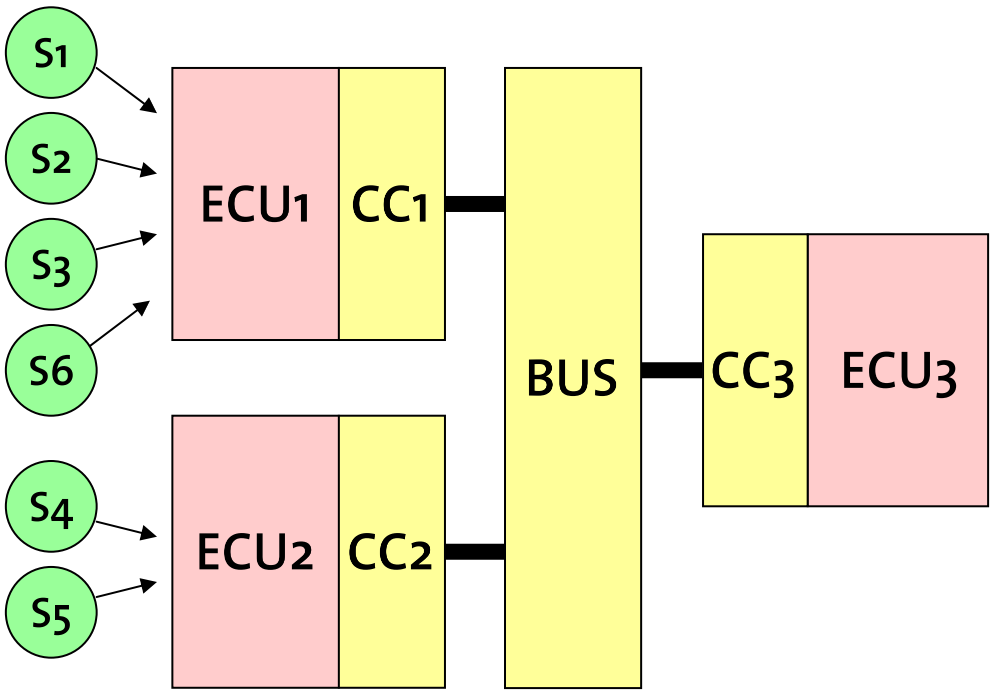




Outline

- Modular Performance Analysis
- **MPA Case Study**

Case Study



6 Real-Time Input Streams
- with jitter
- with bursts
- deadline > period

3 ECU's with own CC's

13 Tasks & 7 Messages
- with different WCED

2 Scheduling Policies
- Earliest Deadline First (ECU's)
- Fixed Priority (ECU's & CC's)

Hierarchical Scheduling
- Static & Dynamic Polling Servers

Bus with TDMA
- 4 time slots with different lengths (#1,#3 for CC1, #2 for CC3, #4 for CC3)

Total Utilization:
- ECU1 59 %
- ECU2 87 %
- ECU3 67 %
- BUS 56 %

Specification Data

Stream	(p,j,d) [ms]	D [s]	Task Chain
S1	(1000, 2000, 25)	8.0	T1.1 → C1.1 → T1.2 → C1.2 → T1.3
S2	(400, 1500, 50)	1.8	T2.1 → C2.1 → T2.2
S3	(600, 0, -)	6.0	T3.1 → C3.1 → T3.2 → C3.2 → T3.3
S4	(20, 5, -)	0.5	T4.1 → C4.1 → T4.2
S5	(30, 0, -)	0.7	T4.1 → C4.1 → T4.2
S6	(1500, 4000, 100)	3.0	T6.1

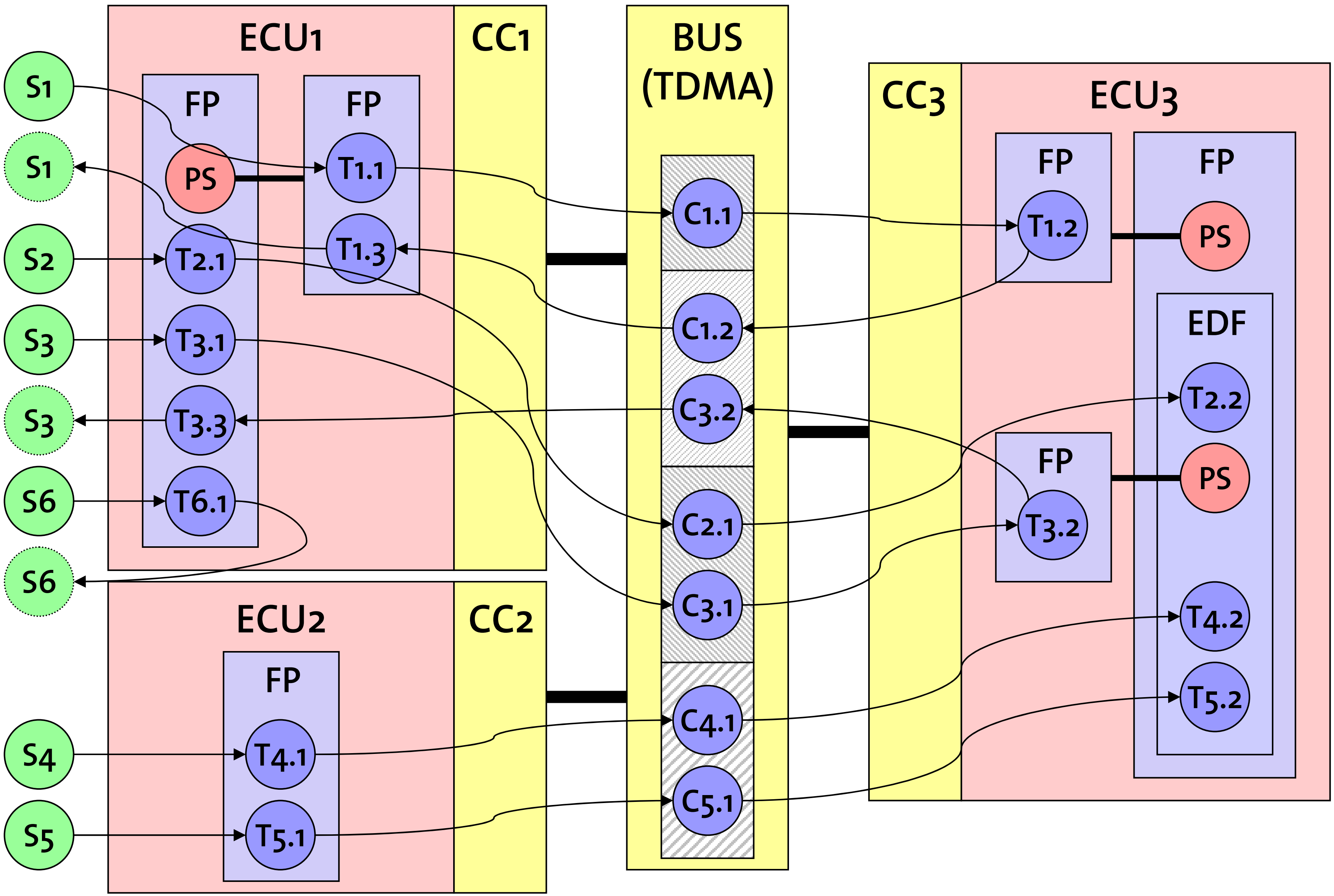
Task	e
T1.1	200
T1.2	300
T1.3	30
T2.1	75
T2.2	25
T3.1	60
T3.2	60
T3.3	40
T4.1	12
T4.2	2
T5.1	8
T5.2	3
T6.1	100

Message	e
C1.1	100
C1.2	80
C2.1	40
C3.1	25
C3.2	10
C4.1	3
C5.1	2

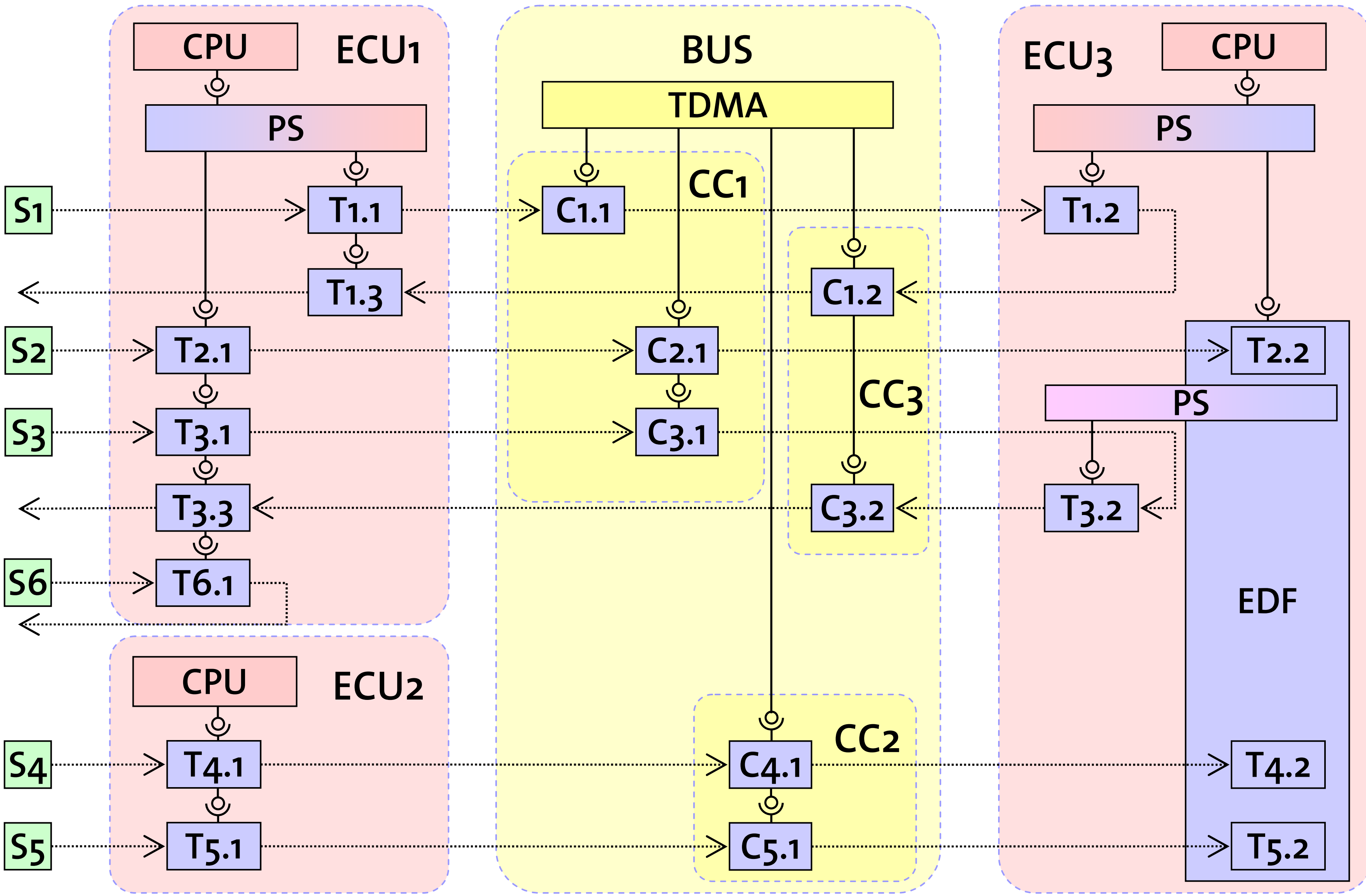
Perdiodic Server	p	e
SPS_{ECU1}	500	200
SPS_{ECU3}	500	250
DPS_{ECU3}	600	120

TDMA	t
Cycle	100
Slot_{CC1a}	20
Slot_{CC1b}	25
Slot_{CC2}	25
Slot_{CC3}	30

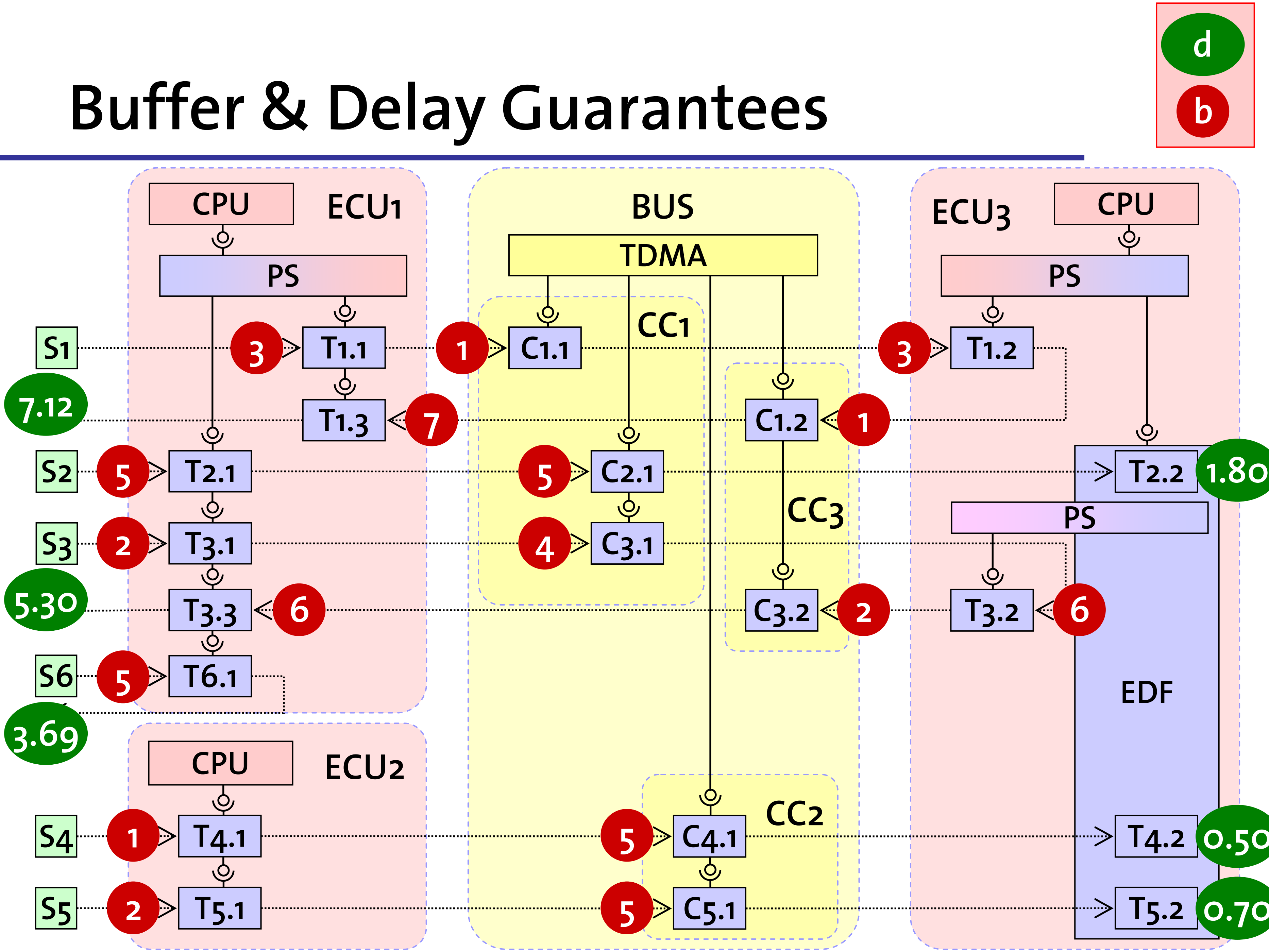
The Distributed Embedded System...



… and its MPA Model

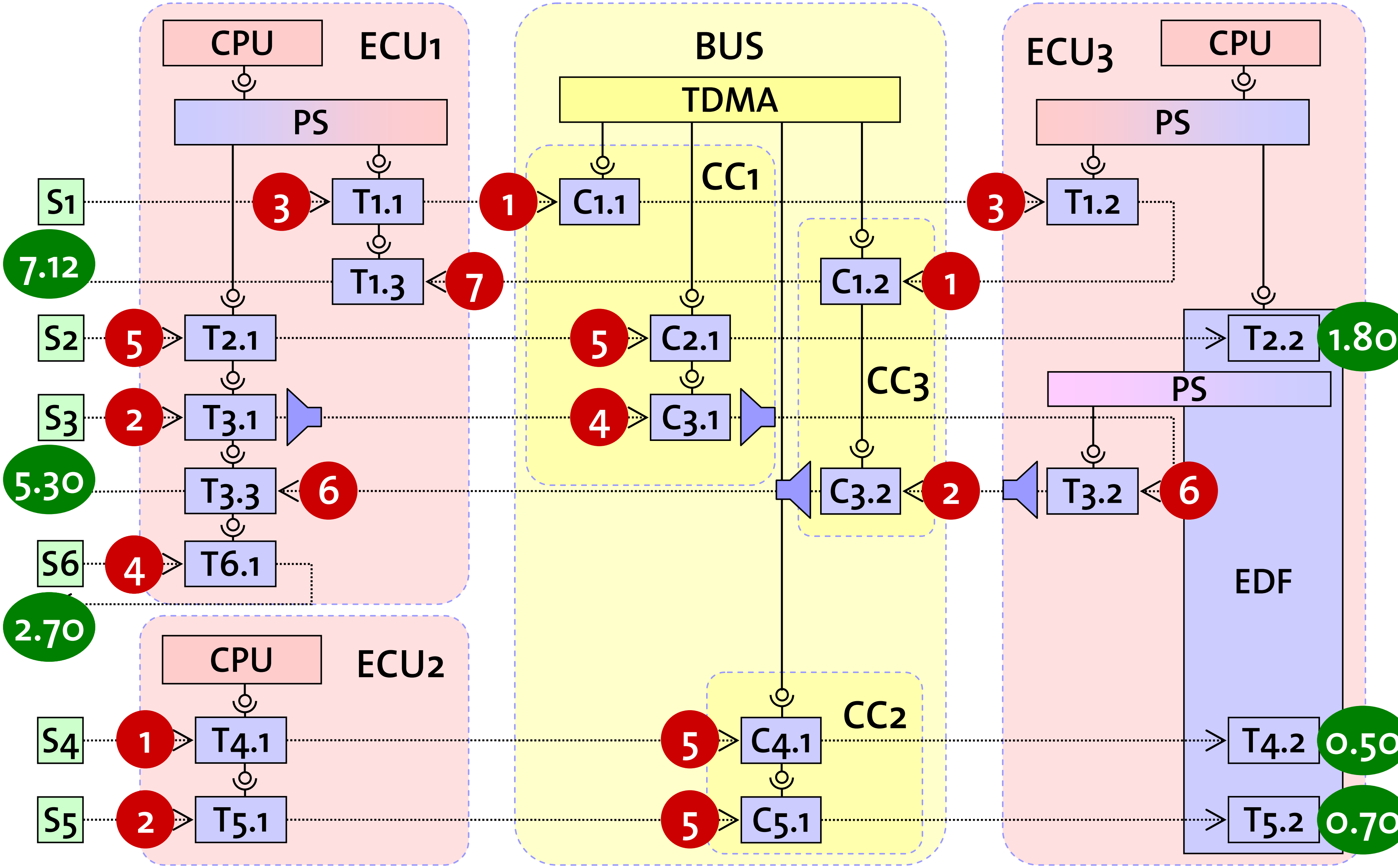


Buffer & Delay Guarantees

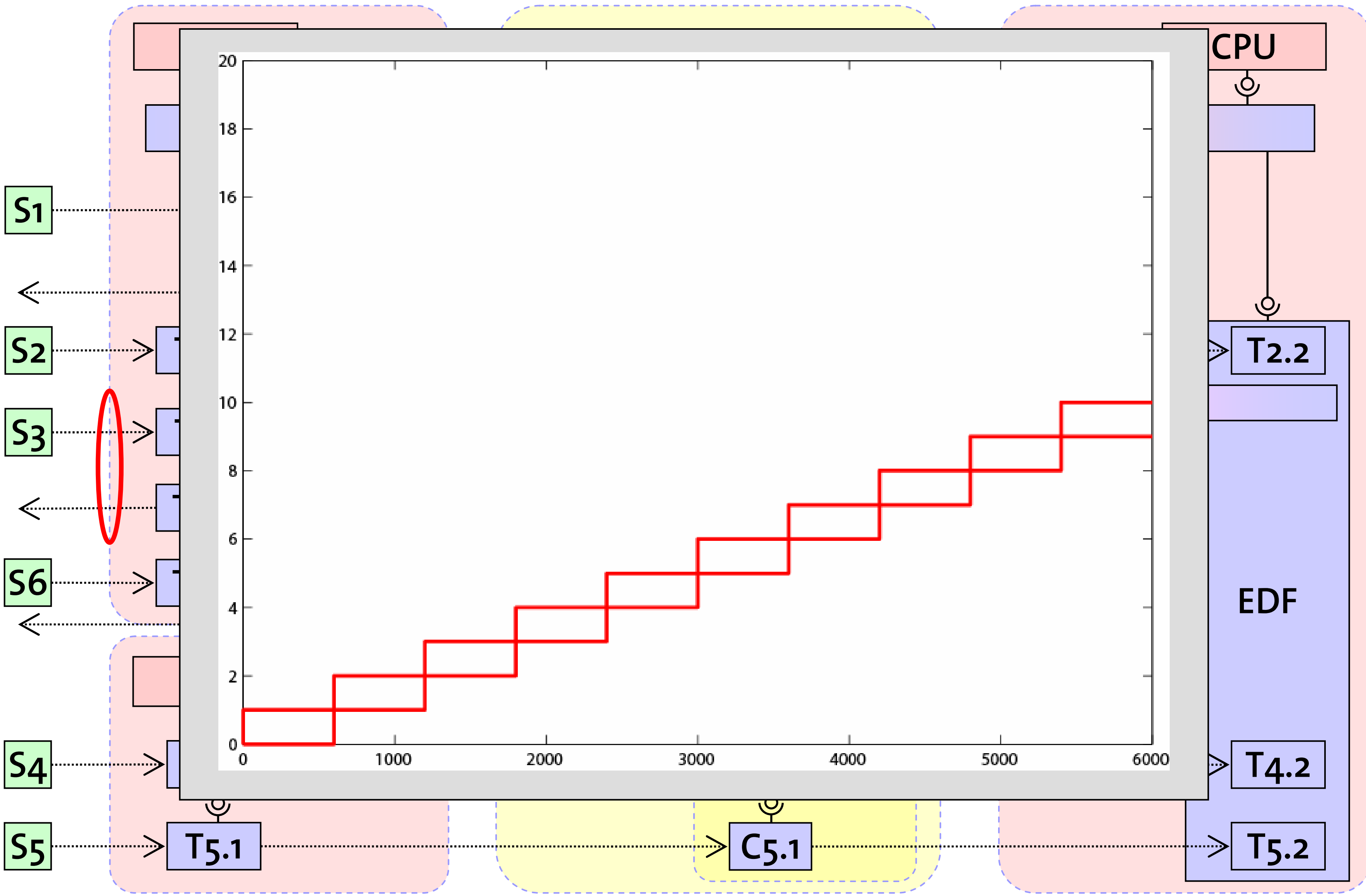


Adding Greedy Shapers

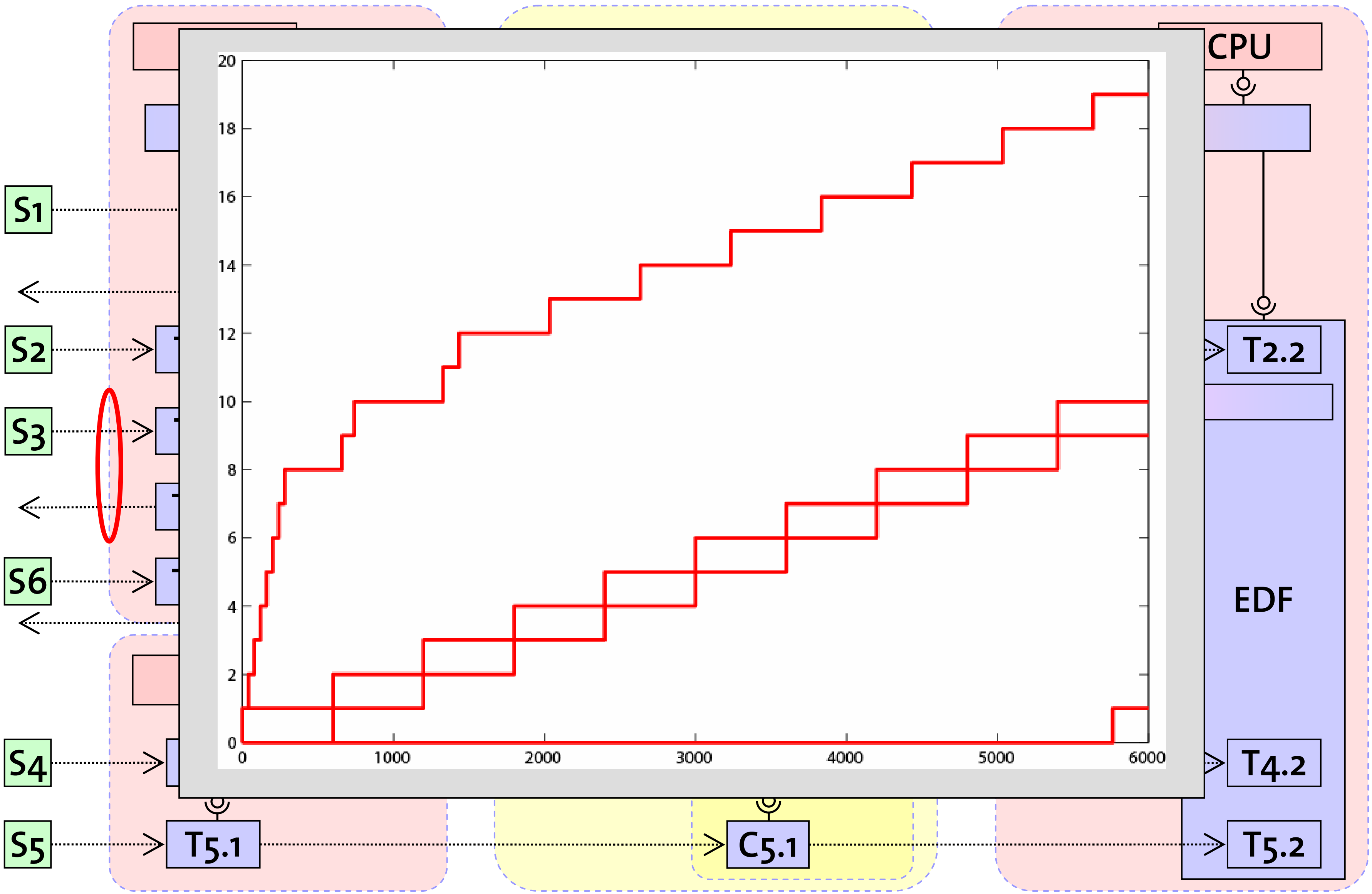
Delay D_{S6}: - 27%
Buffer B_{S6}: - 20%



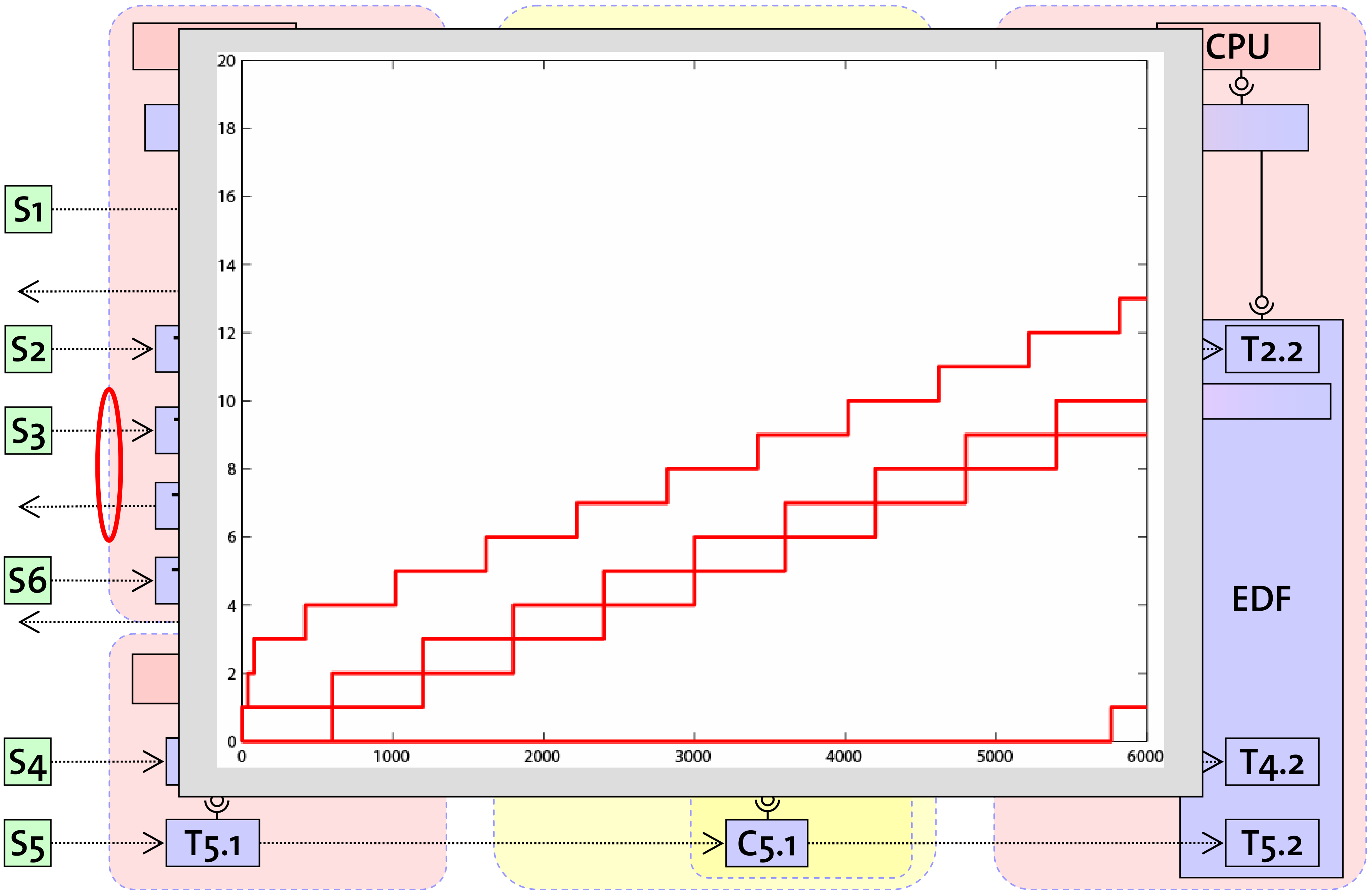
Input of Stream 3



Output of Stream 3



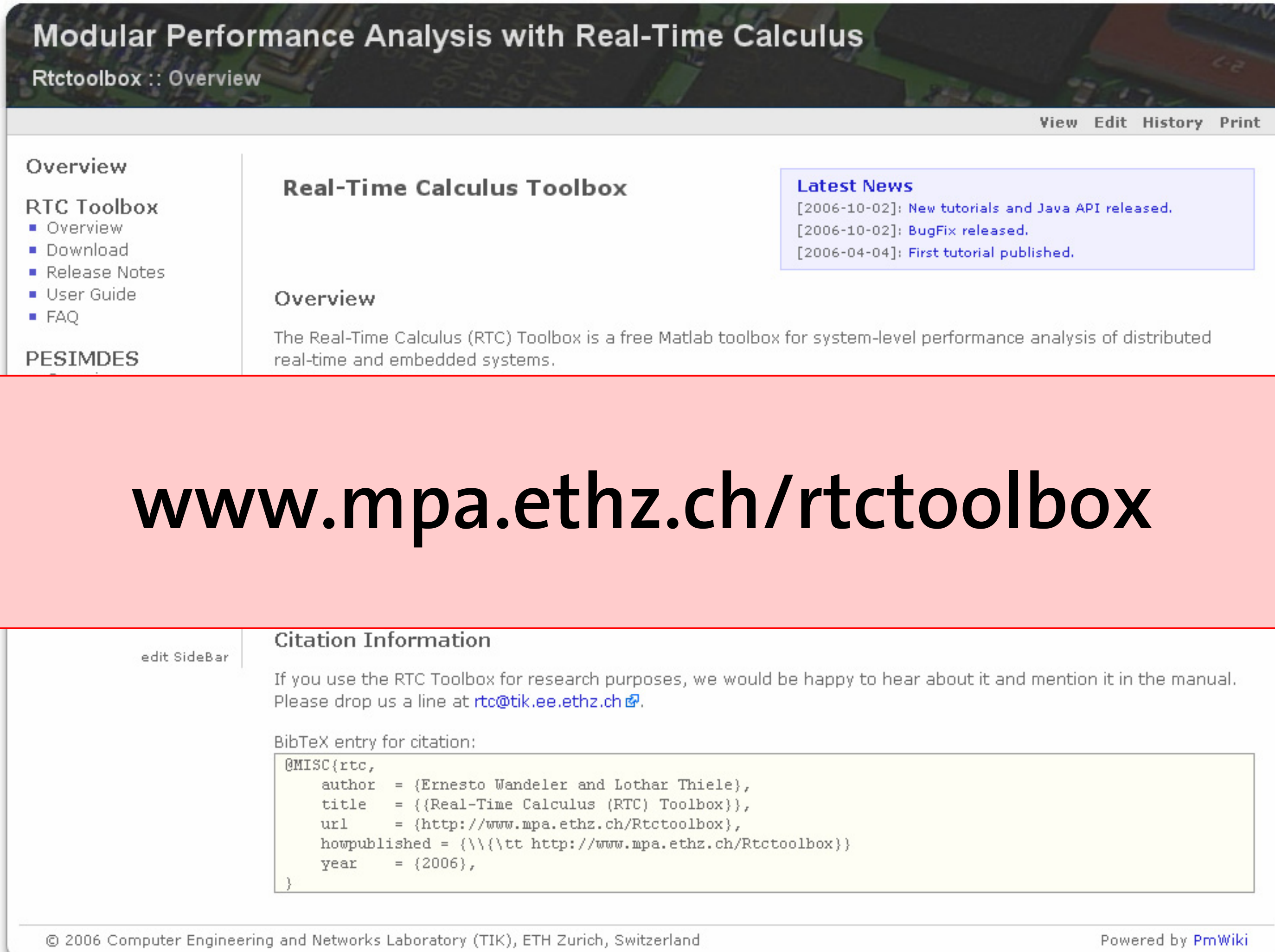
Output of Stream 3 with Greedy Shapers



System Analysis Time

- 10 seconds
  - Pentium Mobile 1.6 GHz
  - Matlab 7 SP2
  - RTC Toolbox

RTC Toolbox: Version 1.0 Released


Modular Performance Analysis with Real-Time Calculus

Rtctoolbox :: Overview

View Edit History Print

Overview

RTC Toolbox
- Overview
- Download
- Release Notes
- User Guide
- FAQ

PESIMDES

edit SideBar

Real-Time Calculus Toolbox

Latest News

[2006-10-02]: New tutorials and Java API released.

[2006-10-02]: BugFix released.

[2006-04-04]: First tutorial published.

Overview

The Real-Time Calculus (RTC) Toolbox is a free Matlab toolbox for system-level performance analysis of distributed real-time and embedded systems.

Citation Information

If you use the RTC Toolbox for research purposes, we would be happy to hear about it and mention it in the manual. Please drop us a line at rtc@tik.ee.ethz.ch.

BibTeX entry for citation:

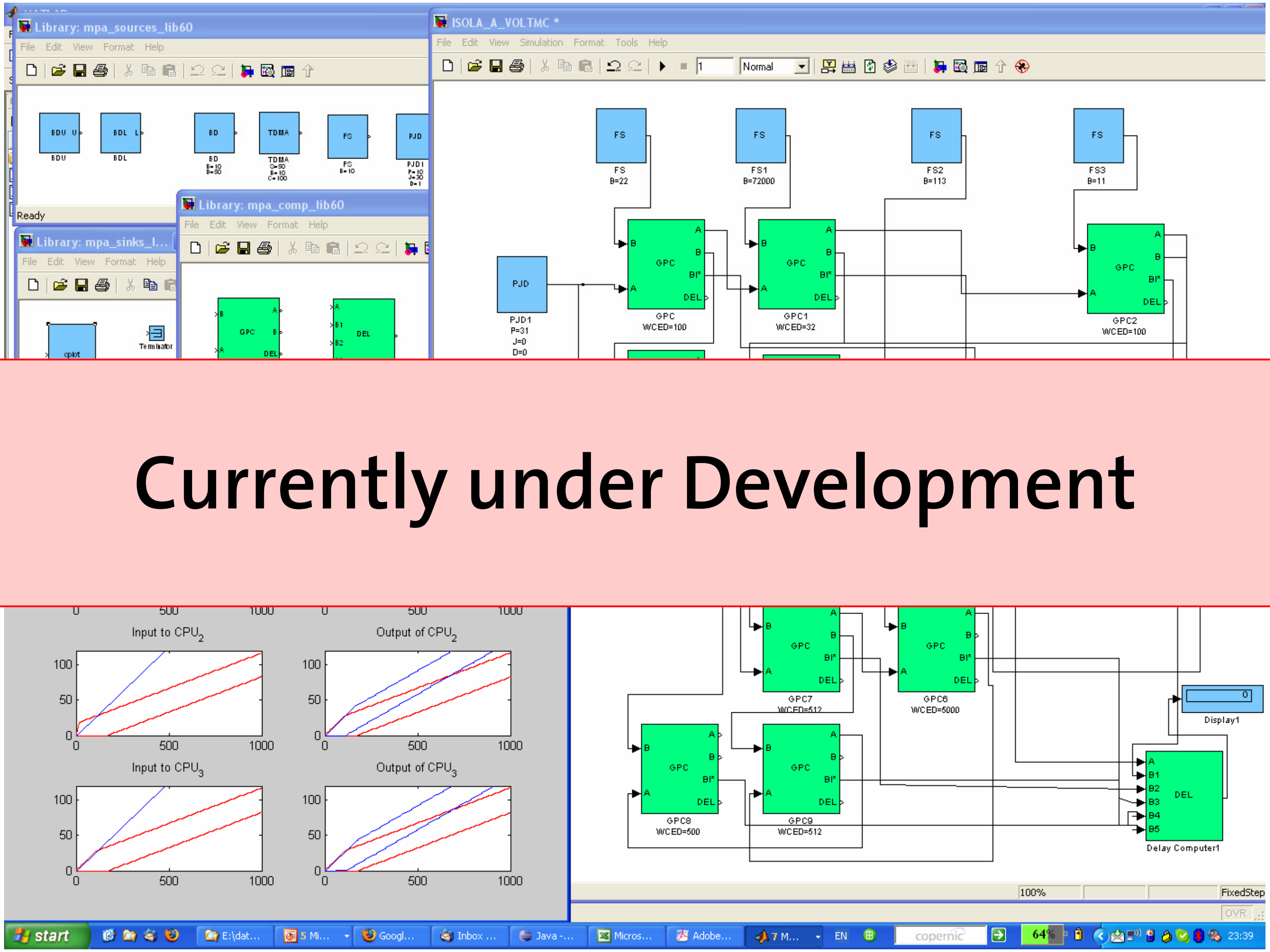
```
@MISC{rtc,
    author  = {Ernesto Wandeler and Lothar Thiele},
    title   = {{Real-Time Calculus (RTC) Toolbox}},
    url     = {http://www.mpa.ethz.ch/Rtctoolbox},
    howpublished = {\\{\tt http://www.mpa.ethz.ch/Rtctoolbox}}
    year    = {2006},
}
```

© 2006 Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland

Powered by PmWiki

www.mpa.ethz.ch/rtctoolbox

RTC Toolbox: Simulink Frontend



Currently under Development

Acknowledgement

- Collaborators:
  - Ernesto Wandeler
  - Samarjit Chakraborty
  - Simon Künzli
  - Alexander Maxiaguine
  - Kai Huang
- Funding:
  - SNF, KTI, MEDEA+/SPEAC, ARTIST2 NoE

Thank you!

www.mpa.ethz.ch/rtctoolbox

Nikolay Stoimenov

nikolays@tik.ee.ethz.ch

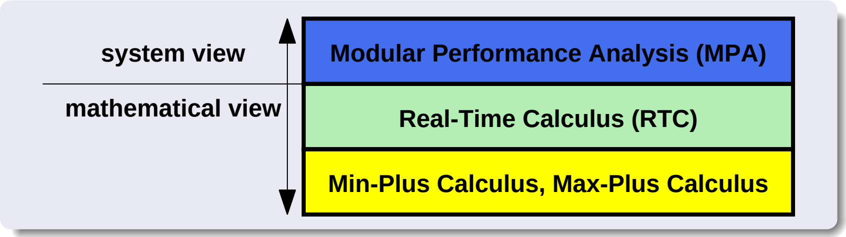
Real-Time Calculus

A Formal Method for the Analysis of Real-Time Systems

Wolfgang Haid

DTU, June 11, 2007

Overview


system view
mathematical view
Modular Performance Analysis (MPA)
Real-Time Calculus (RTC)
Min-Plus Calculus, Max-Plus Calculus

Outline

- Min-Plus Calculus
- Basic Abstractions
- System Modeling
- System Analysis

Application and Foundation

Application of Real-Time Calculus

Real-Time Calculus can be regarded as a worst-case/best-case variant of classical queuing theory. It is a formal method for the analysis of real-time embedded systems.

Foundation of Real-Time Calculus

- Min-Plus Algebra: F. Baccelli, G. Cohen, G. J. Olster, and J. P. Quadrat, *Synchronization and Linearity — An Algebra for Discrete Event Systems*, Wiley, New York, 1992.
- Network Calculus: J.-Y. Le Boudec and P. Thiran, *Network Calculus — A Theory of Deterministic Queuing Systems for the Internet*, Lecture Notes in Computer Science, vol. 2050, Springer Verlag, 2001.
- Formal methods for system level performance analysis

Comparison of Algebraic Structures (I)

Algebraic Structure

- set of elements $\mathcal{S}$
- one or more operators defined on elements of this set

Algebraic Structures With Two Operators $\odot$, $\boxdot$

- plus-times: $\{\mathbb{R}, +, \times\}$
- min-plus: $\{\mathbb{R} \cup +\infty, \inf, +\}$

inf - Reminder

$\inf(\mathcal{S})$ is the greatest lower bound of the elements in a set $\mathcal{S}$.

- $\inf\{[3,4]\} = 3, \quad \inf\{(3,4]\} = 3$
- $\min\{[3,4]\} = 3, \quad \min\{(3,4]\}$ not defined

Comparison of Algebraic Structures (II)

Common Properties: $\boxdot$

- Closure of $\boxdot$: $a \boxdot b \in \mathcal{S}$
- Associativity of $\boxdot$: $a \boxdot (b \boxdot c) = (a \boxdot b) \boxdot c$
- Commutativity of $\boxdot$: $a \boxdot b = b \boxdot a$
- Existence of identity element for $\boxdot$: $\exists \nu : a \boxdot \nu = a$
- Existence of negative element for $\boxdot$: $\exists a^{-1} : a \boxdot a^{-1} = \nu$
- Zero element for $\odot$ absorbing for $\boxdot$: $a \boxdot \epsilon = \epsilon$
- Distributivity of $\boxdot$ w.r.t. $\odot$: $a \boxdot (b \odot c) = a \odot b \boxdot b \times c$

Example: Distributive Law

- plus-times: $a \times (b + c) = a \times b + b \times c$
- min-plus: $a + \inf\{b,\ c\} = \inf\{a + b,\ a + c\}$

Comparison of Algebraic Structures (III)

Common Properties: $\odot$

- Closure of $\odot$: $a \odot b \in \mathcal{S}$
- Associativity of $\odot$: : $a \odot (b \odot c) = (a \odot b) \odot c$
- Commutativity of $\odot$: $a \odot b = b \odot a$
- Existence of identity element for $\odot$: $\exists \varepsilon : a \odot \varepsilon = a$

Different Properties: $\odot$

- Existence of negative element for $\odot$: $\exists -a : a \odot (-a) = \varepsilon$
- Idempotency of $\odot$: $a \odot a = a$

Comparison of System Theories

Plus-Times System Theory

$$f(t) \longrightarrow \boxed{g(t)} \longrightarrow h(t) = (f * g)(t) = \int_0^t f(t-s) \cdot g(s)\, ds$$

- signals
- impulse response
- convolution
- time domain

Min-Plus System Theory

$$R(\Delta) \longrightarrow \boxed{g(\Delta)} \longrightarrow R'(\Delta) \geq (R \otimes g)(\Delta) = \inf_{0 \leq \lambda \leq \Delta} \{ f(\Delta - \lambda) + g(\lambda) \}$$

- streams
- service/shaping curve
- min-plus convolution
- time-interval domain

Min-Plus/Max-Plus Convolution and Deconvolution

Definitions

min-plus convolution: $(f \otimes g)(\Delta) = \inf_{0 \leq \lambda \leq \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$

min-plus deconvolution: $(f \oslash g)(\Delta) = \sup_{\lambda \geq 0} \{f(\Delta + \lambda) - g(\lambda)\}$

max-plus convolution: $(f \overline{\otimes} g)(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \{f(\Delta - \lambda) + g(\lambda)\}$

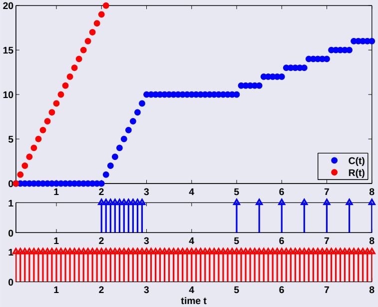
max-plus deconvolution: $(f \overline{\oslash} g)(\Delta) = \inf_{\lambda \geq 0} \{f(\Delta + \lambda) - g(\lambda)\}$

Duality between $\otimes$ and $\oslash$

$f \leq g \otimes h \quad \Leftrightarrow \quad f \oslash h \leq g$

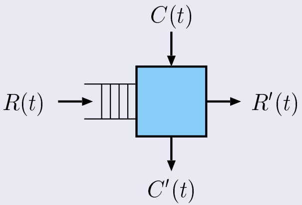
From Streams to Cumulative Functions

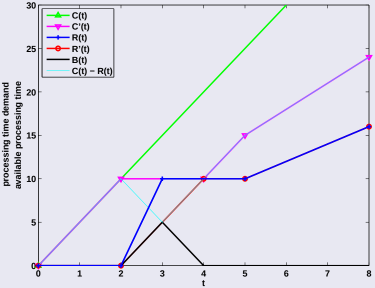
Cumulative Functions



- data streams: $R(t) :=$ number of events in $[0, t)$
- resource streams: $C(t) :=$ available resources in $[0, t)$

Greedy Processing (I)



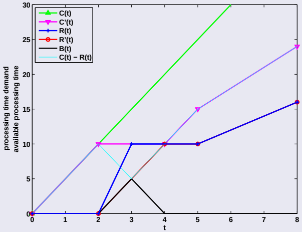


Elementary Relations

$$C(t) = C'(t) + R'(t)$$
$$B(t) = R(t) - R'(t)$$

Greedy Processing (II)

Input/Output Relation



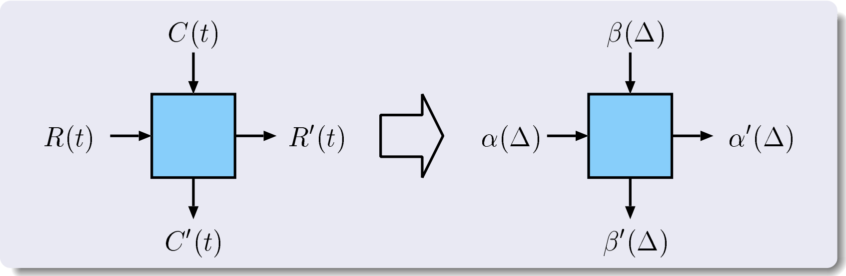
$$R'(t) = C(t) - C'(t) = C(t) - \sup_{0 \le s \le t} \{C(s) - R(s)\} \quad | \inf\{\mathcal{S}\} = \sup -\mathcal{S}$$

$$= C(t) + \inf_{0 \le s \le t} \{R(s) - C(s)\}$$

$$= \inf_{0 \le s \le t} \{R(s) + C(t) - C(s)\} \quad | \text{periodic resource}$$

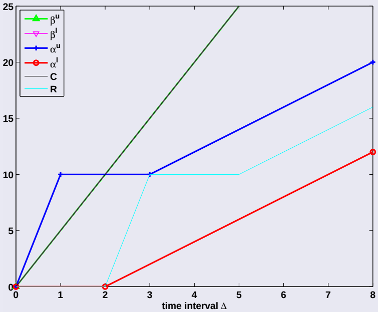
$$= \inf_{0 \le s \le t} \{R(s) + C(t-s)\} \stackrel{!}{=} (R \otimes C)(t)$$

From Cumulative Functions To Bounding Curves



Arrival and Service Curves

Definition



$$\alpha^l(t-s) \leq R[s,t] \leq \alpha^u(t-s), \quad \forall s < t,$$

$$\beta^l(t-s) \leq C[s,t] \leq \beta^u(t-s), \quad \forall s < t,$$

$$\alpha^u(0) = \alpha^l(0) = \beta^u(0) = \beta^l(0) = 0.$$

Upper Arrival Curve (I)

Stream Constraint

$$
\begin{aligned}
R(t) &\leq (R \otimes \alpha^u)(t) \\
&= \inf_{0 \leq s \leq t} \{R(s) + \alpha^u(t-s)\} \\
&\leq R(s) + \alpha^u(t-s), \quad \forall\, 0 \leq s \leq t \\
&\Leftrightarrow \\
R(t) - R(s) &\leq \alpha^u(t-s), \quad \forall s \leq t
\end{aligned}
$$

Upper Arrival Curve (II)

Upper Arrival Curve

$$
\begin{aligned}
\alpha^u(\Delta) &= (R \oslash R)(\Delta) \\
&= \sup_{s \geq 0} \{R(\Delta + s) - R(s)\} \quad |\Delta = t - s \\
&= \sup_{s \geq 0} \{R(t - s + s) - R(s)\} \\
&\geq R(t) - R(s), \quad \forall t \geq s
\end{aligned}
$$

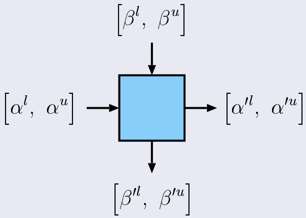
$(R \oslash R)$: Minimum Upper Arrival Curve

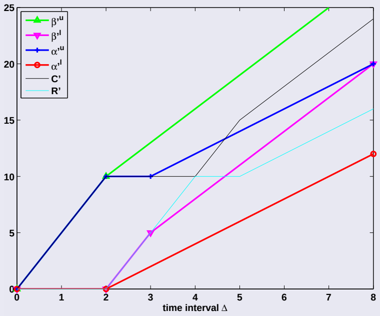
Assume $\widetilde{\alpha}^u$ is an upper arrival curve for R.

- from previous slide: $R \leq R \otimes \widetilde{\alpha}^u$
- from duality property: $R \oslash R \leq \widetilde{\alpha}^u$

Greedy Processing Component

Input/Output Relations



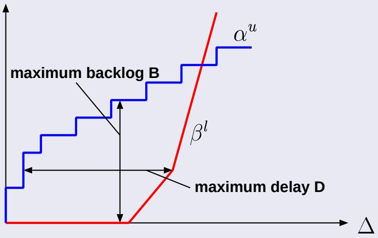


$$\alpha'^u = \min\{(\alpha^u \otimes \beta^u) \oslash \beta^l, \beta^u\}, \quad \alpha'^l = \min\{(\alpha^l \oslash \beta^u) \otimes \beta^l, \beta^l\}$$

$$\beta'^u = (\beta^u - \alpha^l) \overline{\oslash} \ 0, \quad \beta'^l = (\beta^l - \alpha^u) \overline{\otimes} \ 0$$

Backlog and Delay (I)

Definition



$$B = \sup_{0 \leq \lambda} \left\{ \alpha^u(\lambda) - \beta^l(\lambda) \right\}$$

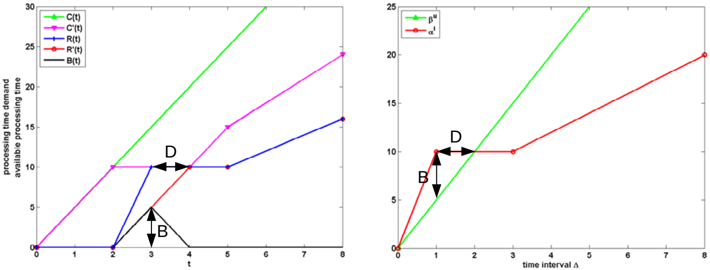
$$D = \sup_{\Delta \leq 0} \left\{ \inf \left\{ \tau \leq 0 : \alpha^u(\Delta) \leq \beta^l(\Delta + \tau) \right\} \right\}$$

Backlog and Delay (II)

Backlog Bound

$$
\begin{aligned}
B(t) = R(t) - R'(t) &= R(t) - \inf_{0 \leq u \leq t}\{R(u) + C(t) - C(u)\} \\
&= \sup_{0 \leq u \leq t} \{(R(t) - R(u)) - (C(t) - C(u))\} \\
&\leq \sup_{0 \leq u \leq t} \{\alpha^u(t-u) - \beta^l(t-u)\} \\
&\leq \sup_{0 \leq \lambda}\{\alpha^u(\lambda) - \beta^l(\lambda)\} \\
&= (\alpha^u \oslash \beta^l)(0)
\end{aligned}
$$

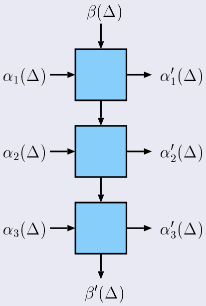
Backlog and Delay (III)



Summary: Fixed-Priority Scheduling

Key Elements of Real-Time Calculus

- min-plus calculus as well-defined mathematical basis
- abstraction of streams: arrival/service curves
- abstraction of processing: greedy processing
- delay and backlog bounds
- modularity



Complex Task Activation Schemes in System Level Performance Analysis

Wolfgang Haid

DTU, June 11, 2007

Context

Keywords

- Distributed embedded real-time systems
- System level performance analysis
- Formal methods for system level performance analysis

A Glimpse on Formal Methods

Advantages

- Hard bounds
- Complete coverage of corner cases
- Faster than simulation

Drawbacks

- Limited modeling capabilities
- Bounds potentially not tight
- Inaccuracy of results

Thesis

To obtain improved accuracy, we can sacrifice some computational effort.

Outline

- Introduction to complex task activation schemes
- Task model and analysis
- MPEG-2 case study
- Conclusion

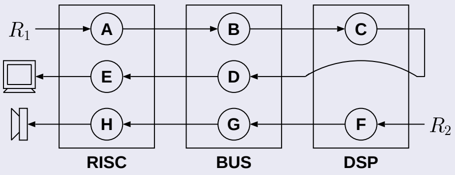
Formal Methods

Frameworks

- Modular Performance Analysis / Real-Time Calculus (MPA/RTC): Samarjit Chakraborty, Simon Künzli, and Lothar Thiele, *A General Framework for Analyzing System Properties in Platform-Based Embedded System Design*, Proc. 6th Design, Automation and Test in Europe (DATE) (Munich, Germany), March 2003, pp. 190–195.
- Symbolic Timing Analysis for Systems (SymTA/S): Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, and Rolf Ernst, *System Level Performance Analysis — The SymTA/S Approach*, IEE Proceedings Computers and Digital Techniques 152 (2005), no. 2, 148–166.

Formal Methods

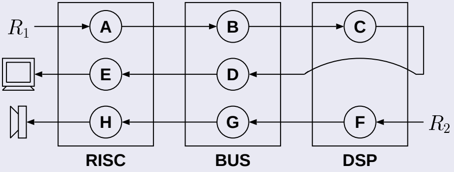
Example: MPEG-2 on Multiprocessor Platform



stream	task	function
video	A	VLD, IQ, IS
	B	data transfer
	C	IDCT, MC
	D	data transfer
	E	assemble video-frames
audio	F	DEC, IMDCT, SYN
	G	data transfer
	H	assemble audio-frames

Formal Methods

Example: MPEG-2 on Multiprocessor Platform



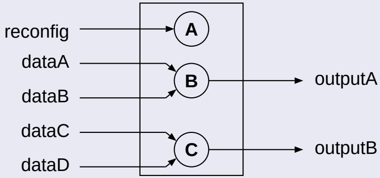
stream	task	function
video	A	VLD, IQ, IS
	B	data transfer
	C	IDCT, MC
	D	data transfer
	E	assemble video-frames
audio	F	DEC, IMDCT, SYN
	G	data transfer
	H	assemble audio-frames

But ...

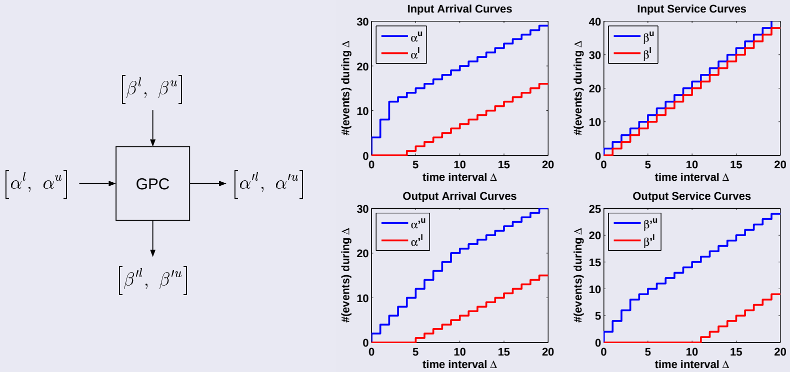
- Tasks have usually more than one input.
- The activation of tasks can depend on complex activation schemes.

Task Model


```
if test(reconfig) then                                   ▷ execute subtask A
    execute code that reconfigures the task;
else if test(dataA) or test(dataB) then                  ▷ execute subtask B
    process first event arrived at dataA or dataB;
    write to outputA;
else if test(dataC) and test(dataD) then                 ▷ execute subtask C
    process first event in dataC and in dataD;
    write to outputB;
end if
```



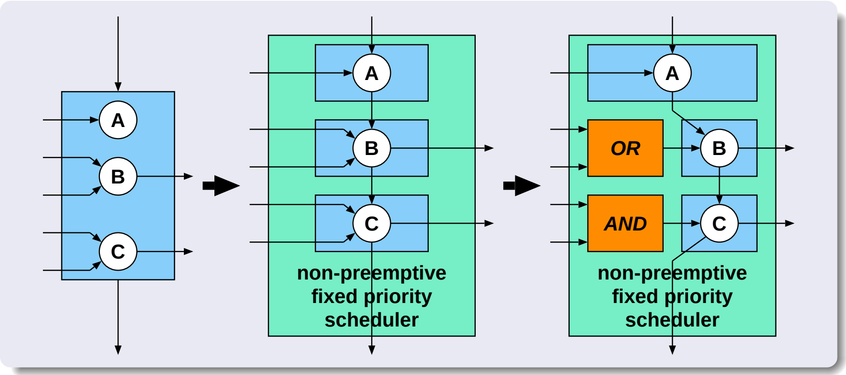
Greedy Processing Component



arrival curve α : $\alpha^l(t-s) \leq R(t) - R(s) \leq \alpha^u(t-s)$,

service curve β : $\beta^l(t-s) \leq C(t) - C(s) \leq \beta^u(t-s)$, $\quad \forall t-s \geq 0$

Analysis Principle



Non-Preemptive Fixed Priority Scheduling

Related Work: Priority Queuing in Network Queueing Theory

- Jean-Yves Le Boudec and Patrick Thiran, *Network Calculus — A Theory of Deterministic Queuing Systems for the Internet*, Lecture Notes in Computer Science, vol. 2050, Springer Verlag, 2001.
- Jens Schmitt, *On Average and Worst Case Behavior in Non-Preemptive Priority Queueing*, Proc. 2003 Intl Symp. on Performance Evaluation of Computer and Telecommunication Systems, 2003, pp. 197–204.

Non-Preemptive Fixed Priority Scheduling

Relations for Preemptive Fixed Priority Scheduling

$$\beta_i^u(\Delta) = \inf_{\lambda \geq 0} \left\{ \beta^u(\Delta + \lambda) - \sum_{j=i+1}^{N} \alpha_j^l(\Delta + \lambda) \right\}$$

$$\beta_i^l(\Delta) = \sup_{0 \leq \lambda \leq \Delta} \left\{ \beta^l(\Delta - \lambda) - \sum_{j=i+1}^{N} \alpha_j^u(\Delta - \lambda) \right\}$$

$i \ldots$ task priority $\qquad N \ldots$ number of tasks

Non-Preemptive Fixed Priority Scheduling

Relations for Non-Preemptive Fixed Priority Scheduling

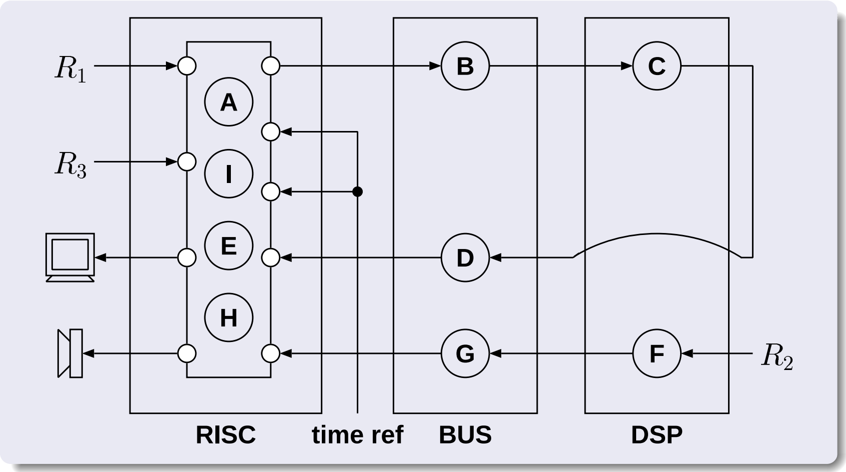
$$\tilde{\beta}_i^u(\Delta) = \min\left\{\beta^u(\Delta), \inf_{\lambda \geq 0}\left\{\beta^u(\Delta+\lambda) - \sum_{j=i+1}^{N} \alpha_j^l(\Delta+\lambda)\right\} + C_i^{max}\right\}$$

$$\tilde{\beta}_i^l(\Delta) = \max\left\{0, \sup_{0 \leq \lambda \leq \Delta}\left\{\beta^l(\Delta-\lambda) - \sum_{j=i+1}^{N} \alpha_j^u(\Delta-\lambda)\right\} - \max_{1 \leq j < i}\left\{C_j^{max}\right\}\right\}$$

$i \ldots$ task priority $\quad$ $N \ldots$ number of tasks

$C_i^{max} \ldots$ maximum number of resource units to process one event

MPEG-2 Case Study



Contributions

- Consideration of complex task activation schemes based on AND/OR semantics
- Modeling of tasks with complex activation schemes in MPA using abstract AND/OR components and non-preemptive fixed priority scheduling
- Derivation and proof of input-output relations of abstract AND/OR component
- Derivation and proof of relations to model non-preemptive fixed priority scheduling
- Application of the results in an MPEG-2 case study

Real-Time Interfaces

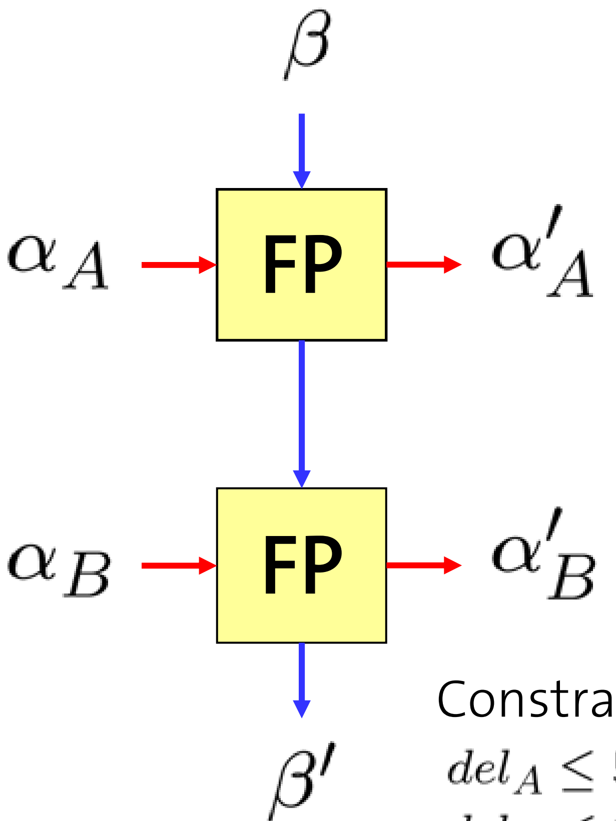
ETH Zurich, Switzerland

Outline

- Real-Time Interfaces / Interface-Based Design
- IBD Case Study

Real-Time Interfaces & Interface-Based Design

Component-Based Design



Constraints:

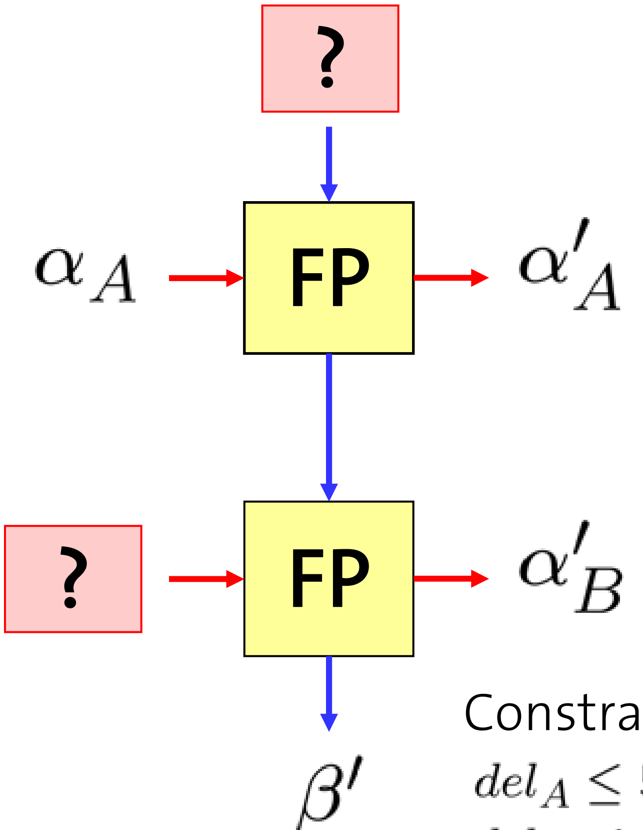
$del_A \leq 5ms$

$del_B \leq 8ms$

Schedulable?

1. ***Design***
2. ***Analysis***
   - Given: ***all*** components, their interconnections structure and all inputs from environment
   - Question: do the components ***work together properly***?

Interface-Based Design



Constraints:

$del_A \leq 5ms$

$del_B \leq 8ms$

1. ***Design and Composition***
   - Given: ***some*** components, their interconnection structure and some inputs from environment
   - Questions: Is there the chance that the components ***work together properly***? What are the ***assumptions*** towards the environment? How can I ***change the environment*** such that the components still work together?

Interface-Based Design

Component Description:

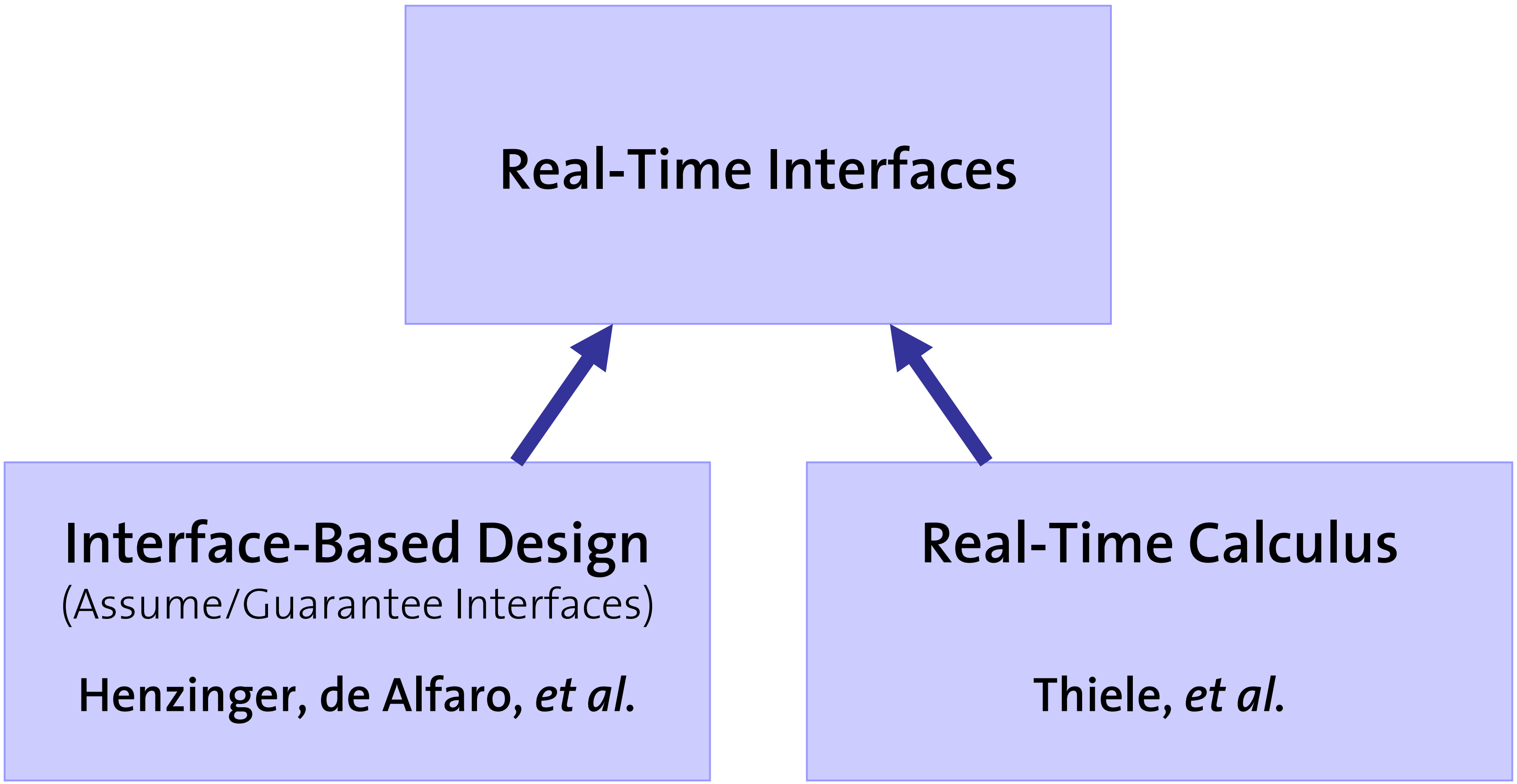
What Does a Component Do?

vs.

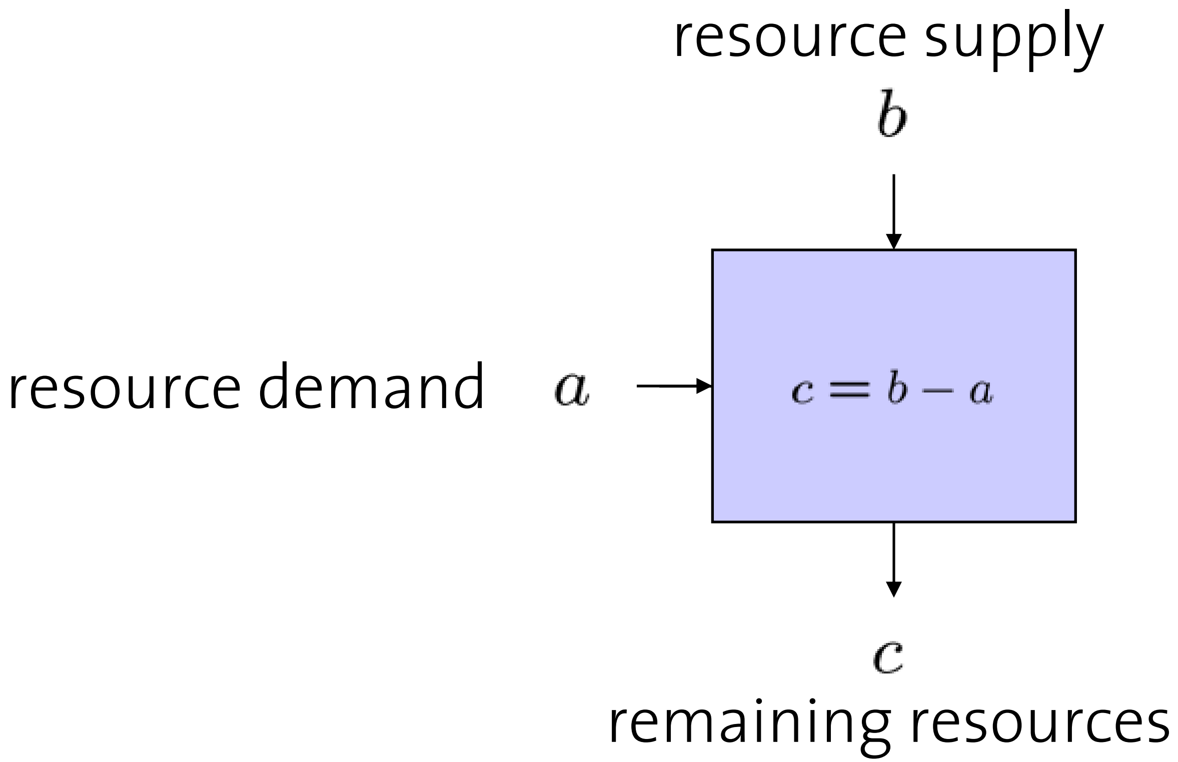
Component Interface:

How Can a Component Be Used?

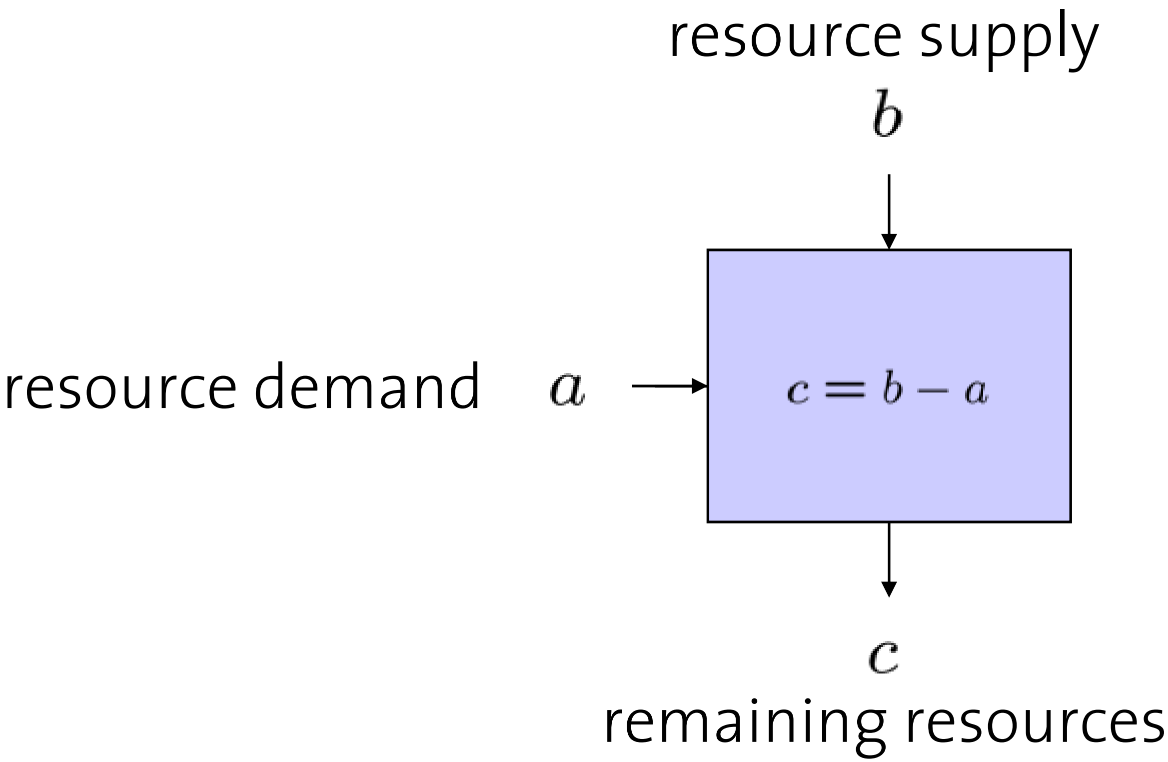
Real-Time Interfaces



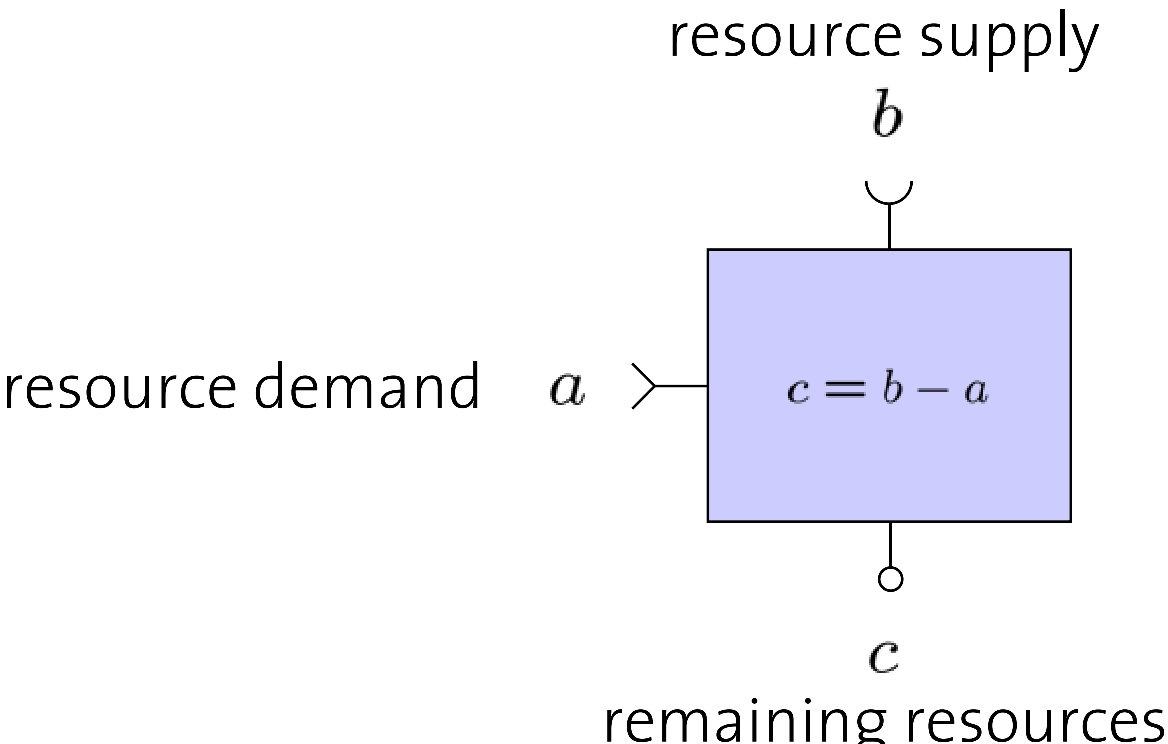
From a Simple Component …



... to its A/G Interface

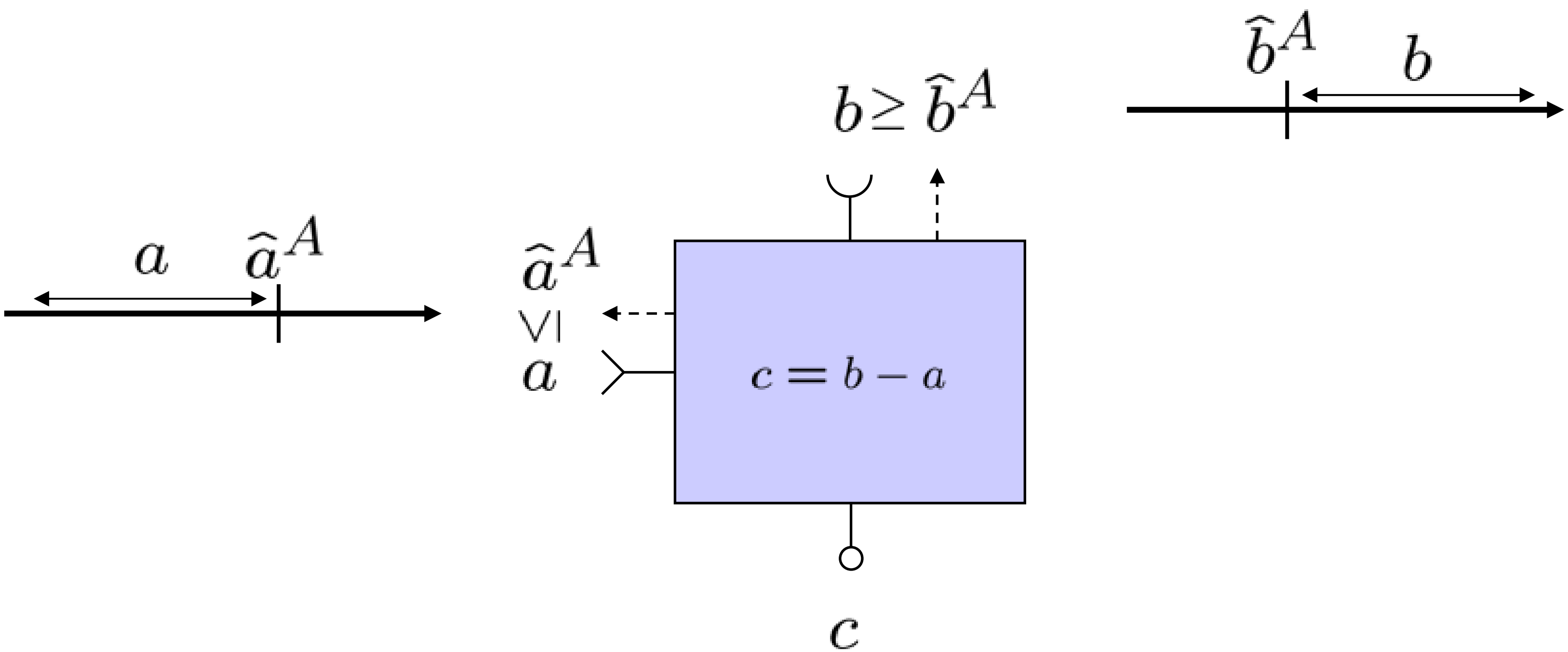


… to its A/G Interface



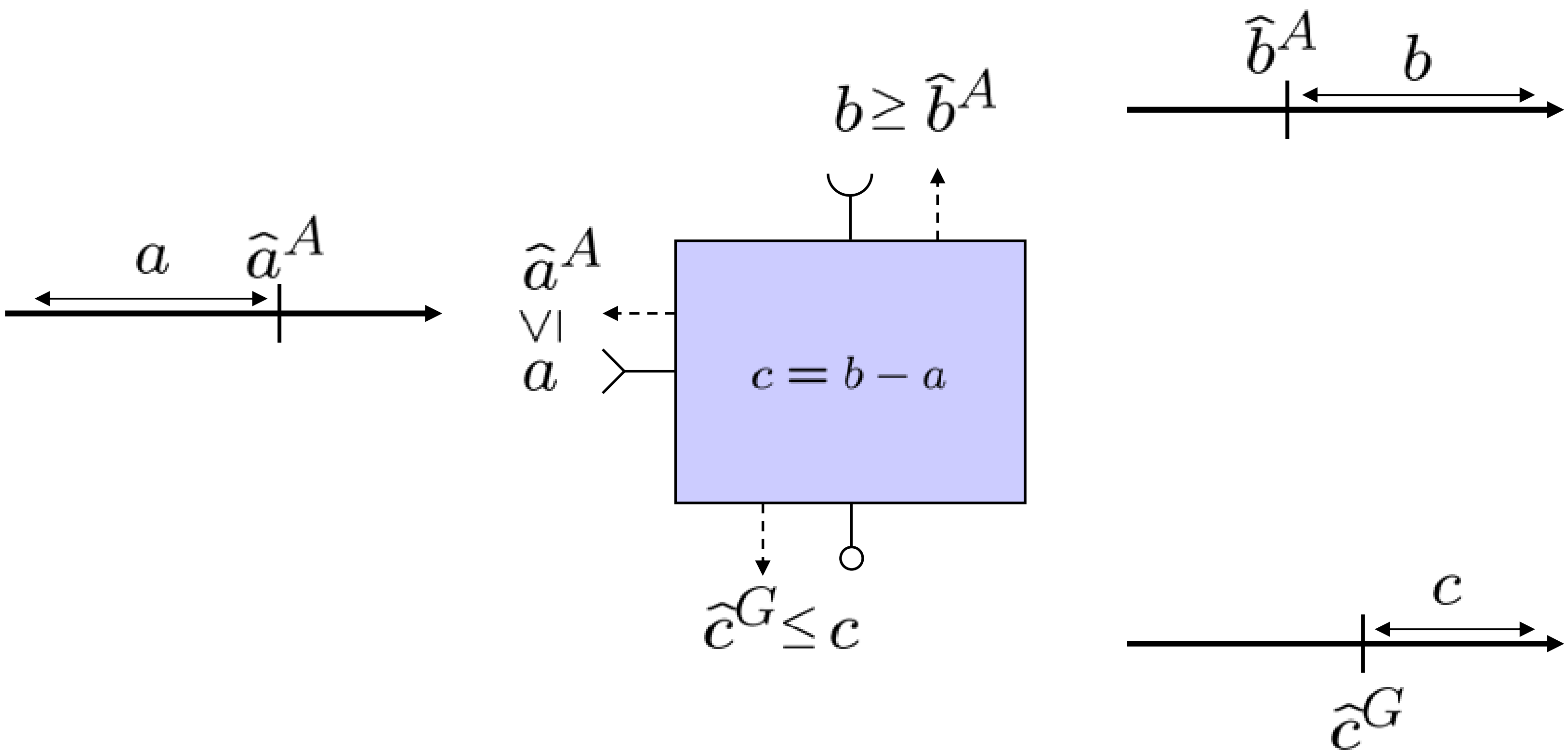
Variables & Types

... to its A/G Interface



Input Assumptions, Predicates & Values

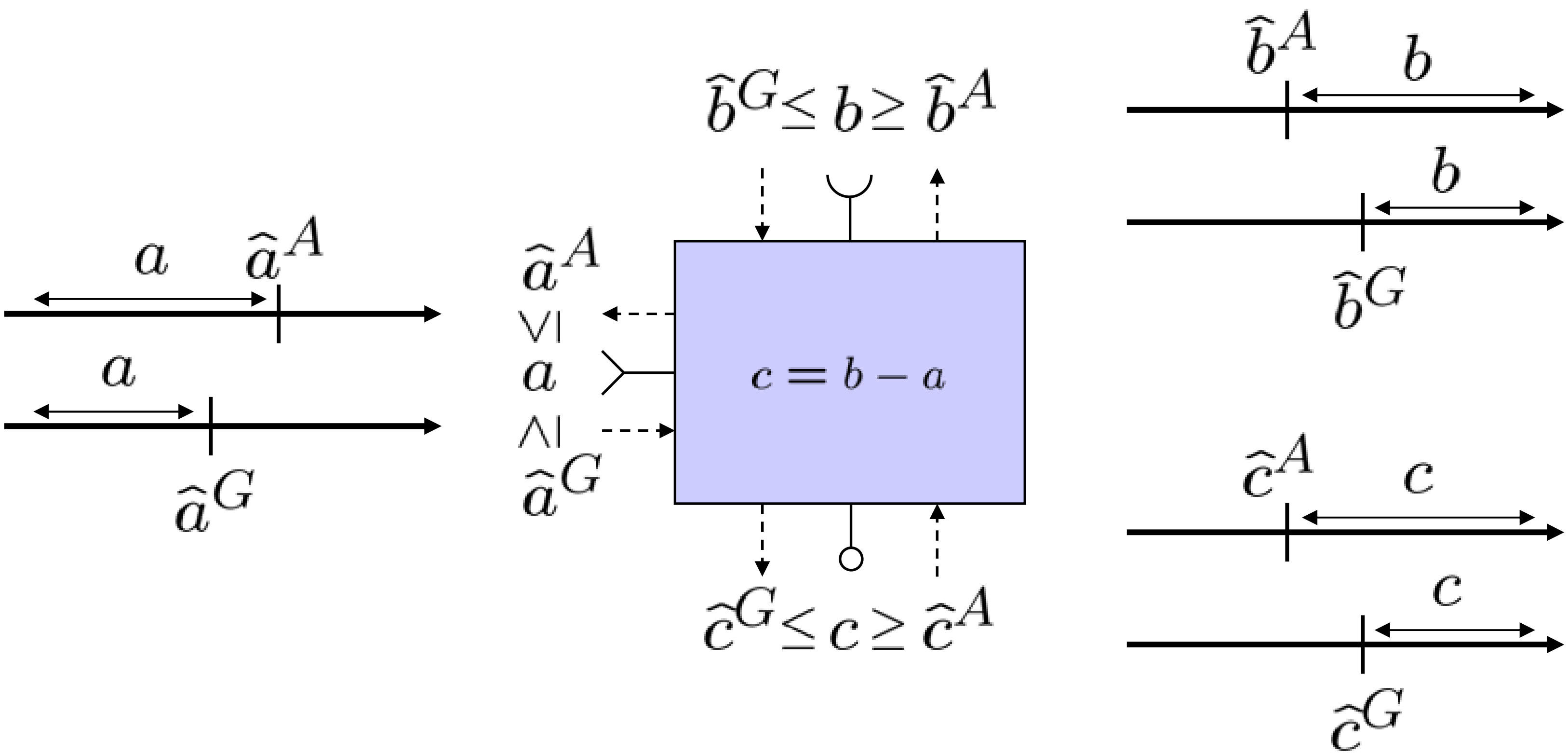
... to its A/G Interface



Output Guarantees, Predicates & Values

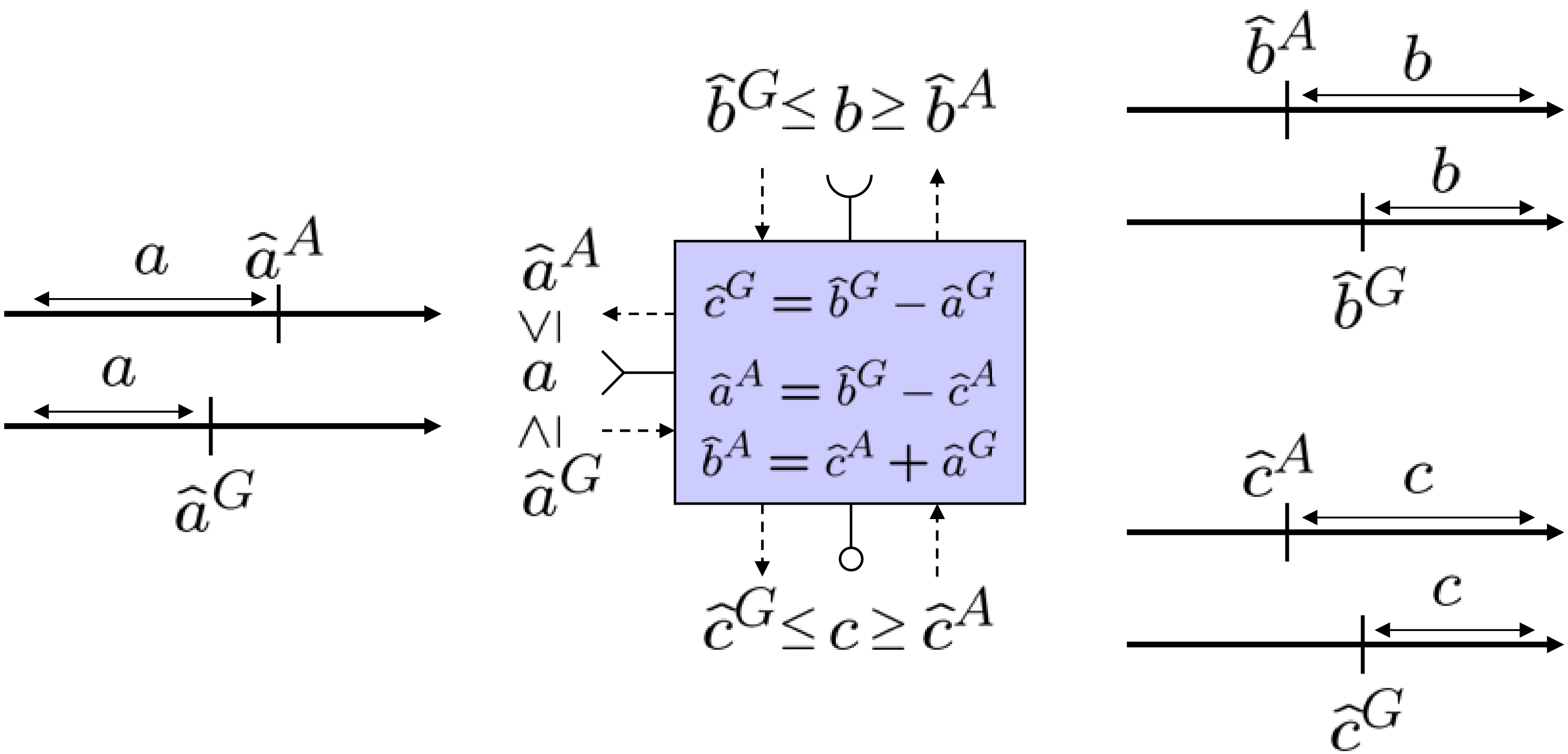
$$(a \leq \widehat{a}^A) \wedge (b \geq \widehat{b}^A) \wedge (\widehat{c}^G \leq c)$$

... to its A/G Interface



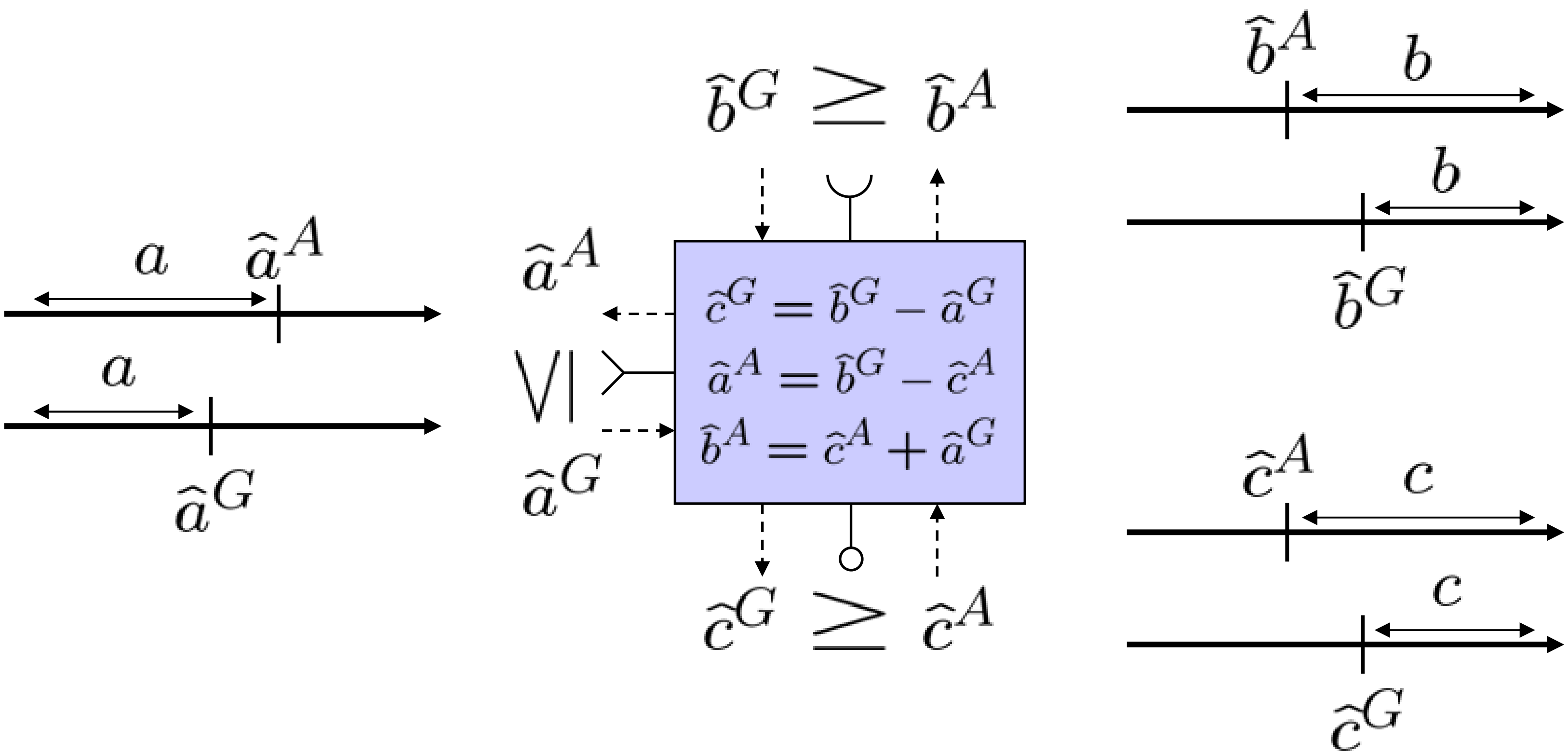
Data from Other Component Interfaces

... to its A/G Interface



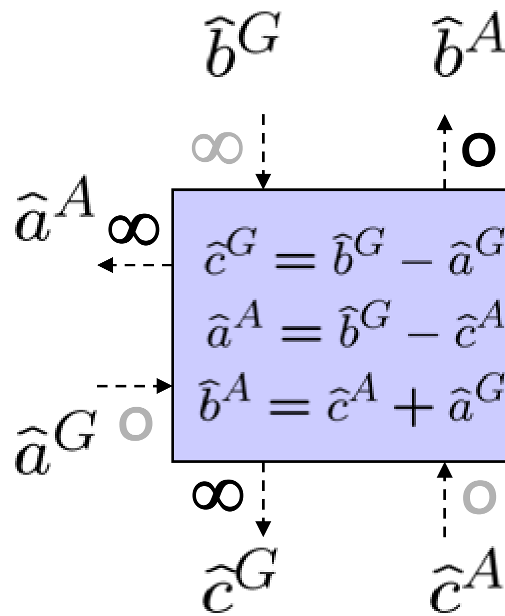
Internal Interface Relations

Compatibility & Composition

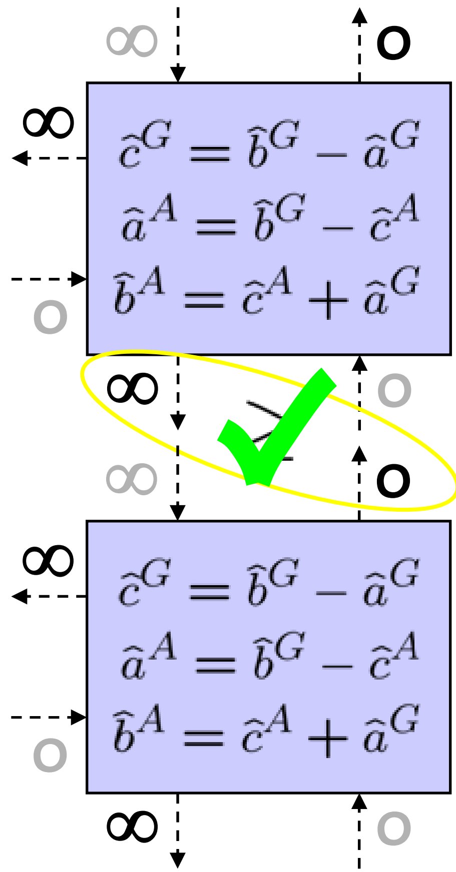


$$(\hat{a}^G \leq \hat{a}^A) \wedge (\hat{b}^G \geq \hat{b}^A) \wedge (\hat{c}^A \leq \hat{c}^G)$$

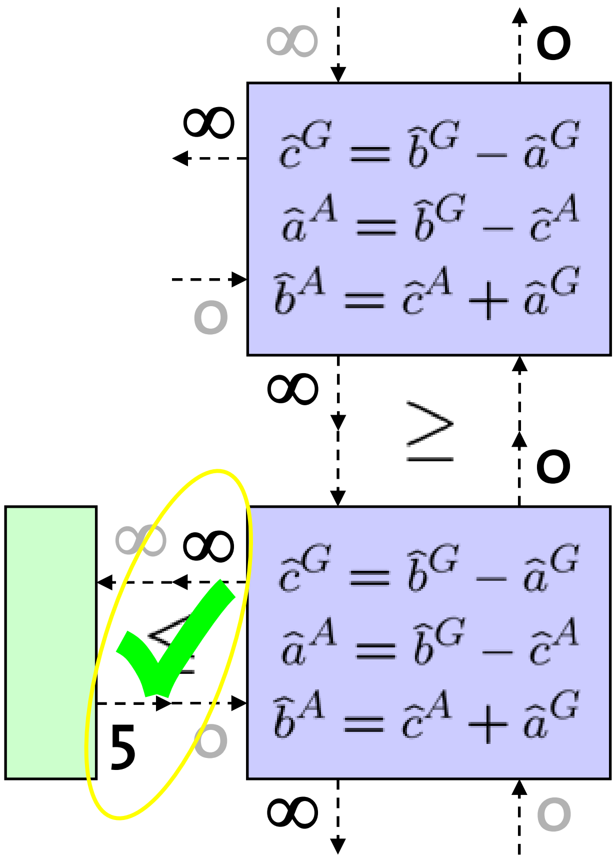
The Weakest Environment



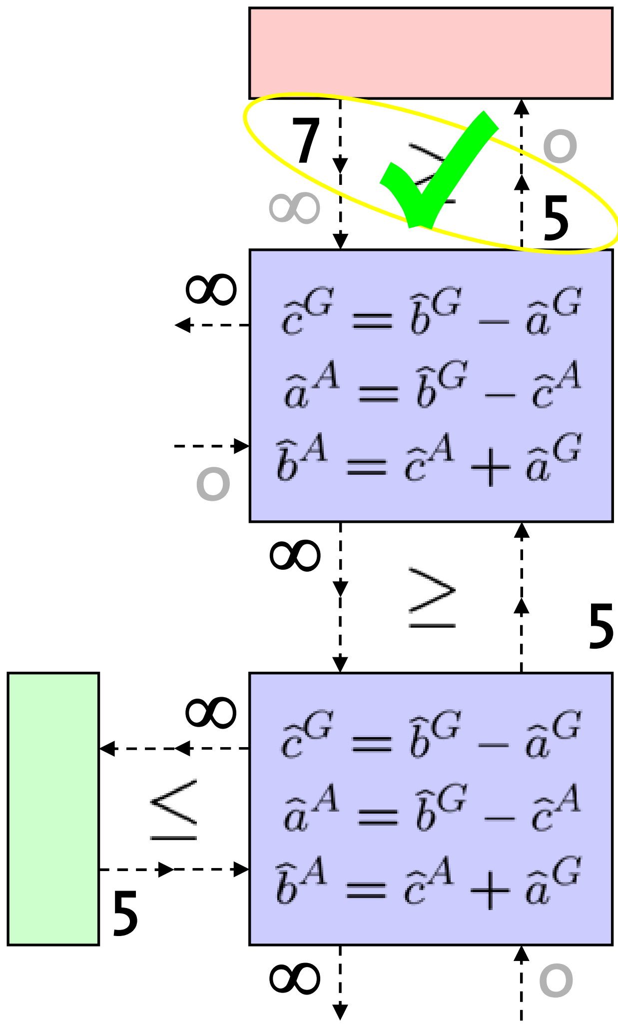
A Simple Example



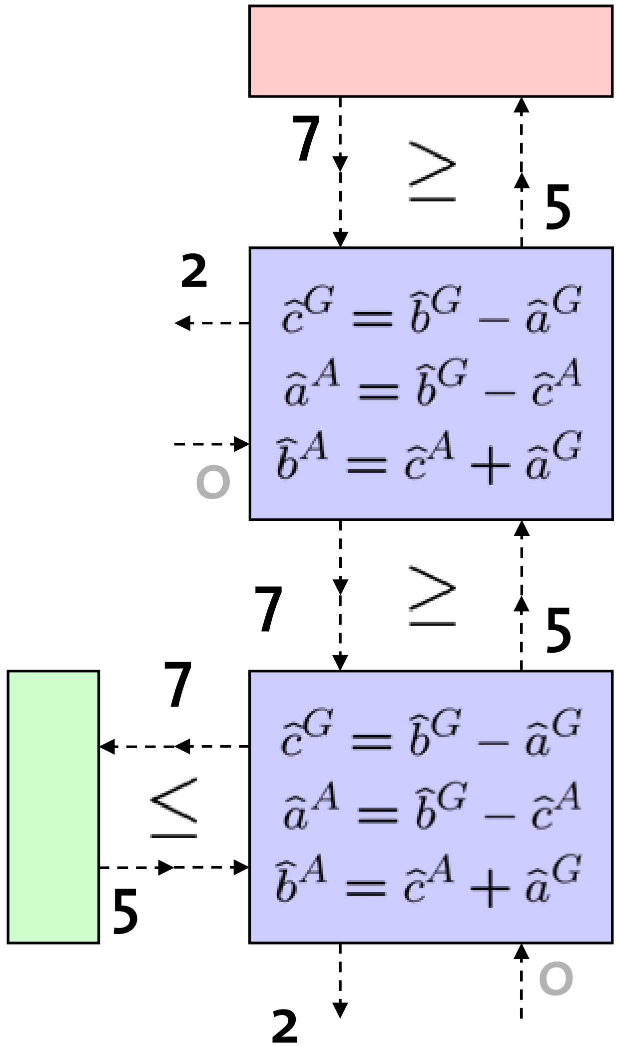
A Simple Example



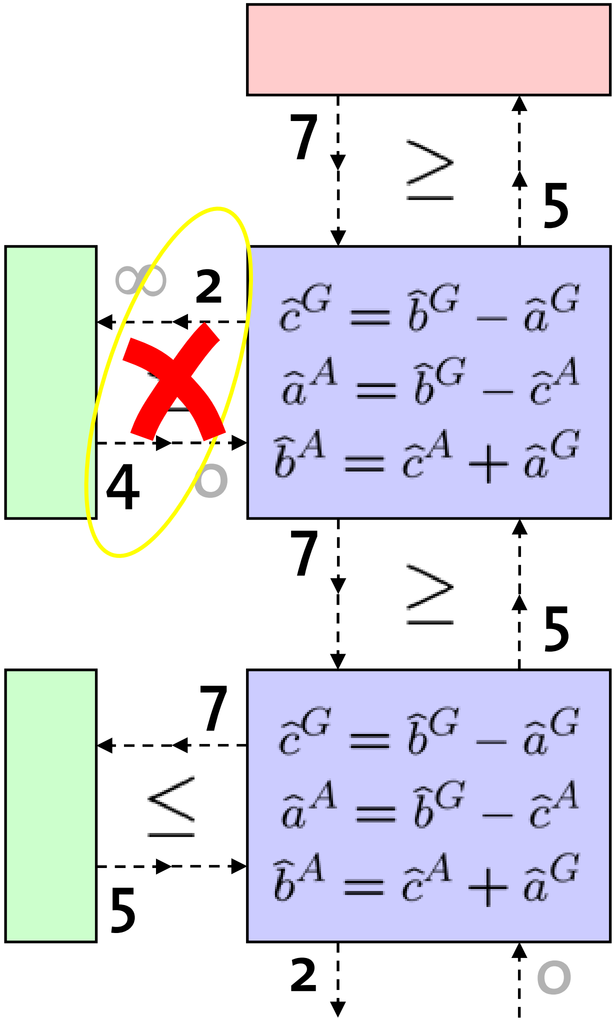
A Simple Example



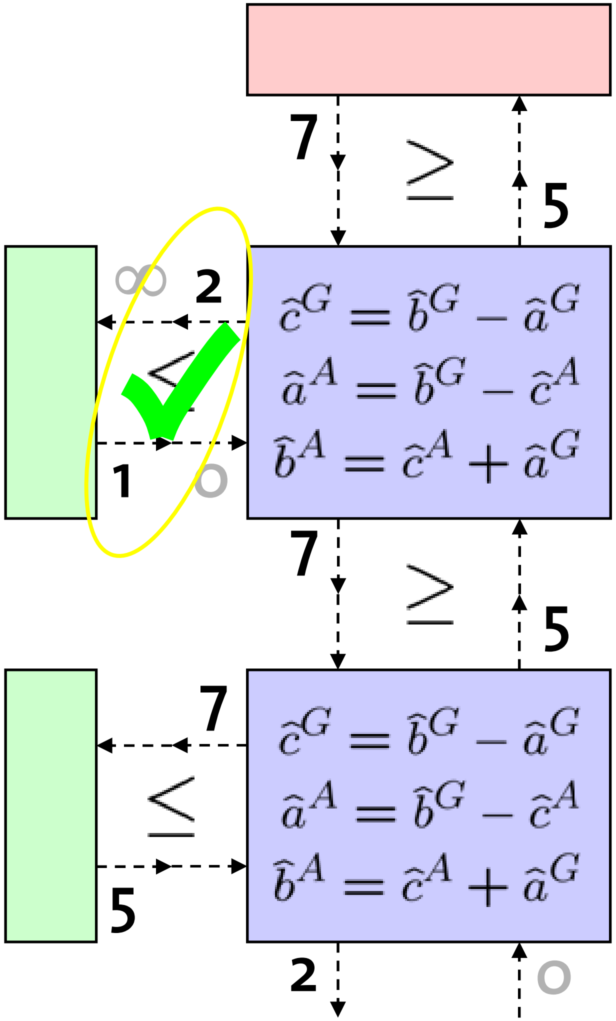
A Simple Example



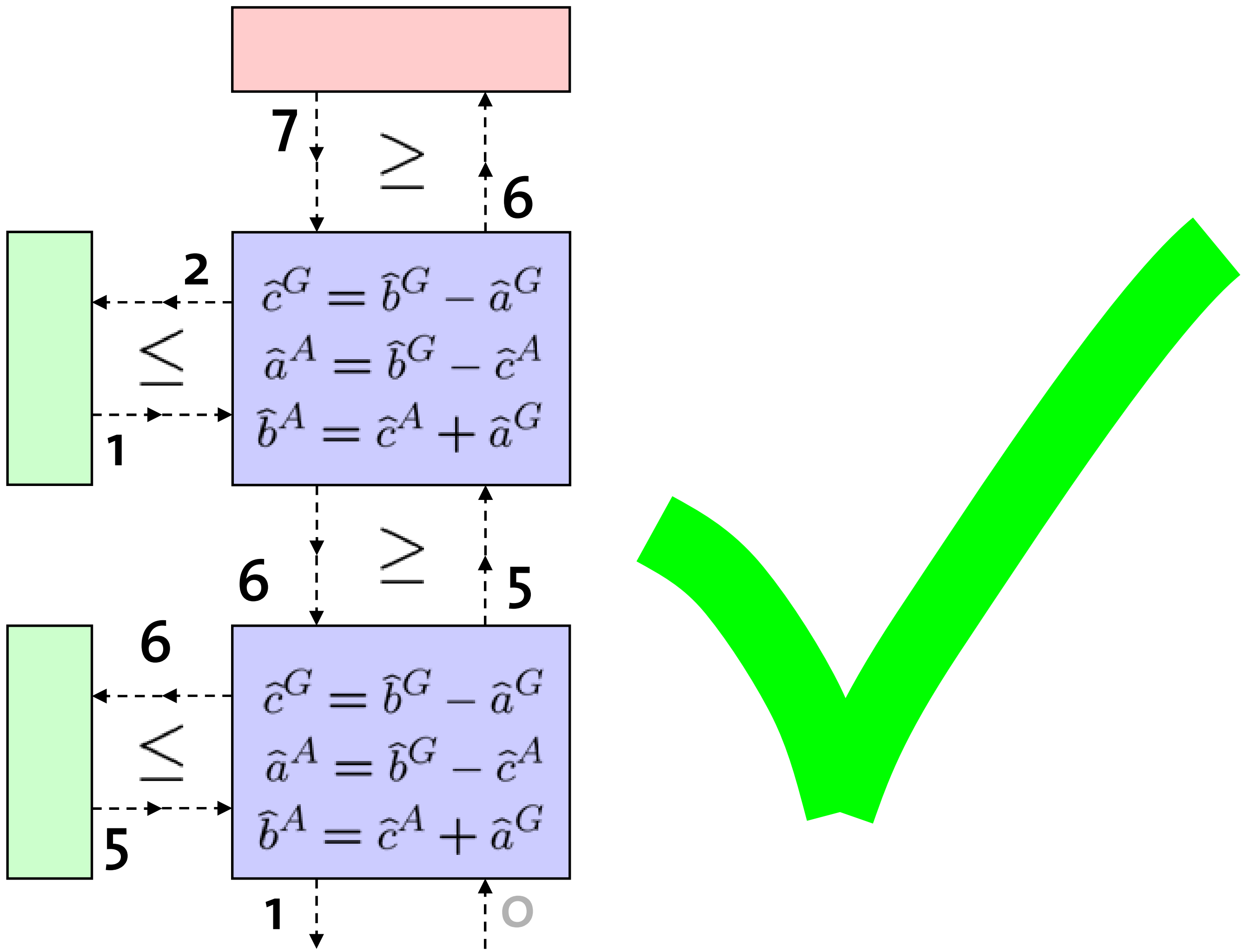
A Simple Example



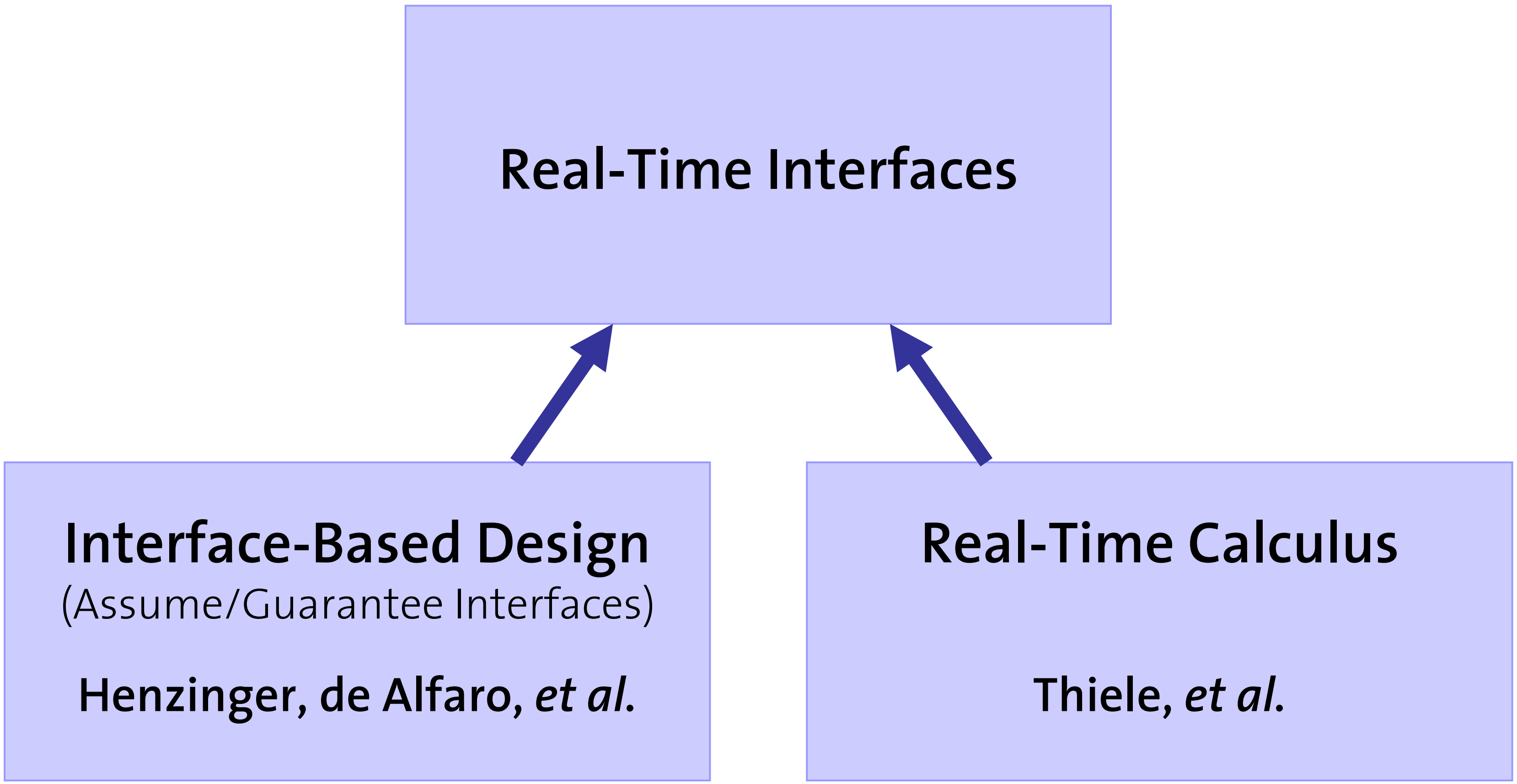
A Simple Example



A Simple Example



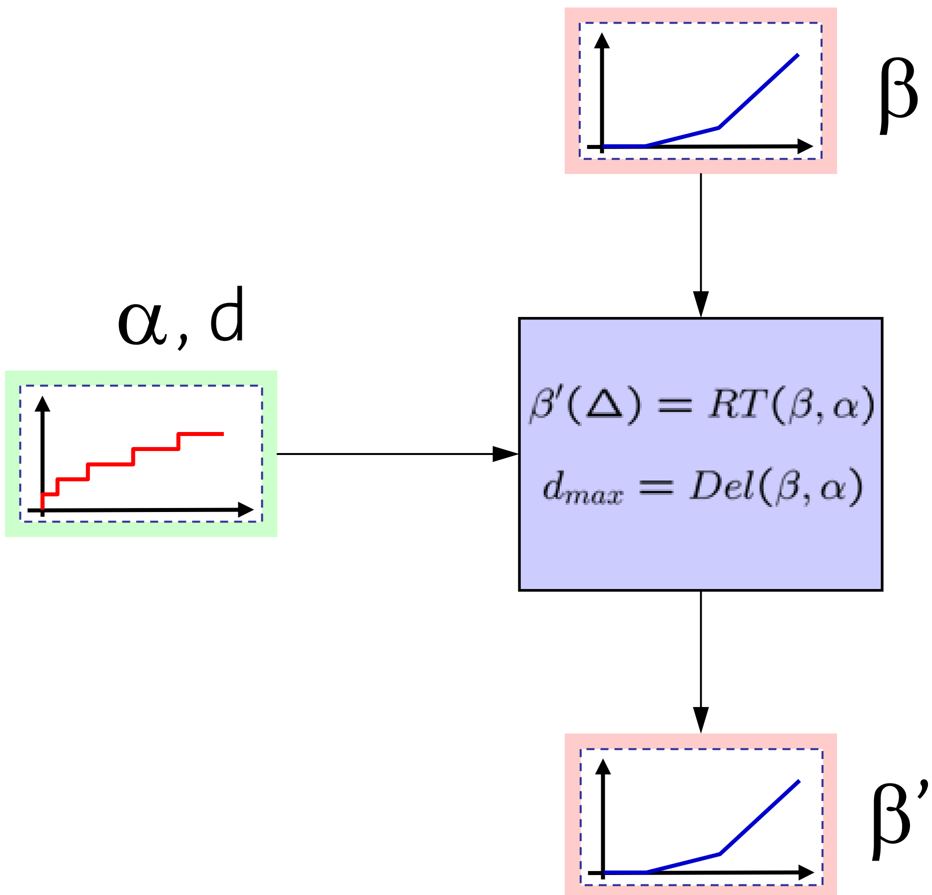
Foundations of Real-Time Interfaces



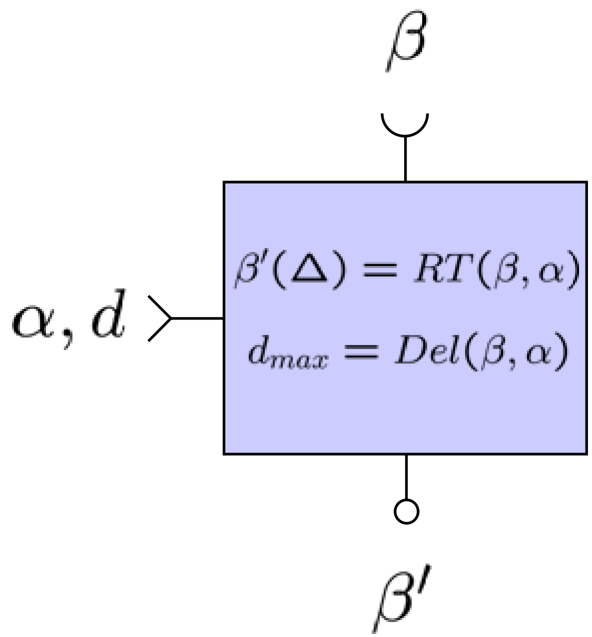
Three Steps to Real-Time Interfaces

- Step 1: Abstract Components
- Step 2: Interface Variables and Predicates
- Step 3: Internal Interface Relations

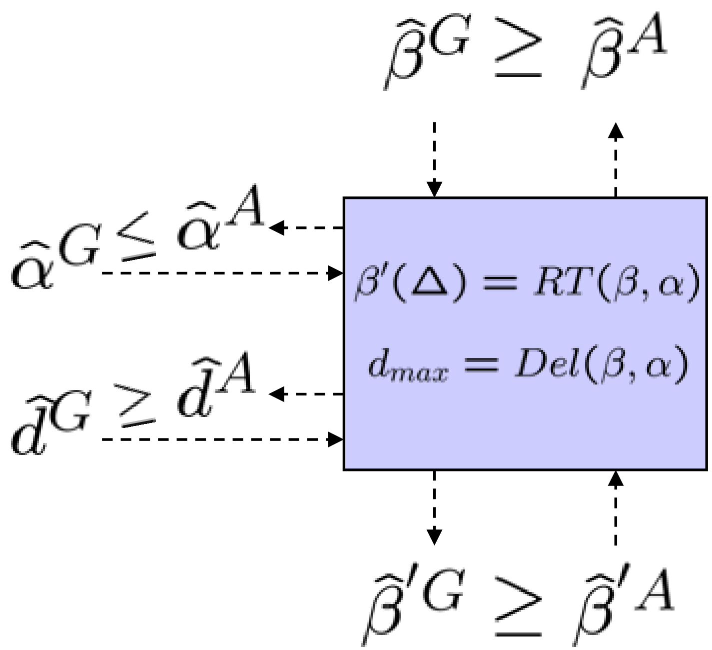
Step 1: Abstract Component



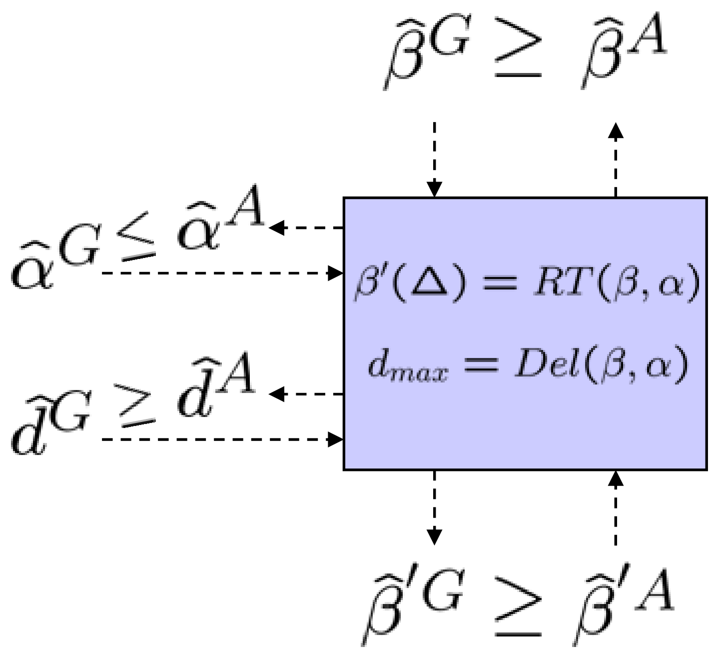
Step 2: Interface Variable & Predicates



Step 2: Interface Variable & Predicates



Step 3: Internal Interface Relations



Fixed Priority Scheduling

$$
\begin{aligned}
\hat{\beta}'^G &= RT(\hat{\beta}^G, \hat{\alpha}^G) \\
\hat{\beta}^A &= \max\left\{\hat{\alpha}^G(\Delta - \hat{d}^G), RT^{-\beta}(\hat{\beta}'^A, \hat{\alpha}^G)\right\} \\
\hat{\alpha}^A &= \min\left\{\hat{\beta}^G(\Delta + \hat{d}^G), RT^{-\alpha}(\hat{\beta}'^A, \hat{\beta}^G)\right\} \\
\hat{d}^A &= Del(\hat{\beta}^G, \hat{\alpha}^G)
\end{aligned}
$$

Applications

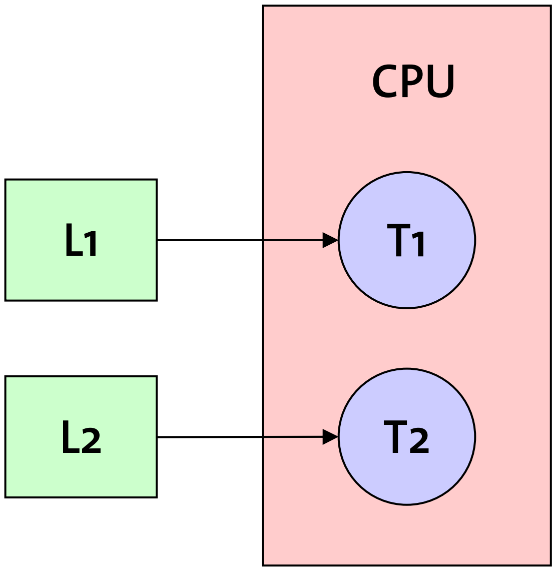
- *Interface-Based Design of RT Systems*
  - Find minimum processor speed for complex systems with mixed hierarchical scheduling.
  - Find optimal TDMA slot and cycle length allocations.
  - Specify maximum allowable input stream rates.
  - ...
- Answering of design questions, e.g. resource dimensioning
- On-Line Load Adaptation
- On-Line Service Adaptation
- On-Line Admission Tests
- ...

A Simple System with FP Scheduling...

CPU: Fully Available

Load 1: p = 3, d = 6

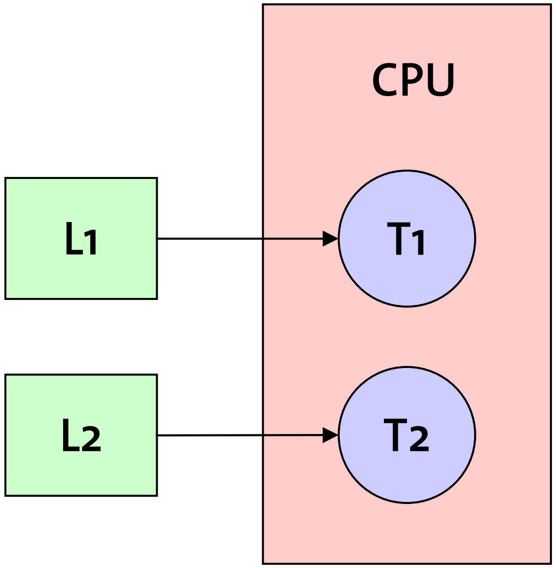
Load 2: p = 6, d = 7

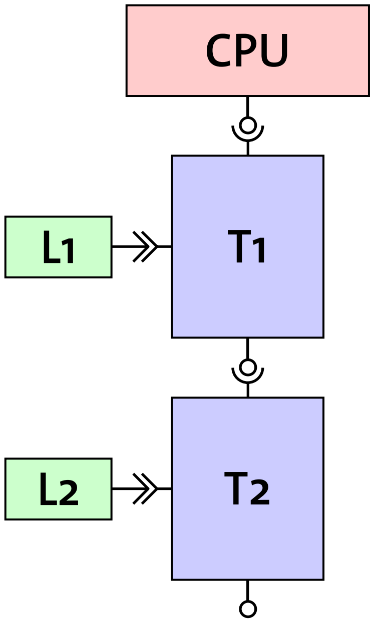


T1: WCET = 1

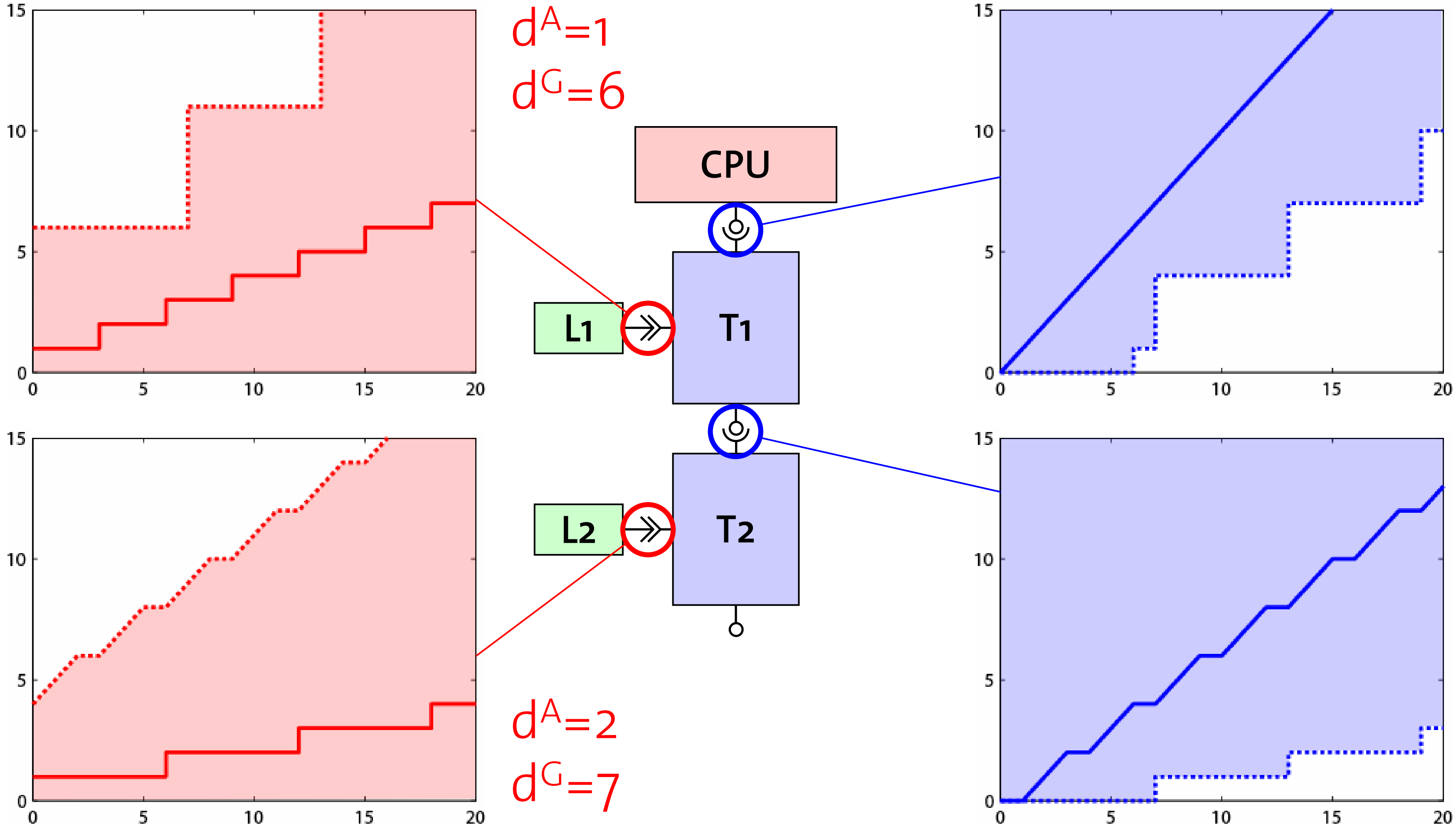
T2: WCET = 1

... and its Real-Time Interface Model





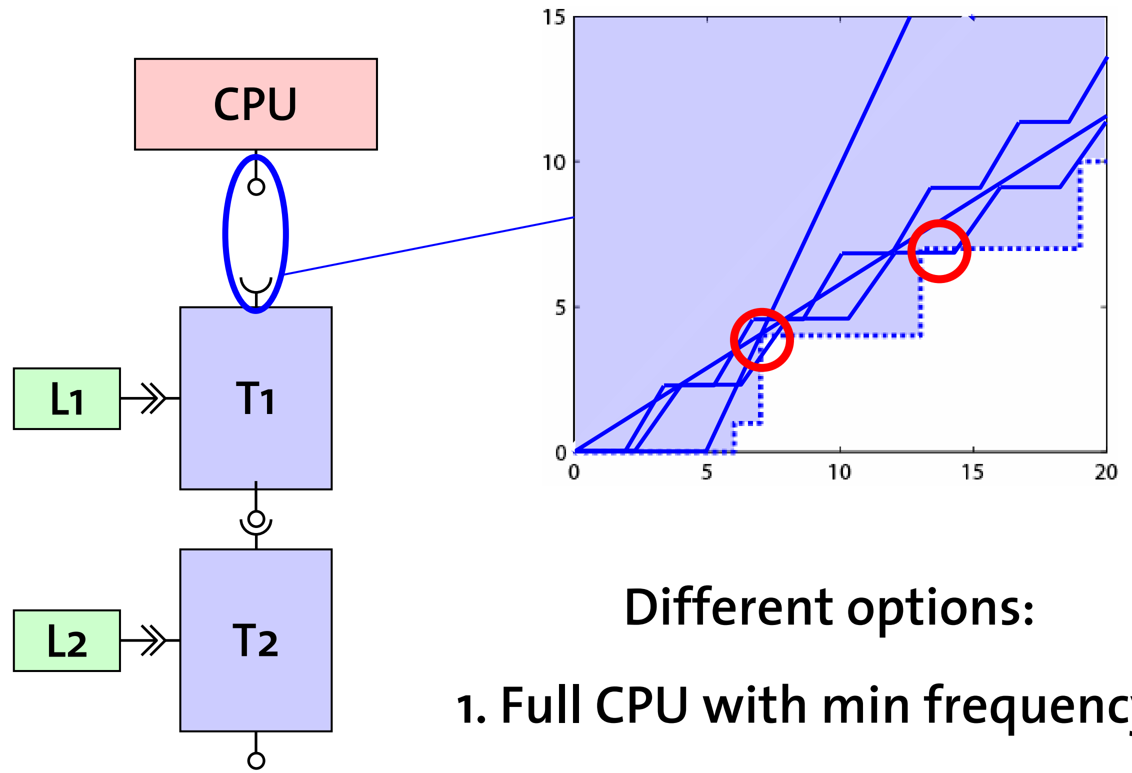
Schedulability Analysis



Applications

- Interface-Based Design of RT Systems
  - Find minimum processor speed for complex systems with mixed hierarchical scheduling.
  - Find optimal TDMA slot and cycle length allocations.
  - Specify maximum allowable input stream rates.
  - ...
- ***Answering of design questions, e.g. resource dimensioning***
- On-Line Load Adaption
- On-Line Service Adaption
- On-Line Admission Tests
- ...

Resource Dimensioning



Different options:

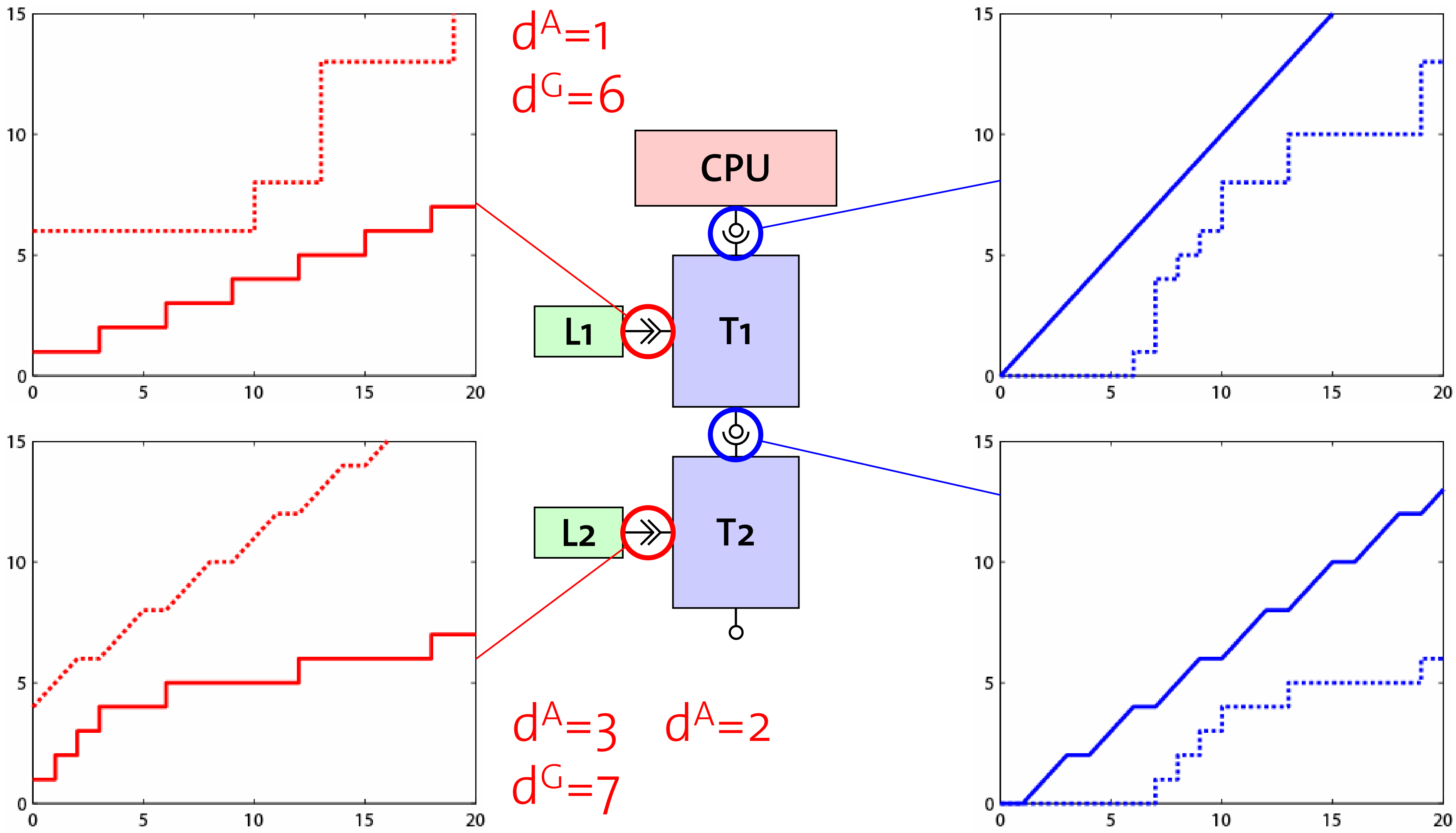
1. Full CPU with min frequency
2. CPU with bounded delay
3. CPU with TDMA, and others

Applications

- Interface-Based Design of RT Systems
  - Find minimum processor speed for complex systems with mixed hierarchical scheduling.
  - Find optimal TDMA slot and cycle length allocations.
  - Specify maximum allowable input stream rates.
  - …
- Answering of design questions, e.g. resource dimensioning
- *On-Line Load Adaption*
- *On-Line Service Adaption*
- *On-Line Admission Tests*
- …

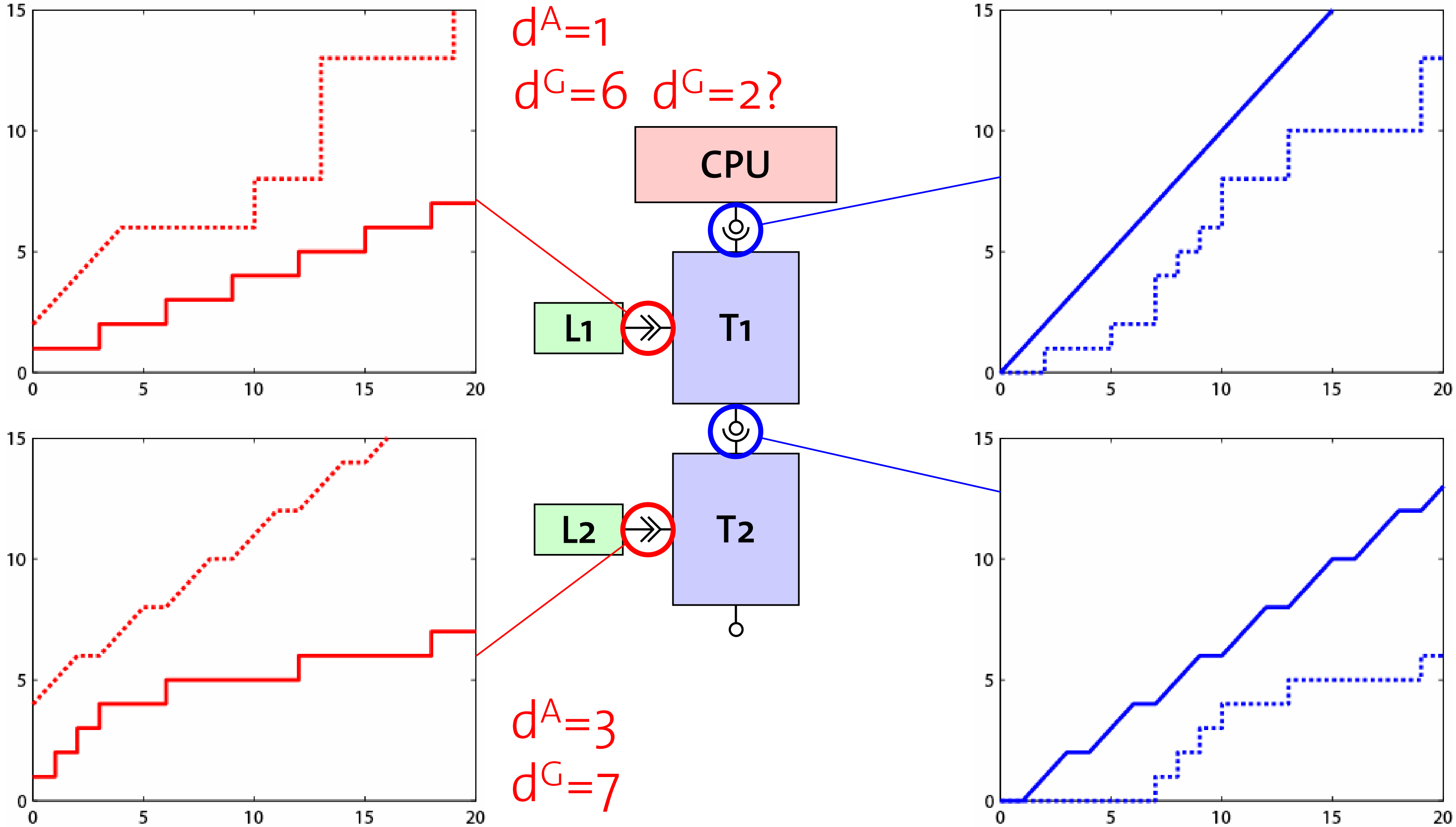
On-Line Load Adaption (Burstiness)

1. Check
2. Admit
3. Update



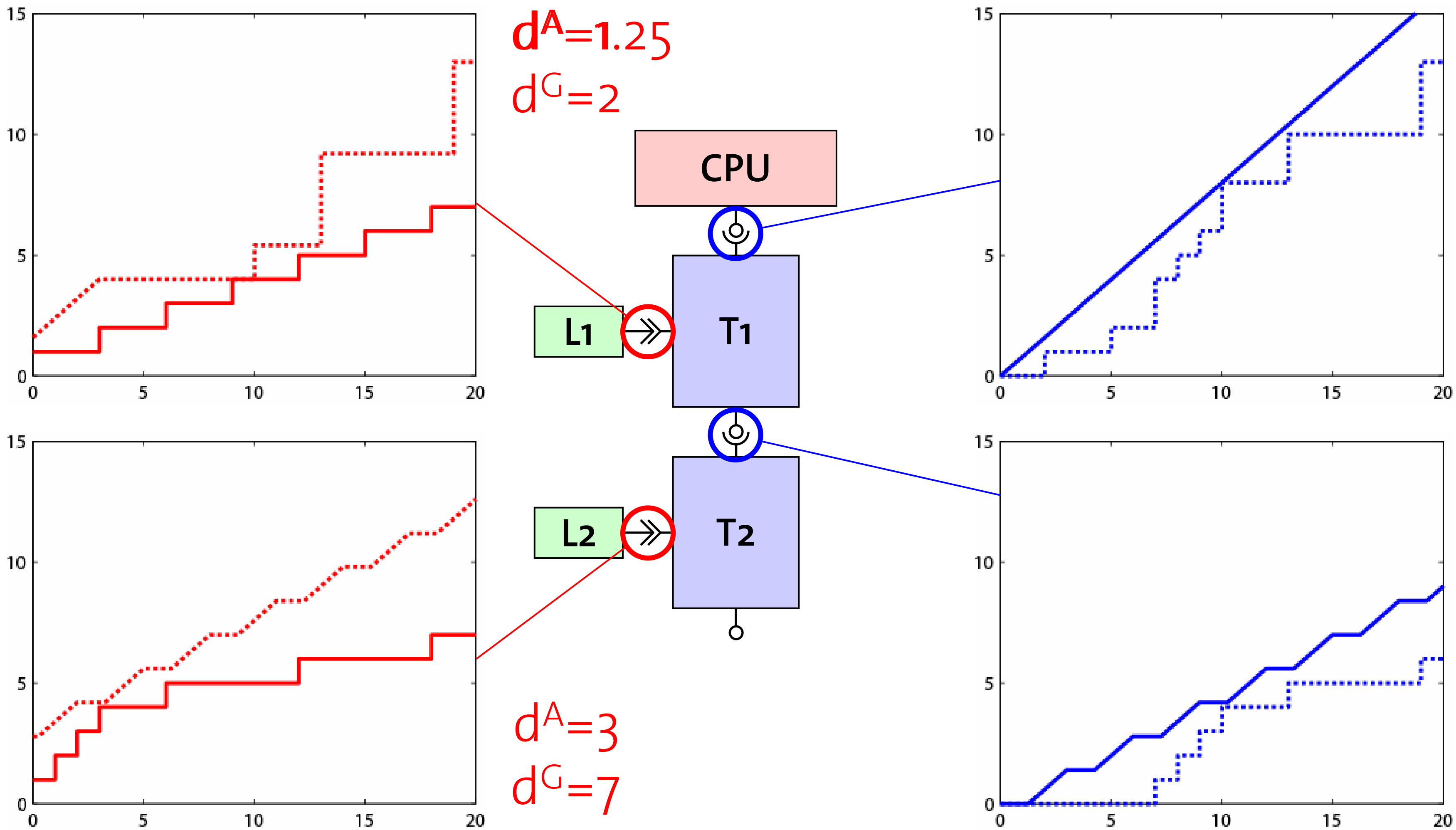
On-Line Load Adaption (Delay)

1. Check
2. Admit
3. Update



On-Line Service Adaption

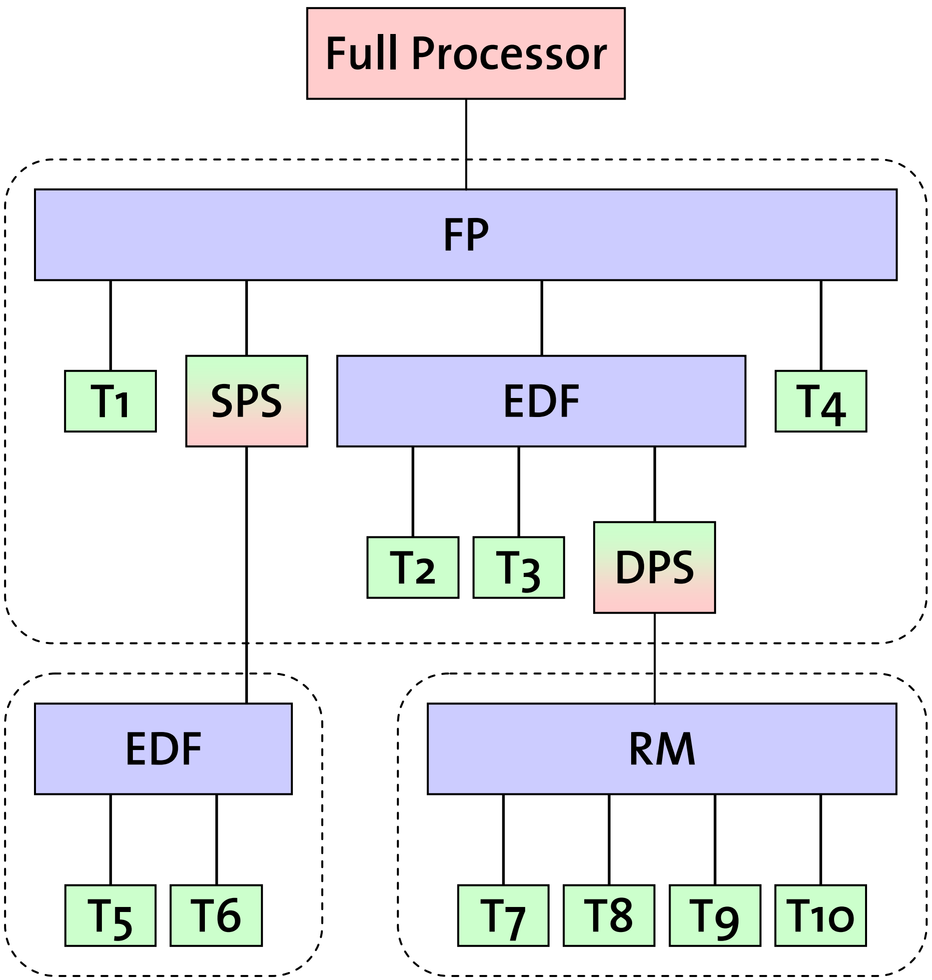
1. Check
2. Admit
3. Update



Outline

- Real-Time Interfaces / Interface-Based Design
- IBD Case Study

A System with Complex Scheduling...



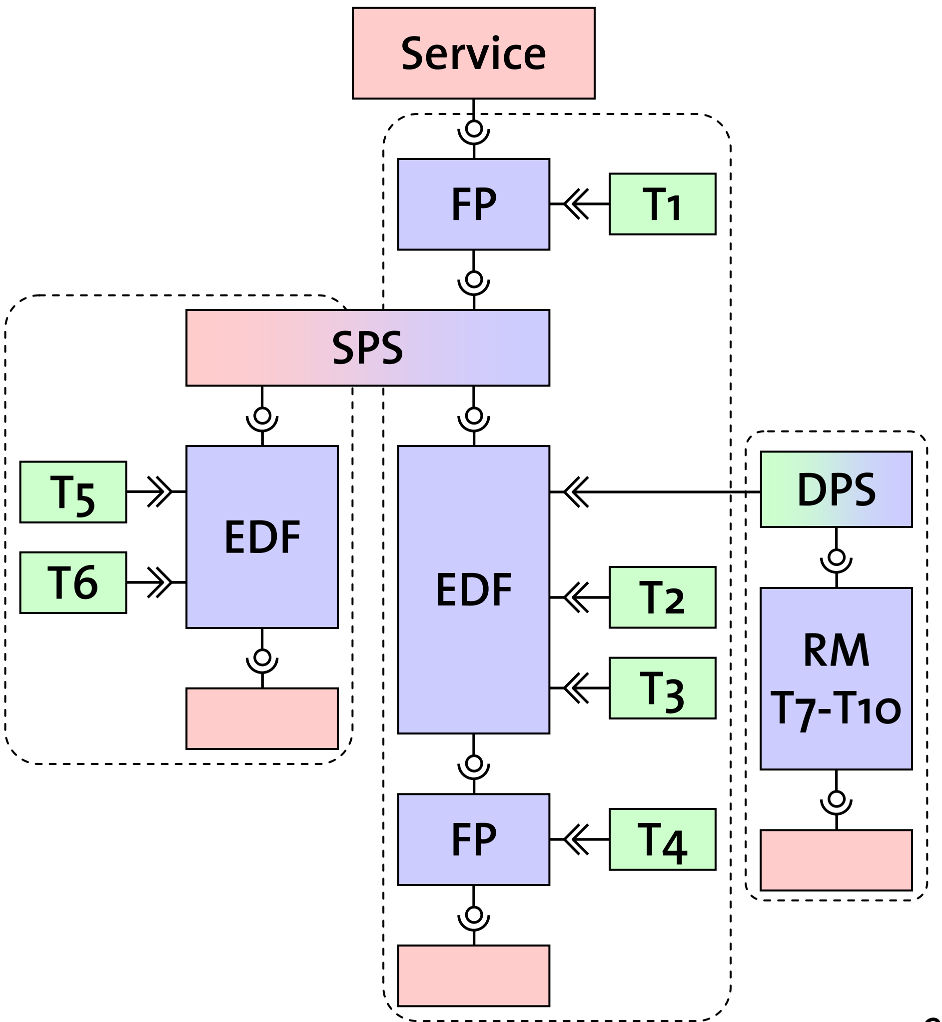
10 Tasks
- with jitter
- with bursts
- deadline = period
- deadline < period
- deadline > period

3 Scheduling Policies
- Rate Monotonic
- Earliest Deadline First
- Fixed Priority

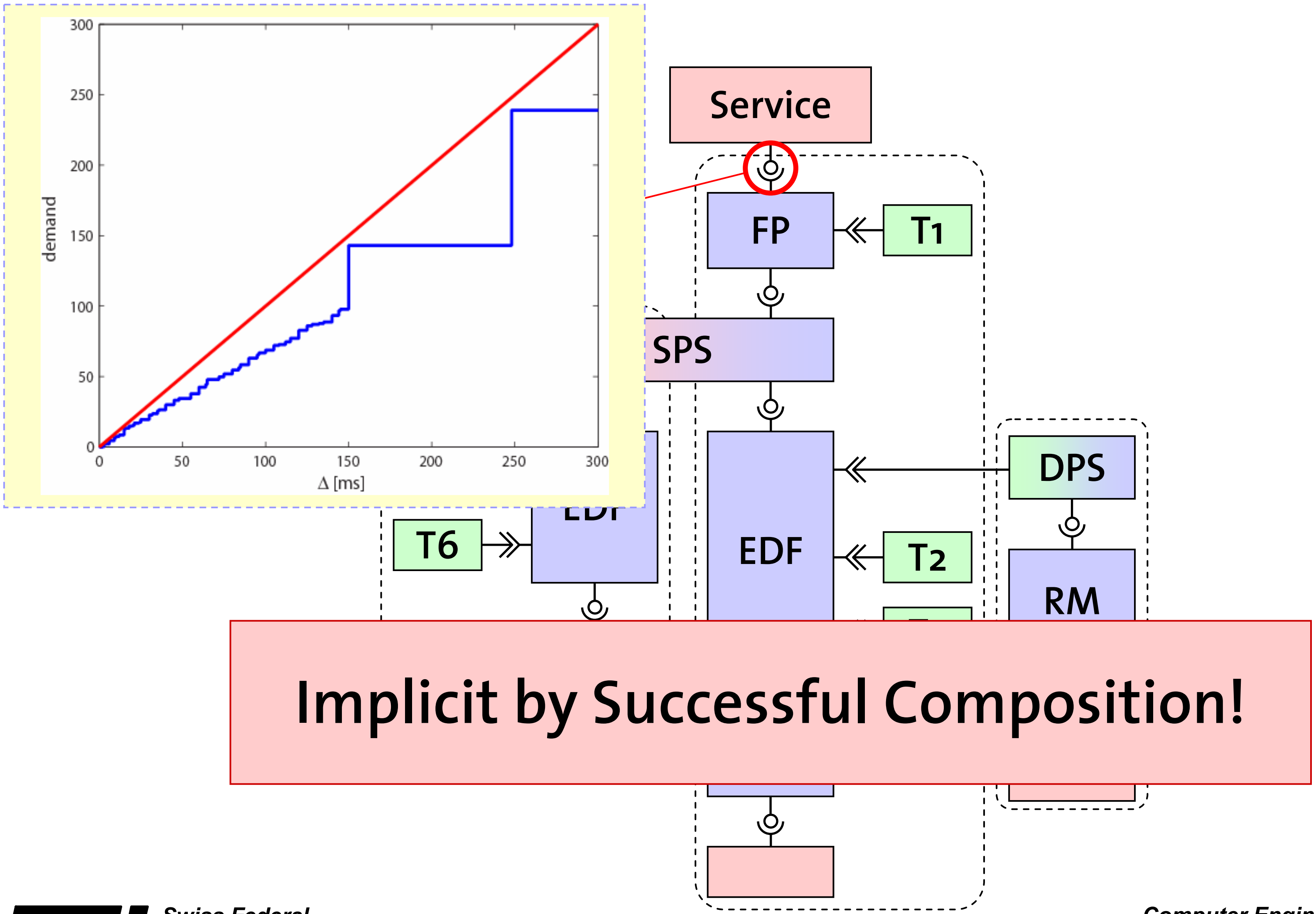
Hierarchical Scheduling
Static & Dynamic Polling Servers

Total Utilization: 98.5%

... and its Real-Time Interface Model

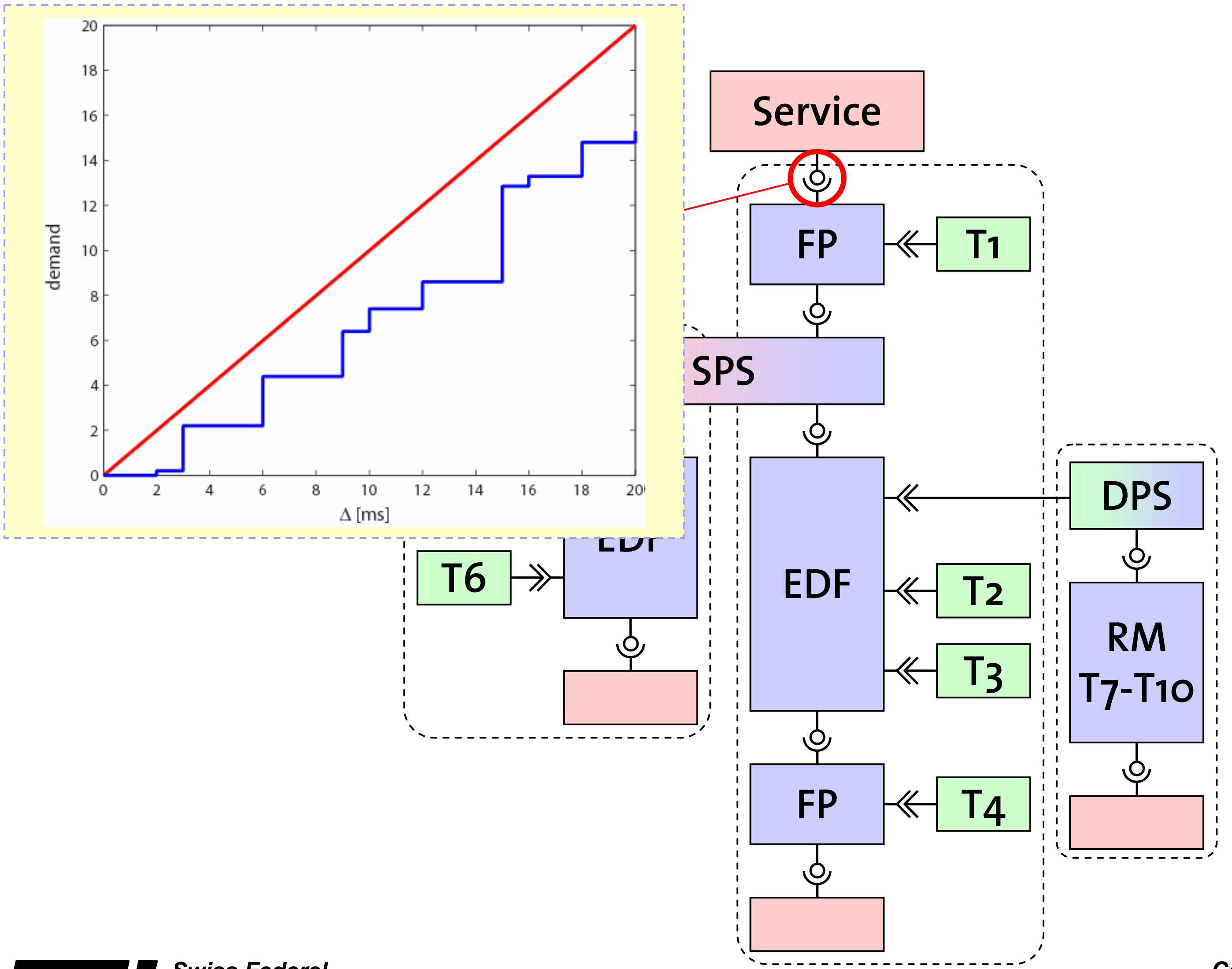


Schedulability Analysis

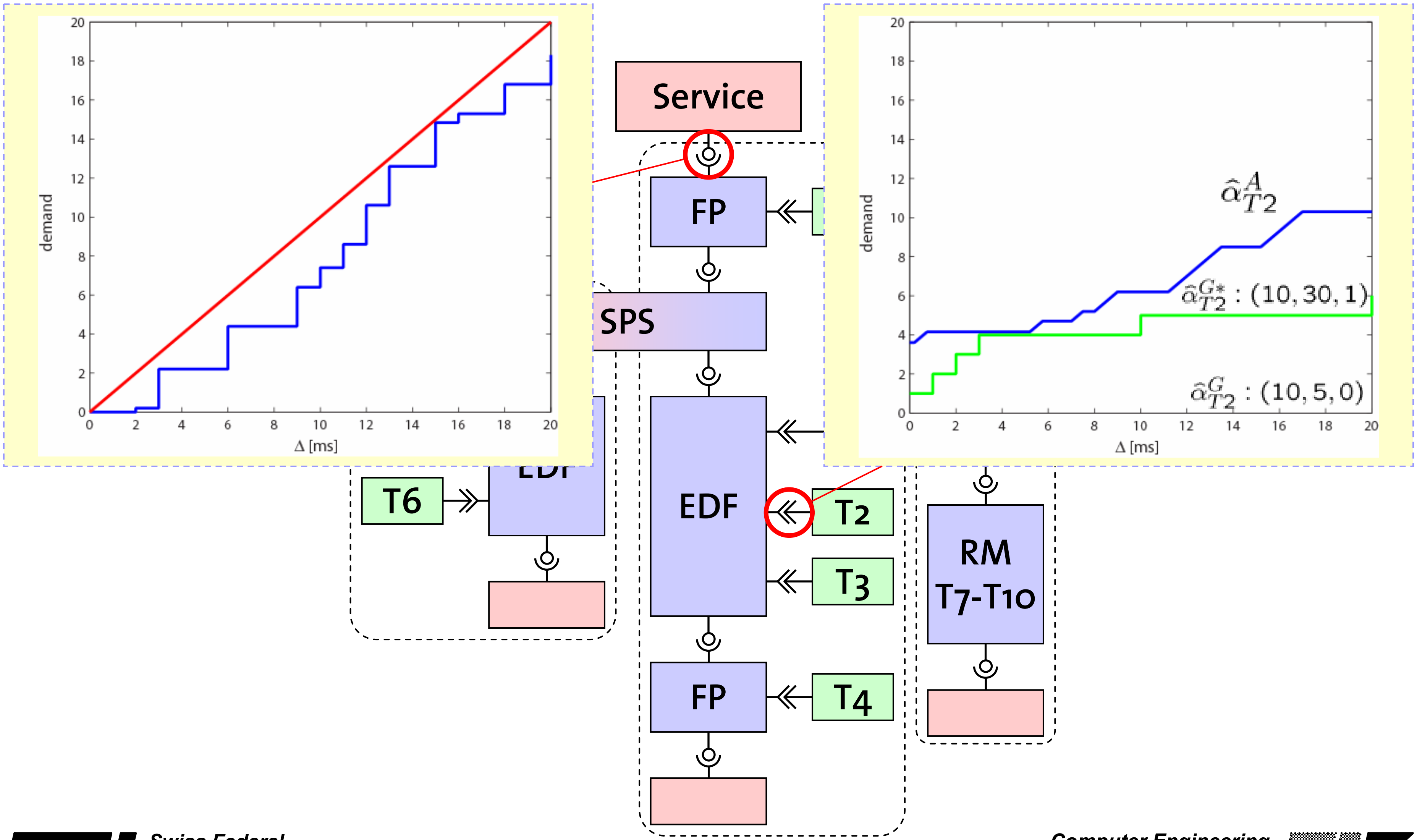


Implicit by Successful Composition!

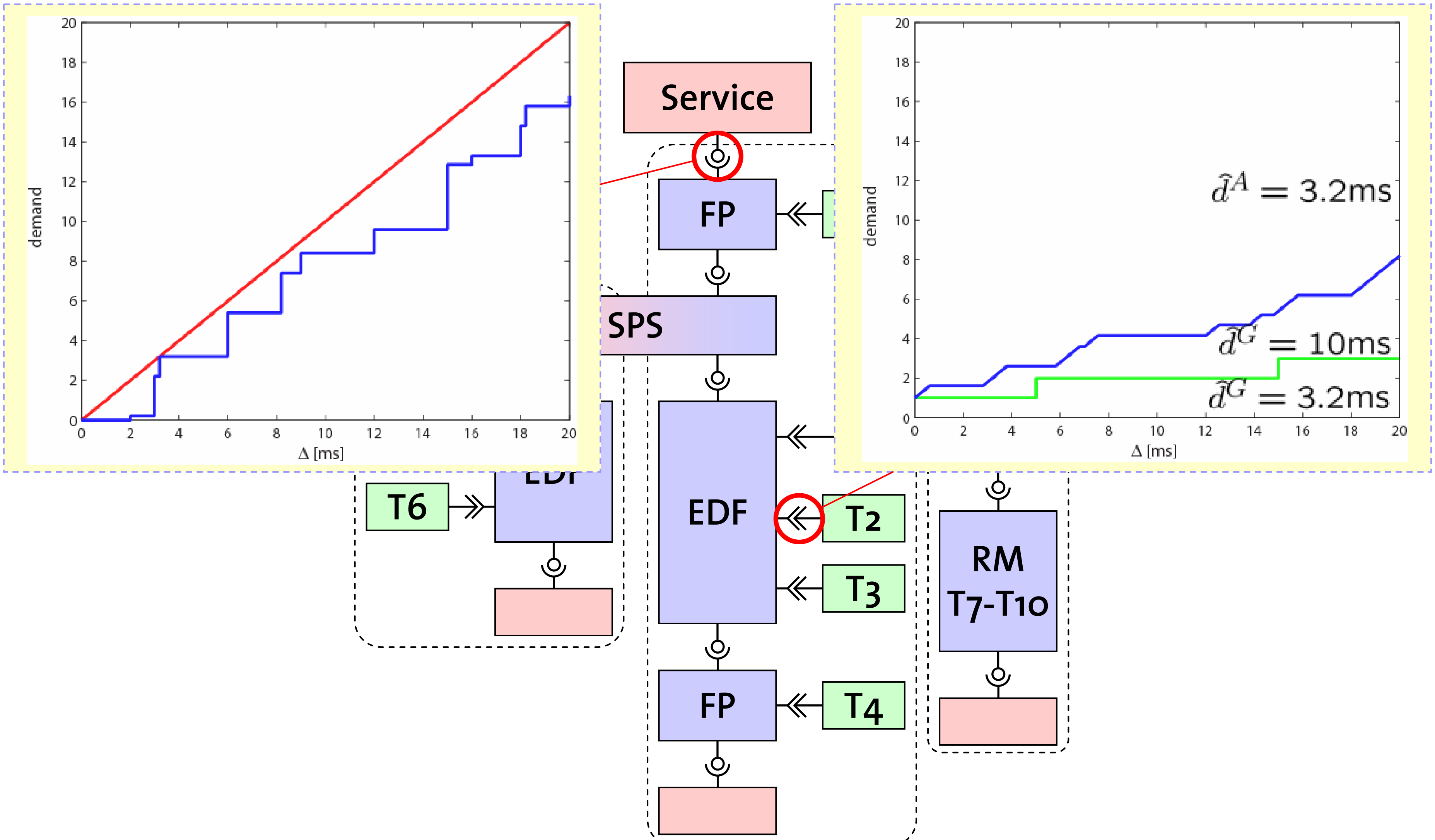
Schedulability Analysis



System Adaption I: Burstiness of T2



System Adaption II: Deadline of T2



System Analysis Time

- < 1 second
  - Pentium Mobile 1.6 GHz
  - Matlab 7 SP2
  - RTC Toolbox

Thank you!

www.mpa.ethz.ch/rtctoolbox

Nikolay Stoimenov
nikolays@tik.ee.ethz.ch

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Modular Performance Analysis with Real-Time Calculus

6. Comparison with other analysis approaches

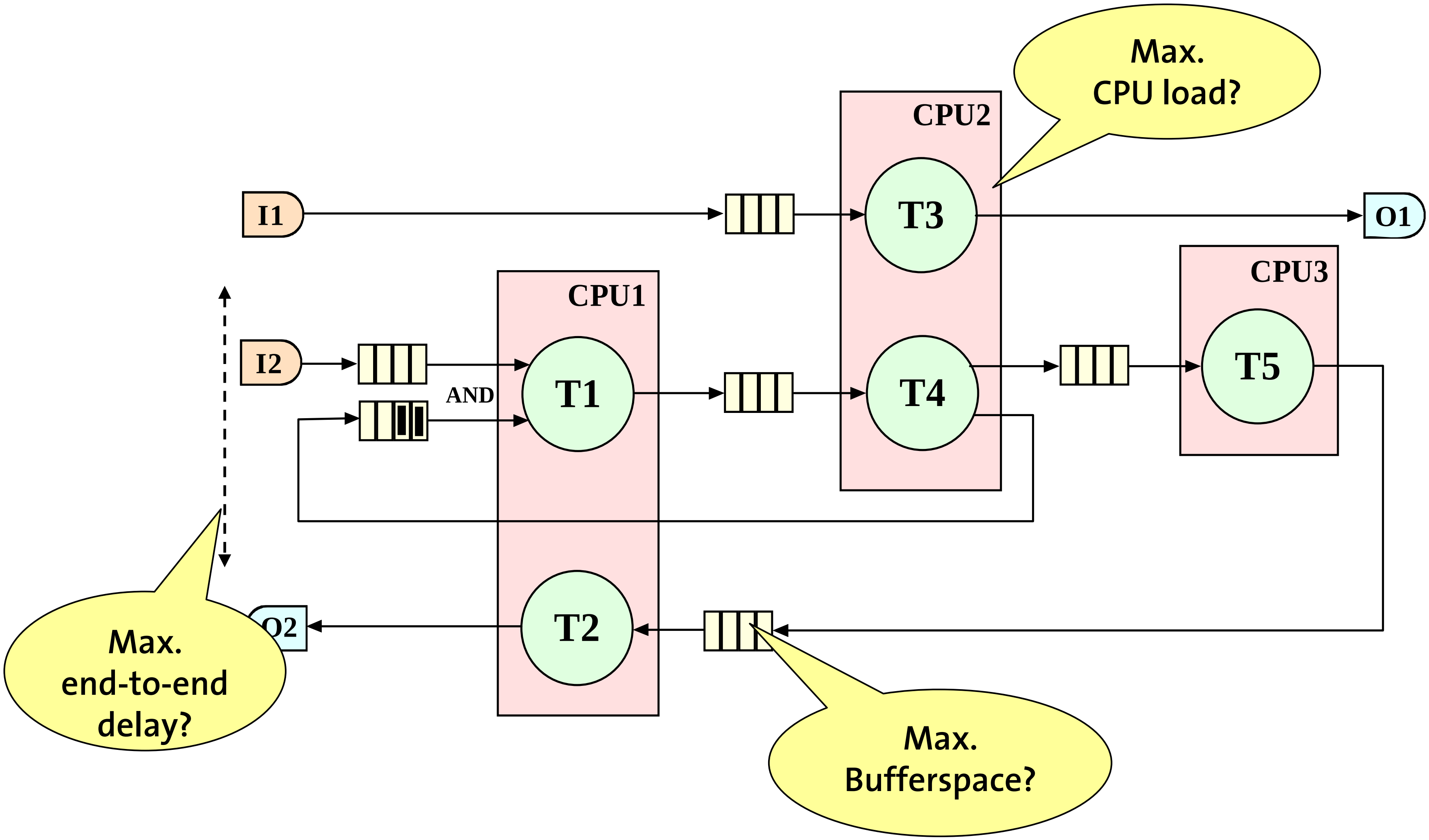
Simon Perathoner

ARTIST2 PhD Course on Automated Formal Methods for Embedded Systems
DTU - Lyngby, Denmark - June 11, 2007

Outline

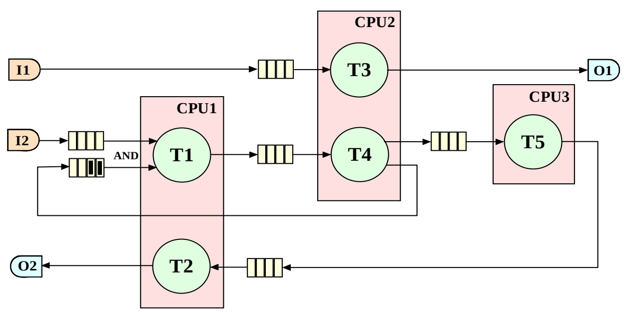
- Motivation
- Abstractions
- Benchmarks
- Conclusions

System level performance analysis



Formal analysis methods

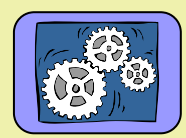
Distributed system



Abstraction 3

$$r_i = C_i + \sum_{\forall j \in hp(i)} \lceil \frac{r_i}{T_j} \rceil C_j$$

Analysis method 3



Performance values

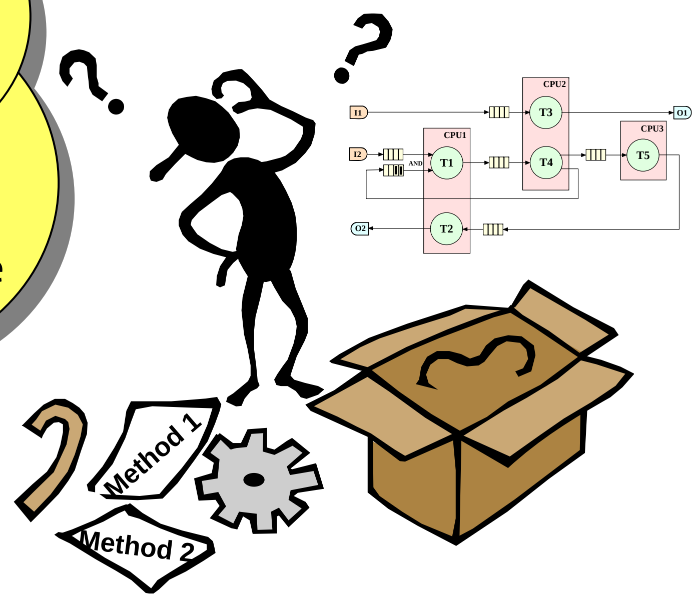
Motivating questions

- What is the influence of the different models on the analysis accuracy ?
- Does abstraction matter ?
- Which abstraction is best suited for a given system ?

Evaluation and comparison of abstractions is needed !

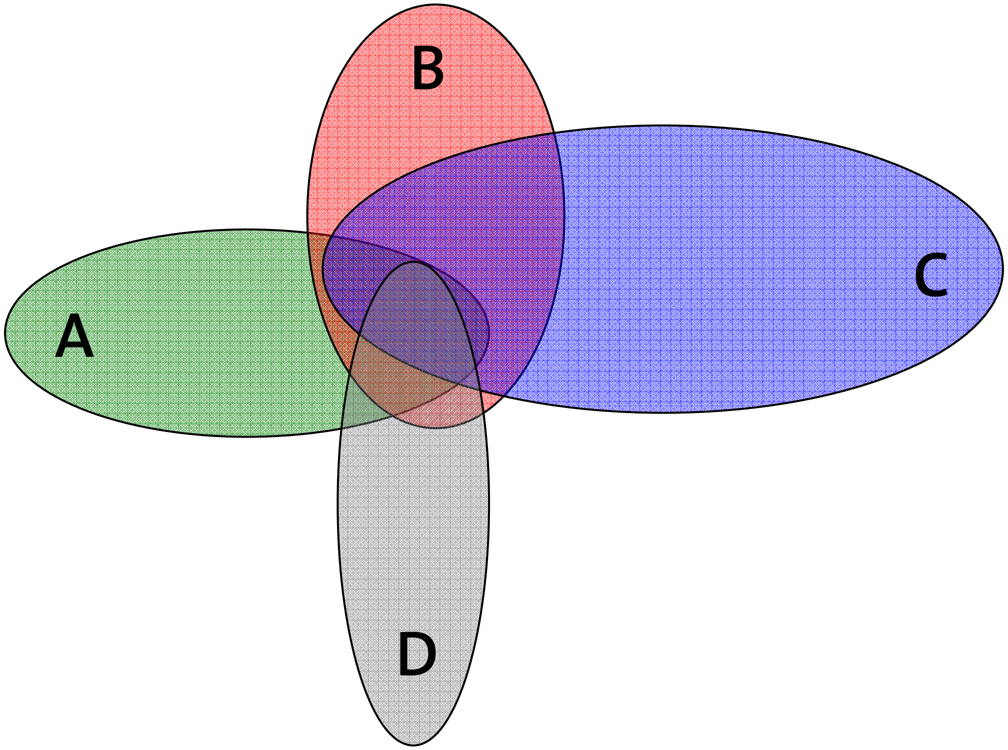
How can we compare different abstractions ?

modeling effort
modular
performance metrics
correctness
scope
scalability
usability
analysis time
accuracy
exact
efficient implementation
simple
compositional
holistic
stepwise refinement
tool support
learning curve
easy to use



What makes a direct comparison difficult?

- Many aspects can not be quantified
- Models cover different scenarios:



Intention

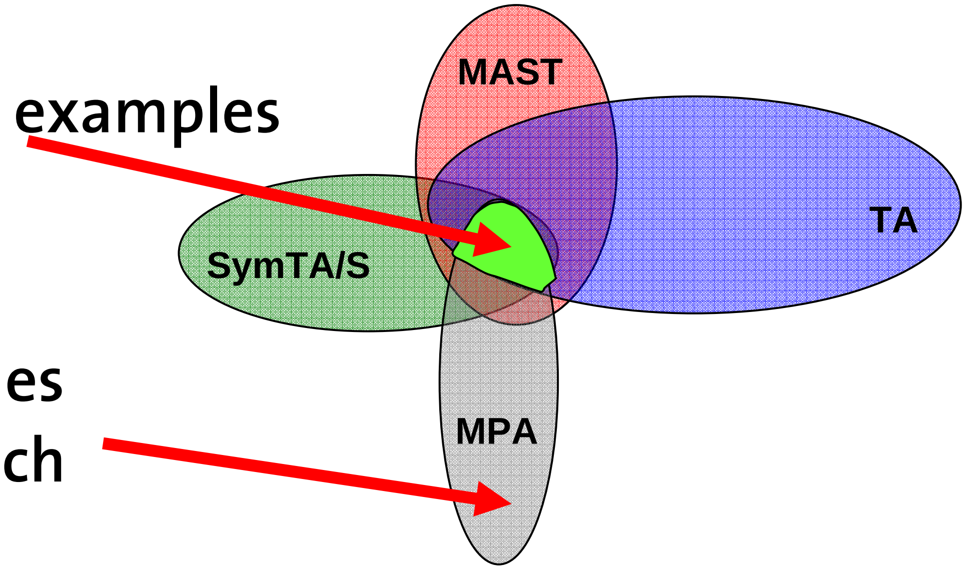
Compare models and methods that analyze the timing properties of distributed systems:

- **SymTA/S [Richter *et al.*]**
- **MPA-RTC [Thiele *et al.*]**
- **MAST [González Harbour *et al.*]**
- **Timed automata based analysis [Yi *et al.*]**

...

Approach

- Leiden Workshop on Distributed Embedded Systems: http://www.tik.ee.ethz.ch/~leiden05/
- Define a set of benchmark examples that cover common area
- Define benchmark examples that show the power of each method



Expected (long term) results

- Understand the modeling power of different methods
- Understand the relation between models and analysis accuracy
- Improve methods by combining ideas and abstractions

Contributions

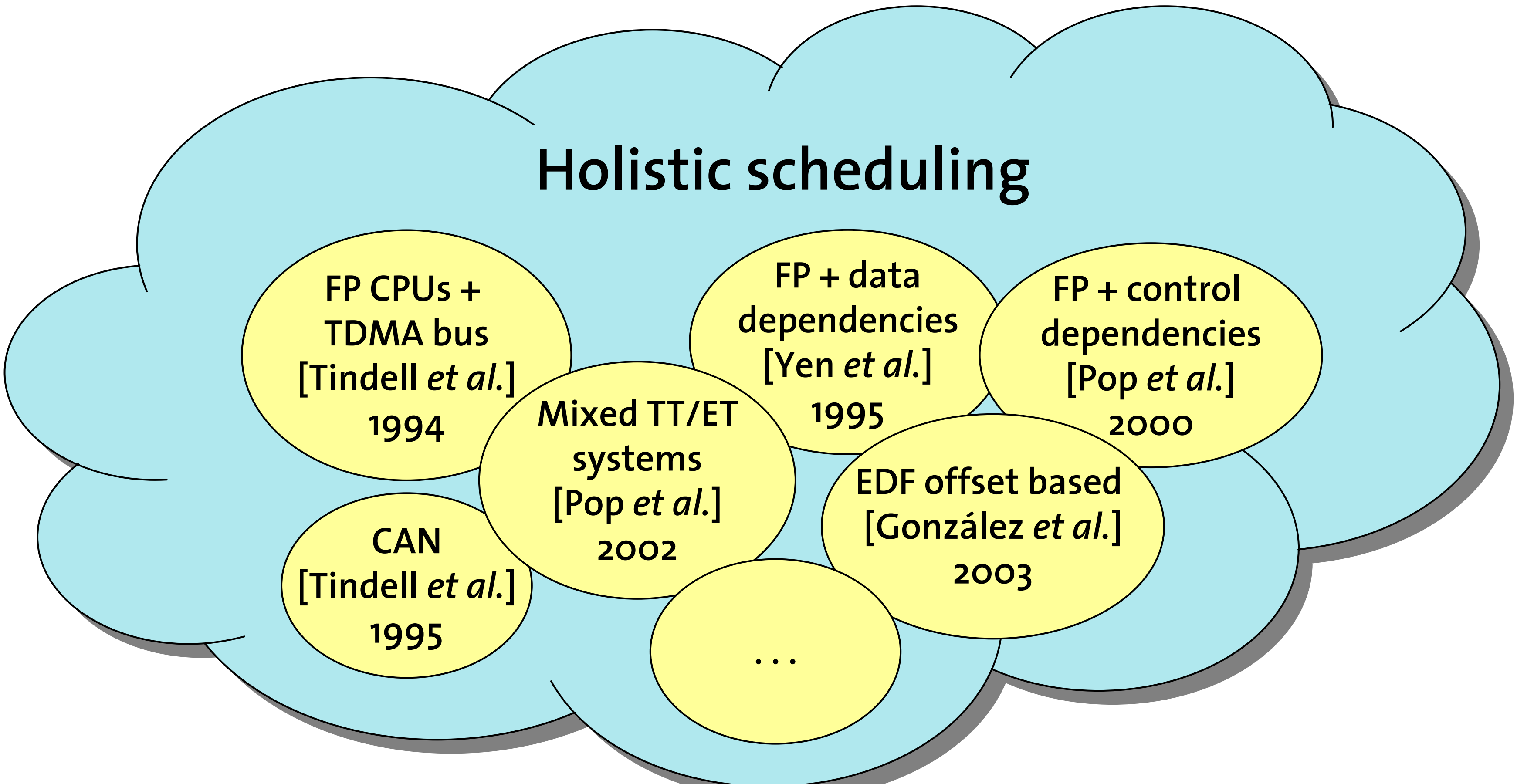
- We define a set of benchmark systems aimed at the evaluation of performance analysis techniques
- We apply different analysis methods to the benchmark systems and compare the results obtained in terms of accuracy and analysis times
- We point out several analysis difficulties and investigate the causes for deviating results

Outline

- Motivation
- Abstractions
- Benchmarks
- Conclusions

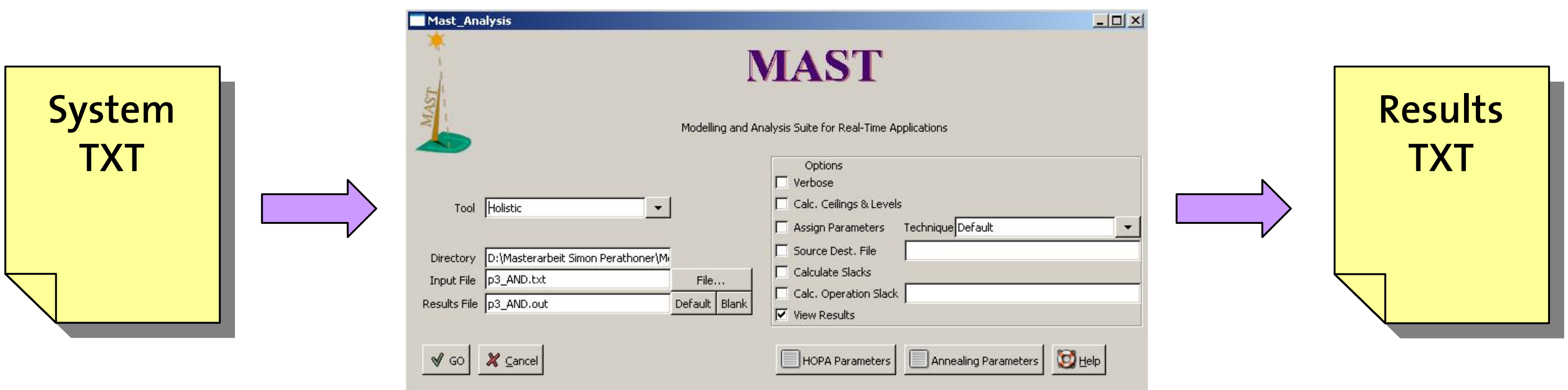
Abstraction 1 - Holistic scheduling

Basic concept: extend concepts of classical scheduling theory to distributed systems



Holistic scheduling – MAST tool

MAST - The Modeling and Analysis Suite for Real-Time Applications [González Harbour *et al.*]



Abstraction 2 – The SymTA/S approach

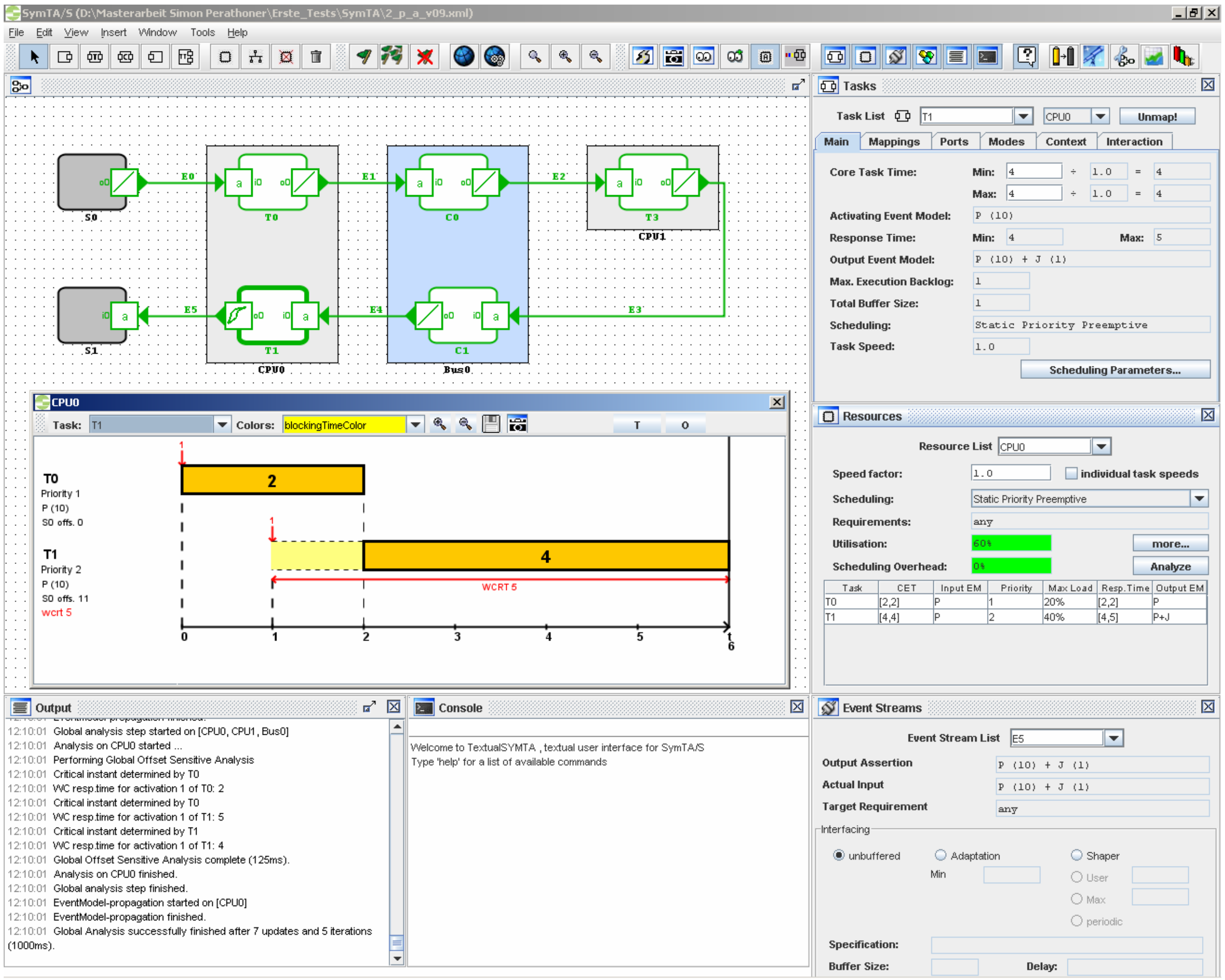
Basic concept: Application of classical scheduling techniques at resource level and propagation of results to next component

Problem: The local analysis techniques require the input event streams to fit given standard event models

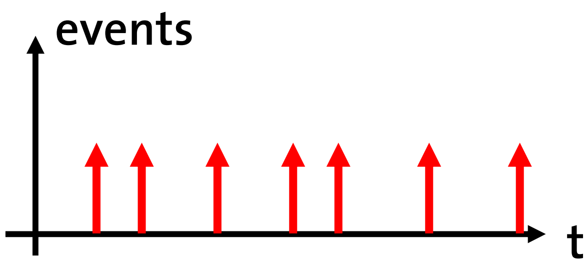
Solution: Use appropriate interfaces: EMIFs & EAFs

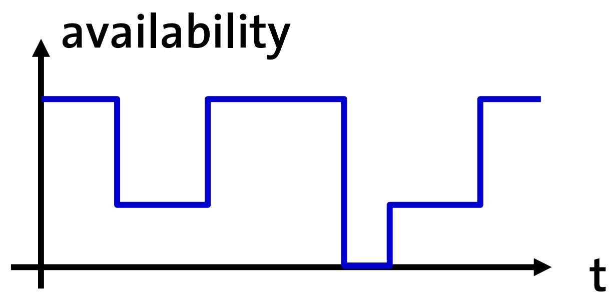
SymTA/S – Tool

SYMTA VISION



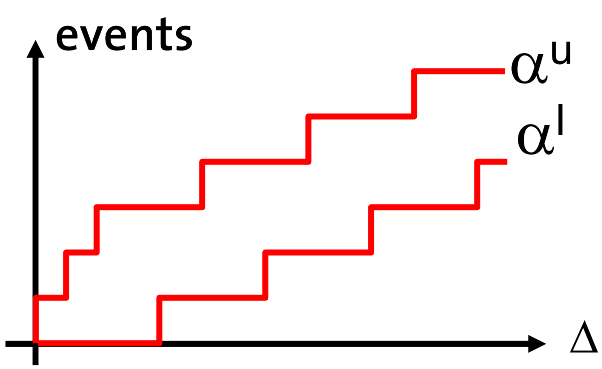
Abstraction 3 – MPA-RTC



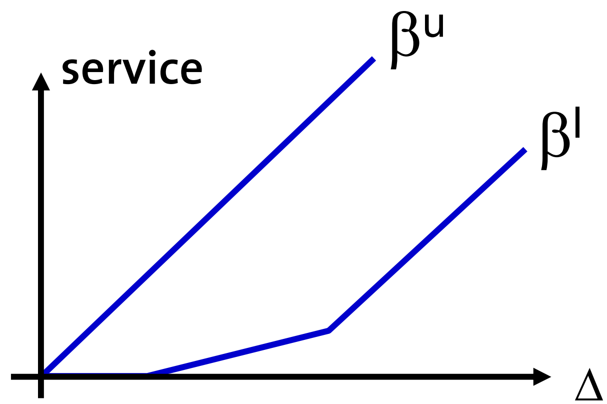


Load model

Service model

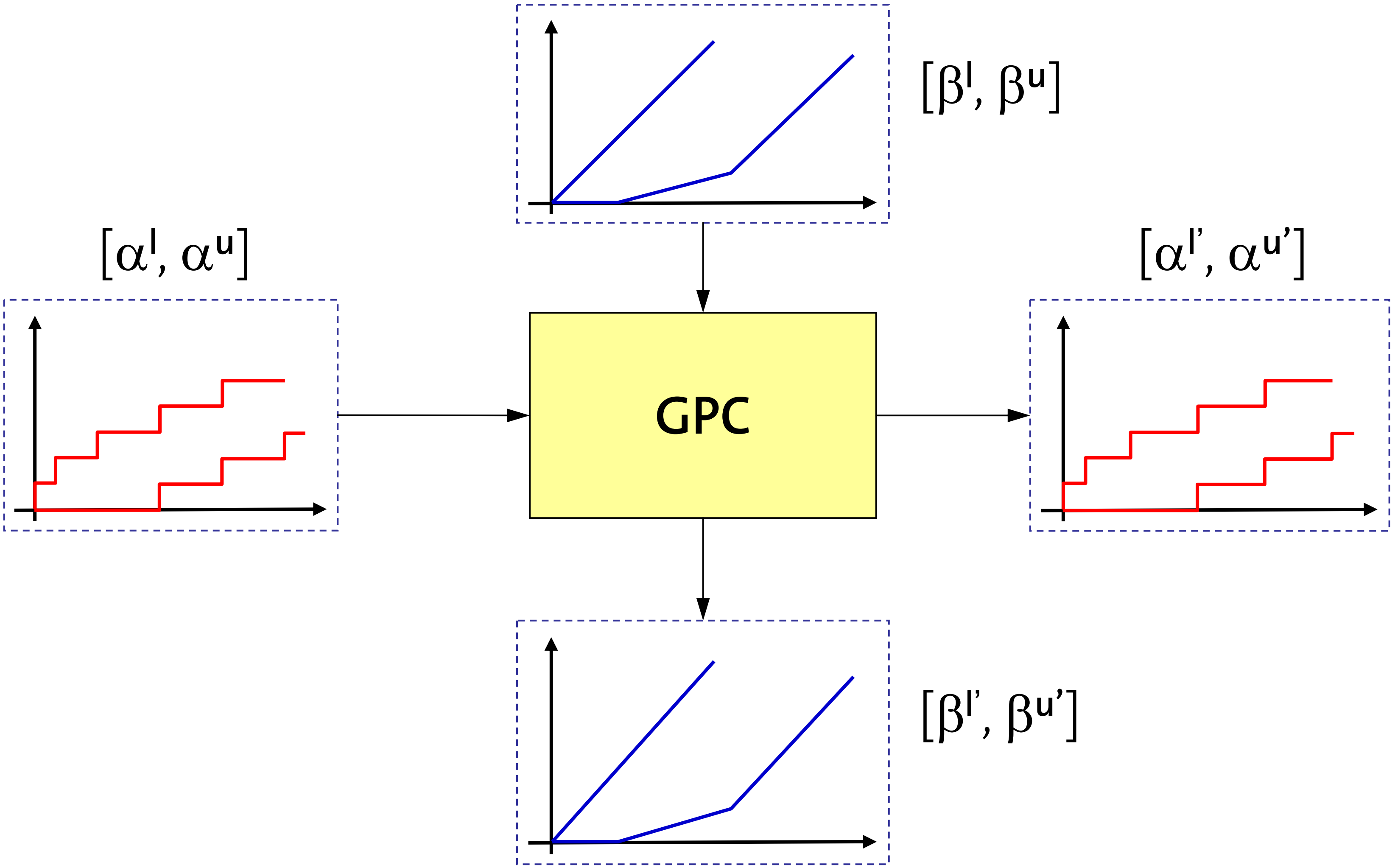


Arrival curves

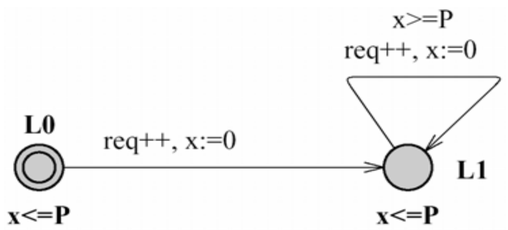


Service curves

Abstraction 3 – MPA-RTC



Abstraction 4 - TA based performance analysis

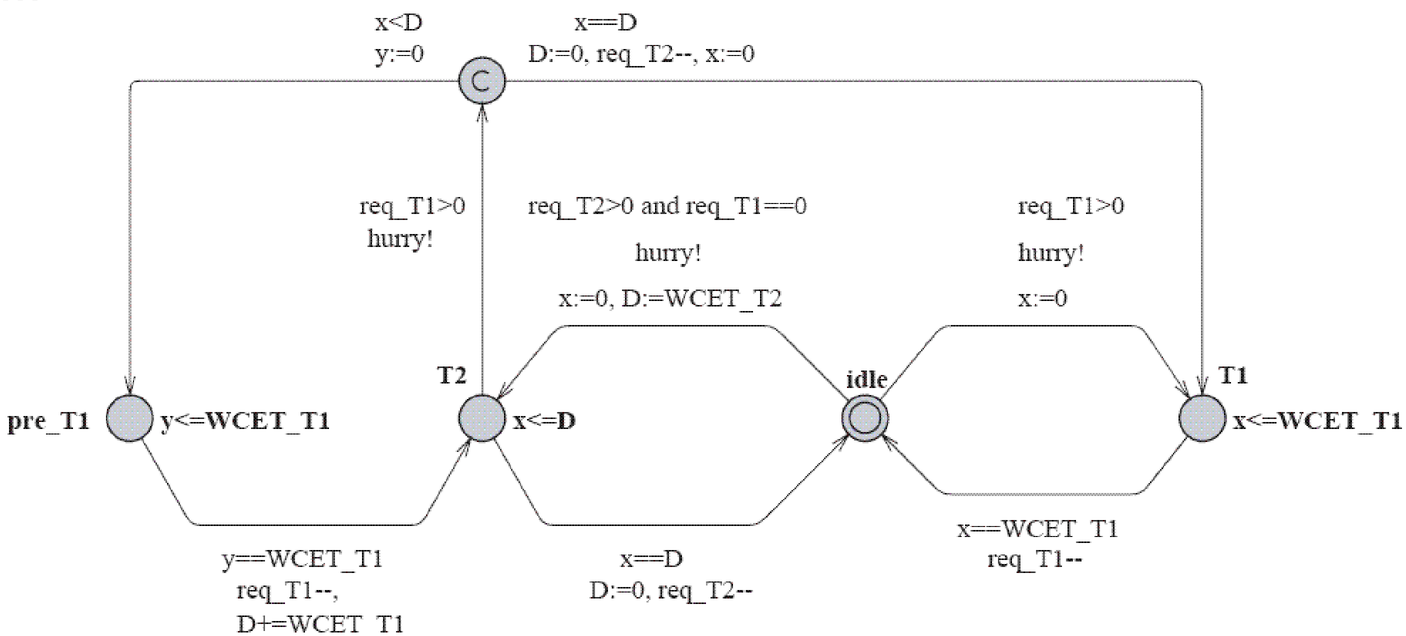


periodic stream

Verification of performance properties by model checking (UPPAAL)

Exact performance values

fixed priority scheduling



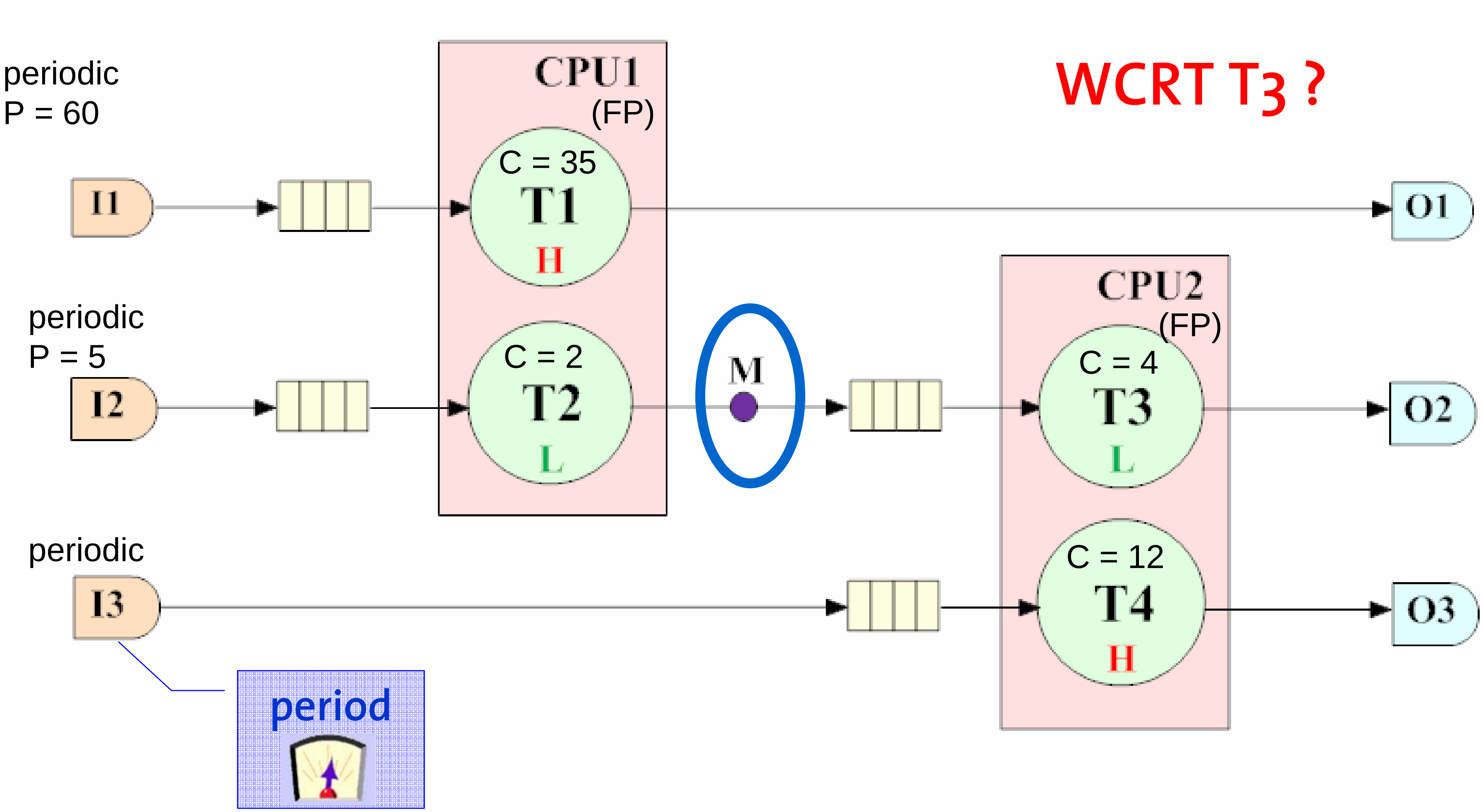
Outline

- Motivation
- Abstractions
- Benchmarks
- Conclusions

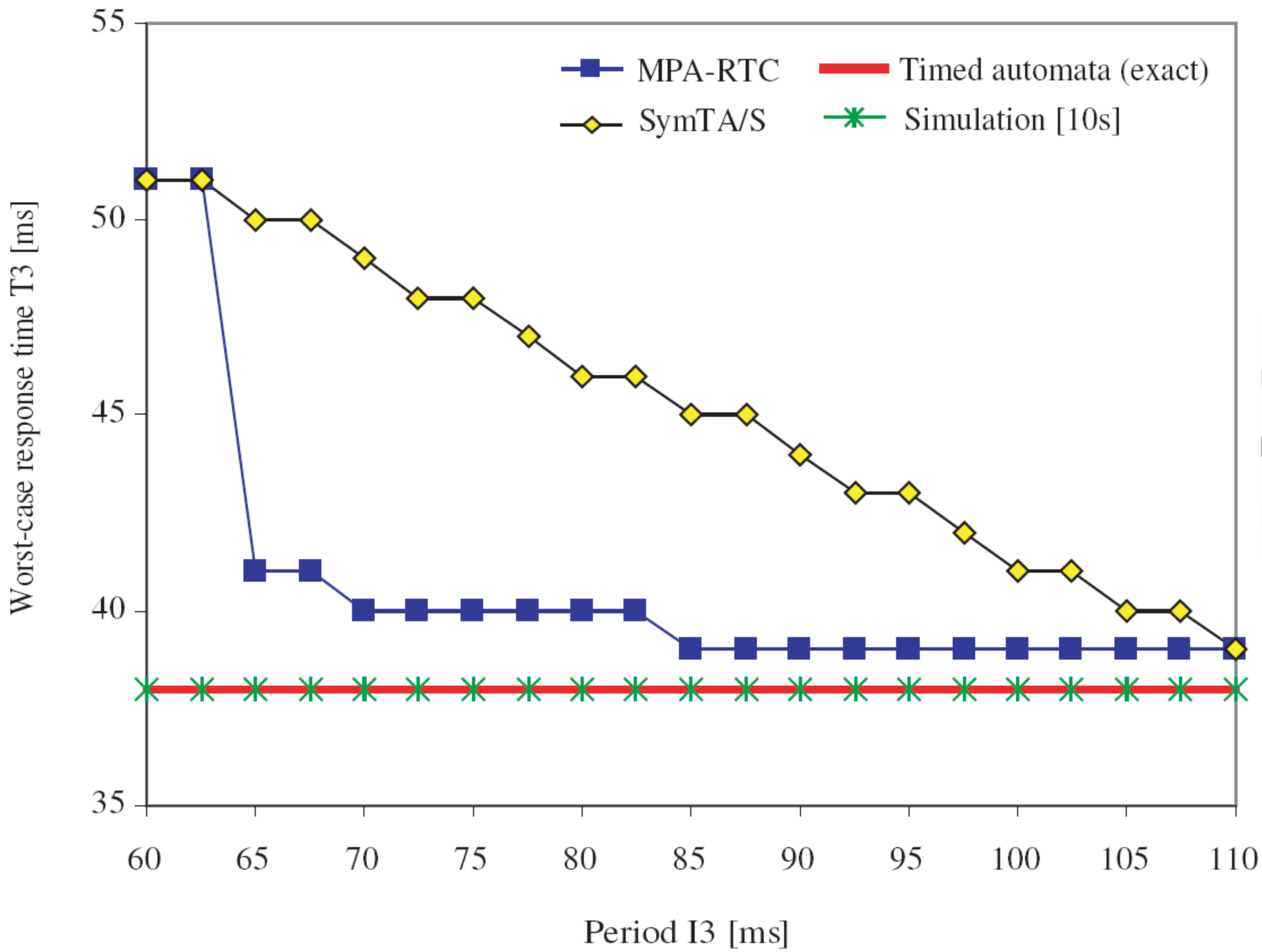
Benchmarks

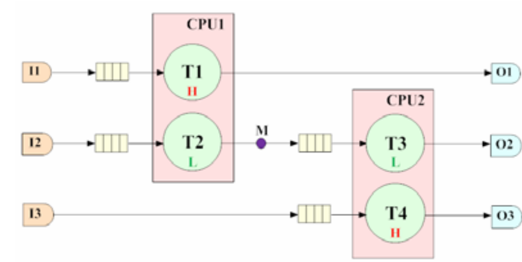
- Pay burst only once
- Complex activation pattern
- Variable feedback
- Cyclic dependencies
- AND/OR task activation
- Intra-context information
- Workload correlation
- Data dependencies

Benchmark 1 – Complex activation pattern



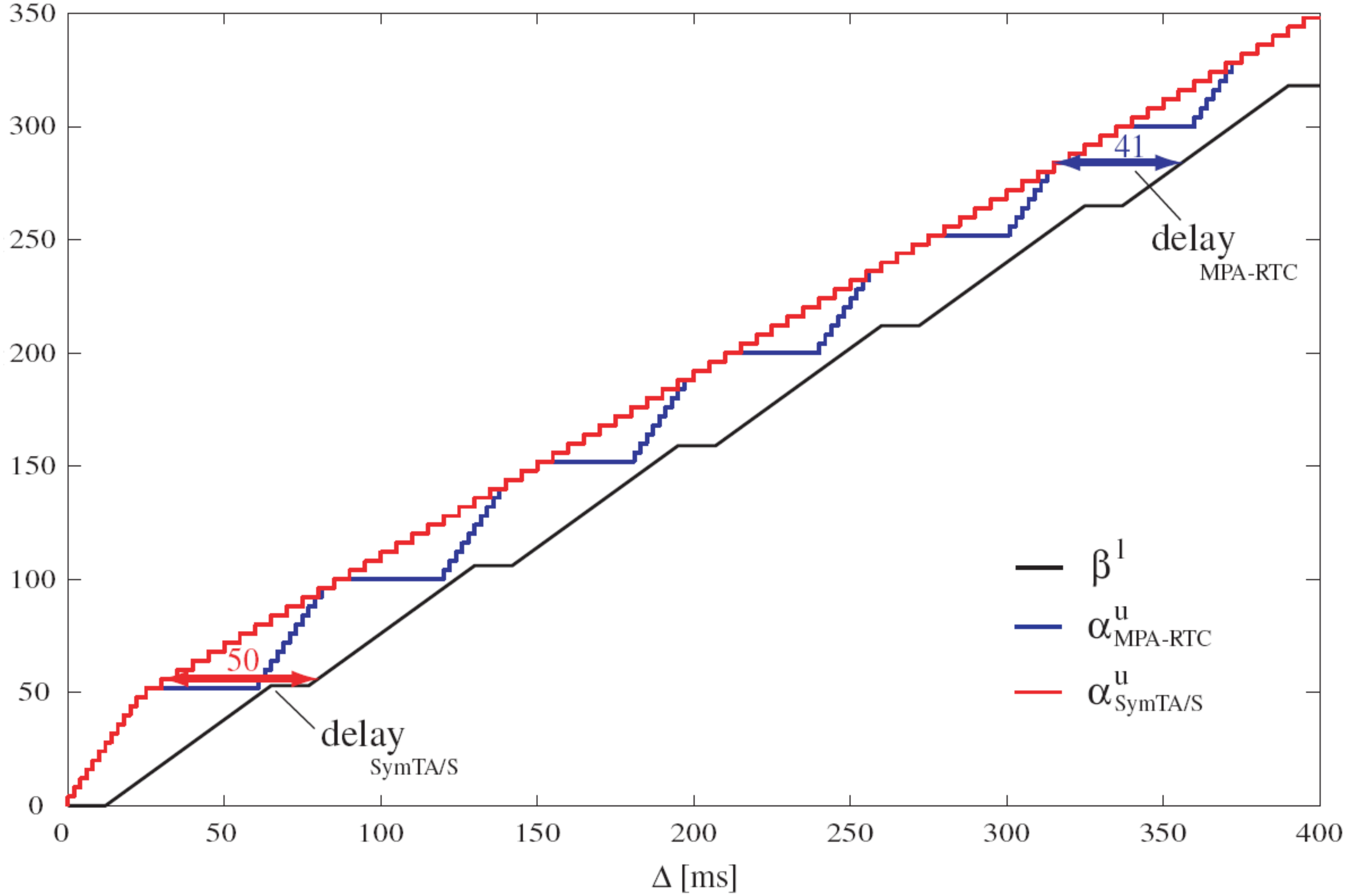
Benchmark 1 – Analysis results



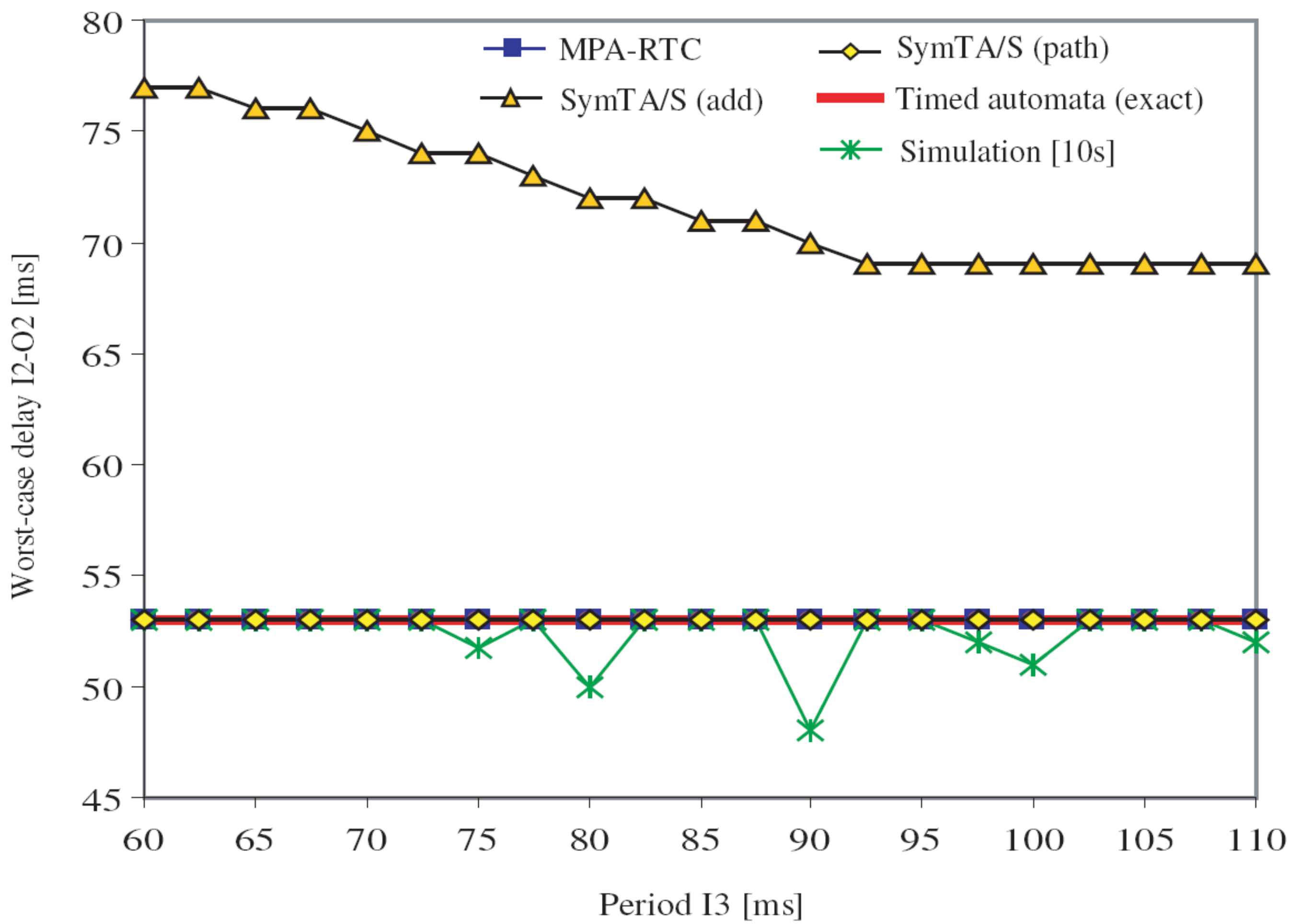


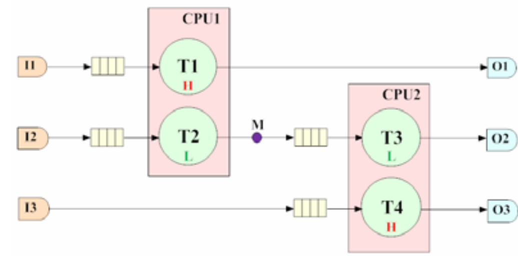
Benchmark 1 – Result interpretation

P_{I_3} = 65 ms

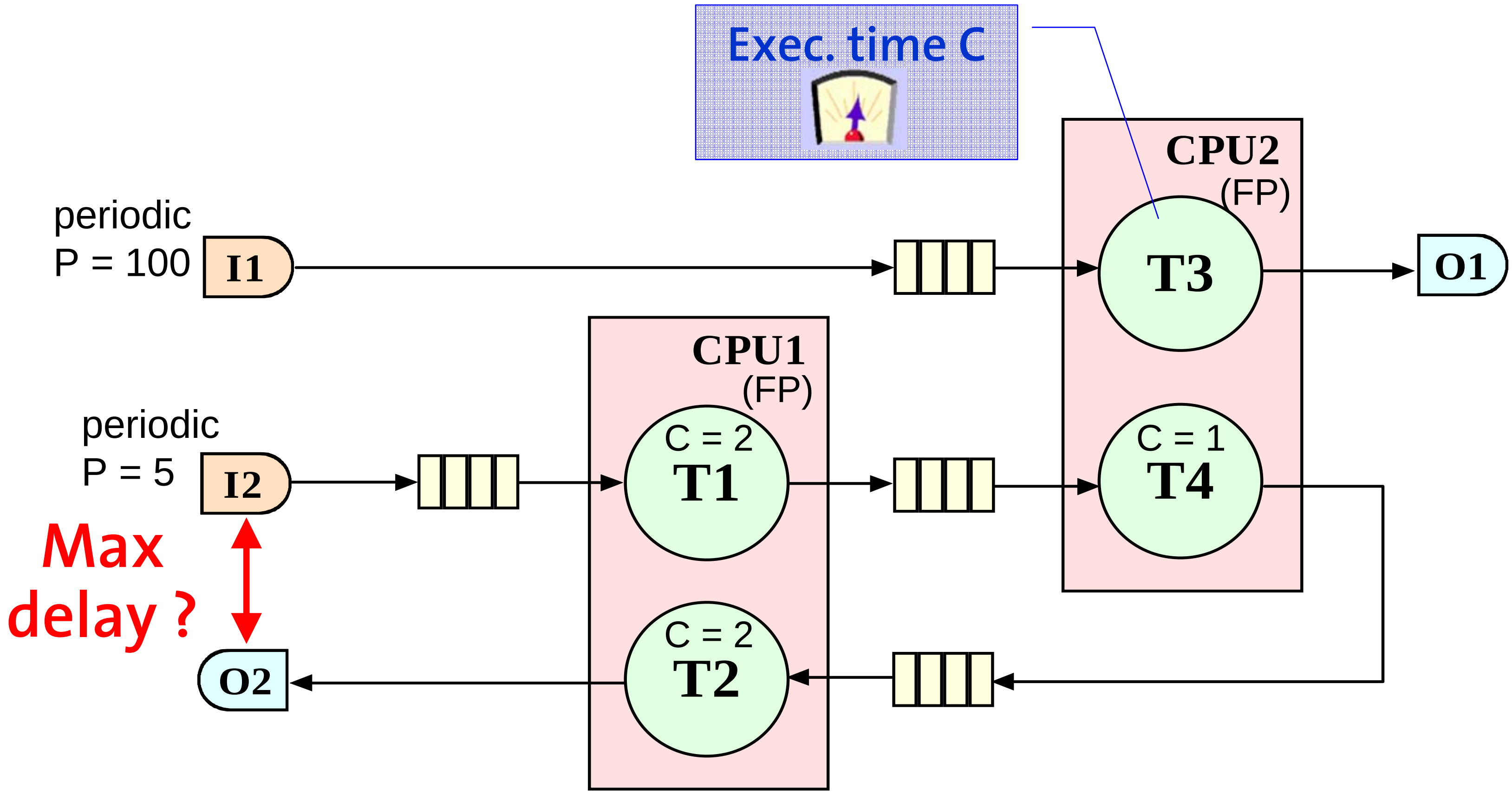


Benchmark 1 – Worst case Delay I2-O2

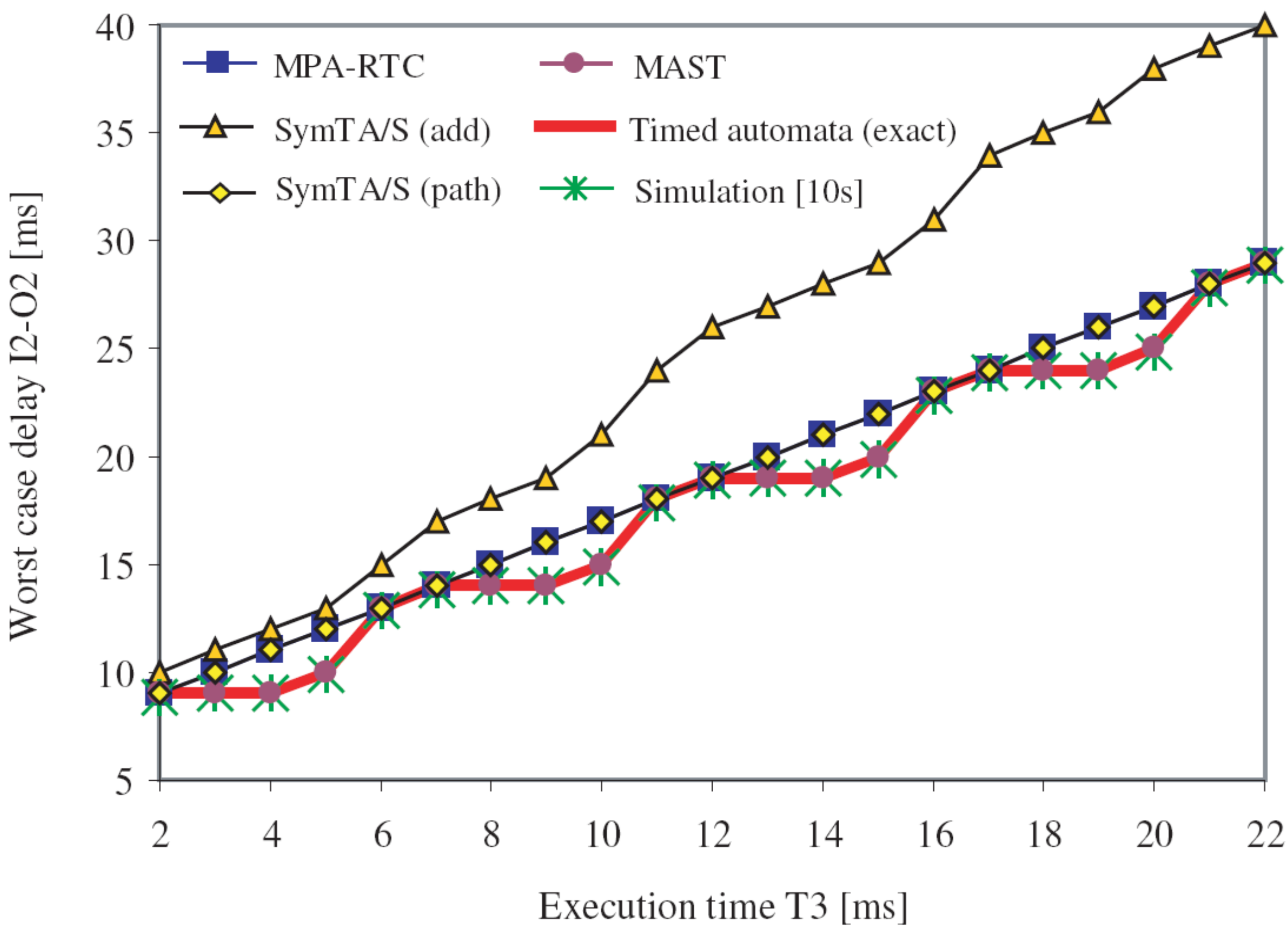


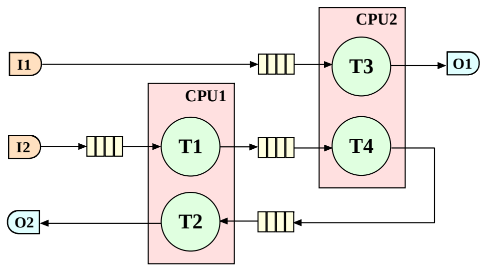


Benchmark 2 – Variable feedback

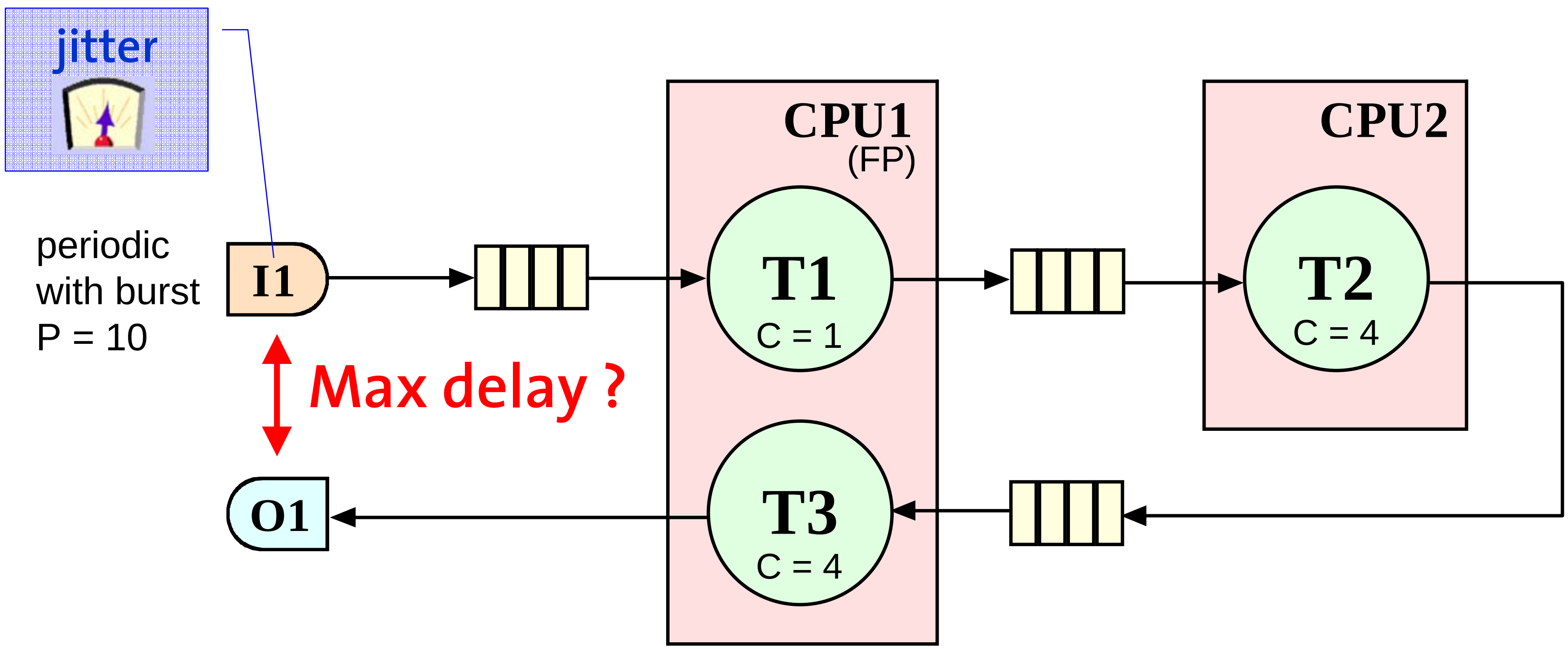


Benchmark 2 – Analysis results



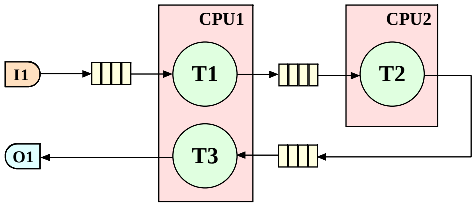


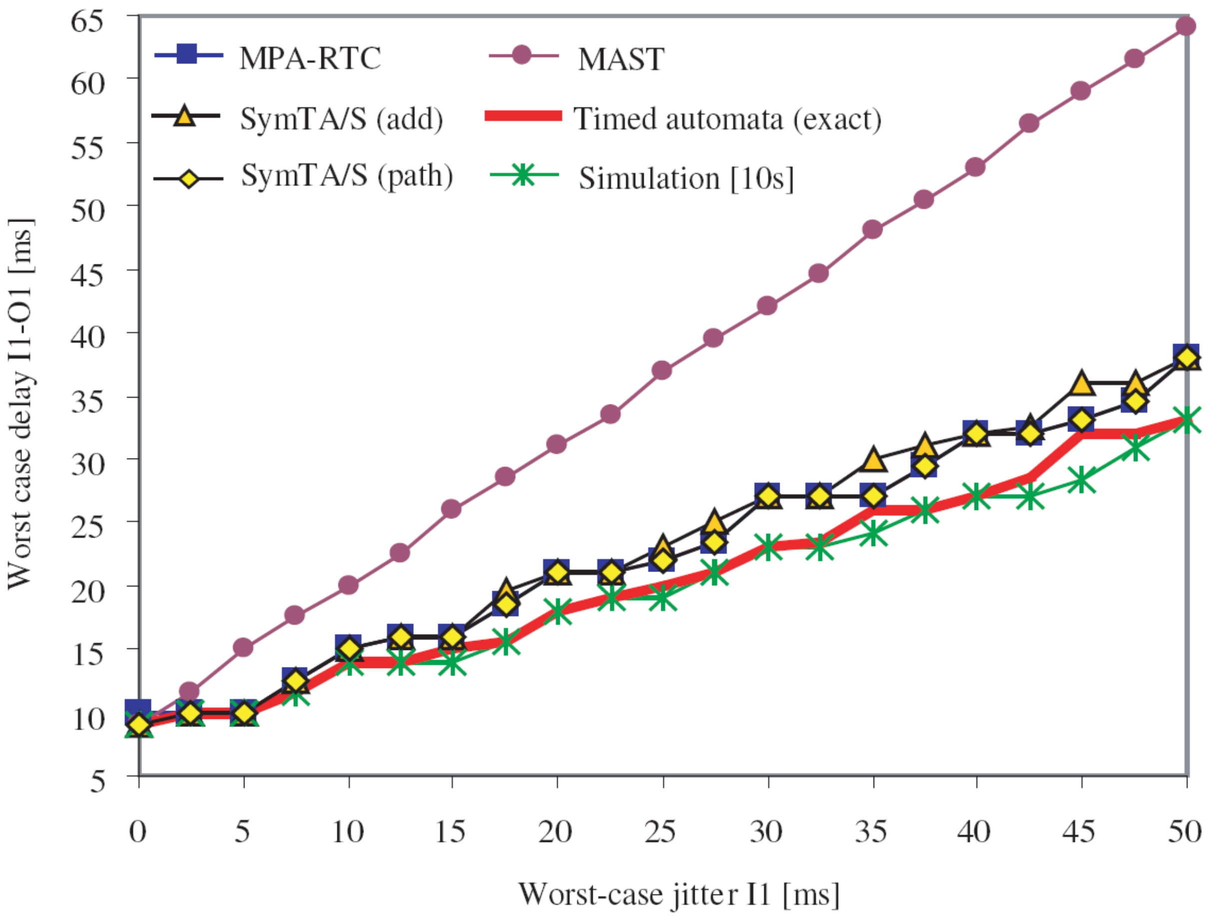
Benchmark 3 – Cyclic dependencies



Benchmark 3 – Analysis results

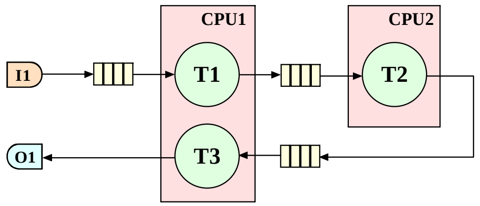
Scenario 1: priority T1 = high
priority T3 = low

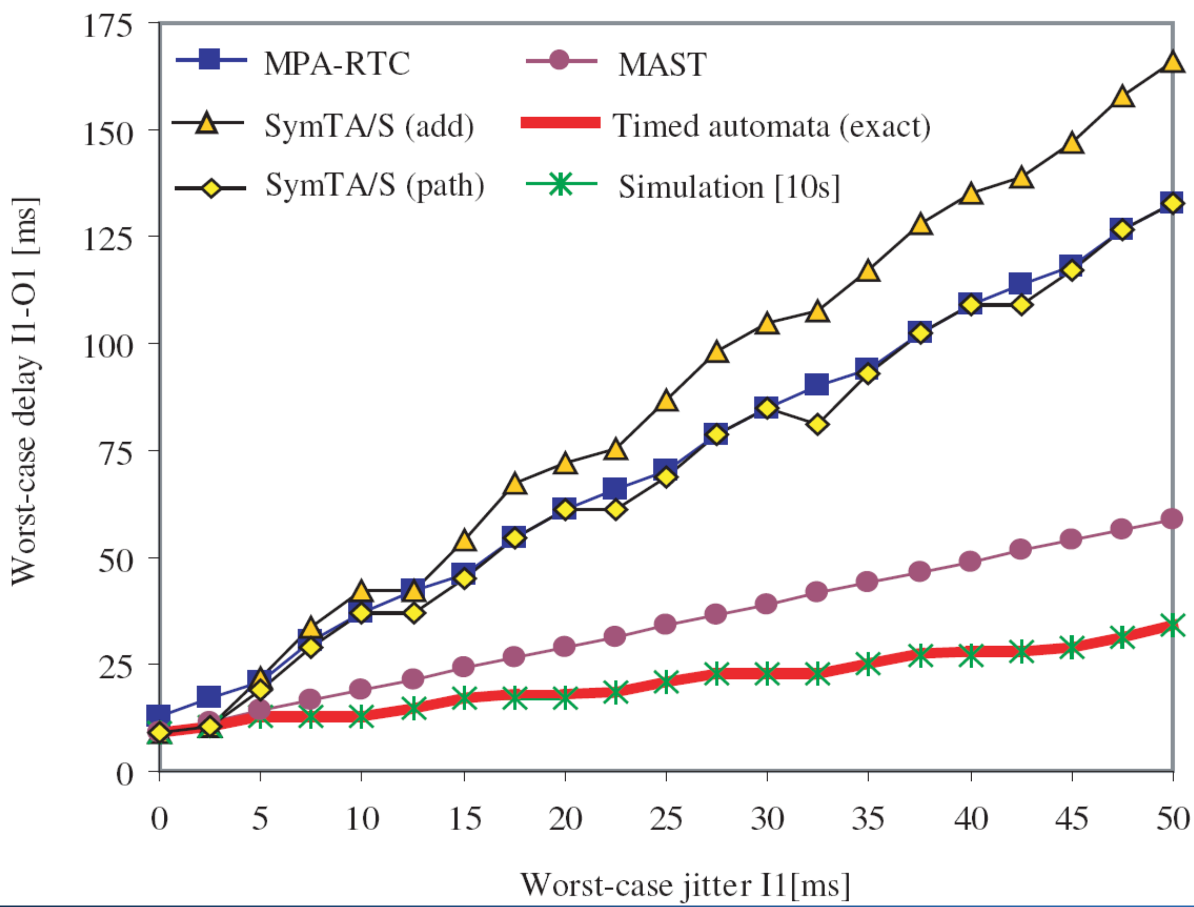




Benchmark 3 – Analysis results

Scenario 2: priority T1 = low
priority T3 = high





Analysis times [s]

		B1	B2	B3 (sc.1)	B3 (sc.2)	B4
MPA-RTC	min	0.60	0.03	0.01	0.04	0.03
	med	1.06	0.04	0.01	0.15	0.05
	max	19.72	0.08	0.04	0.30	0.20
SymTA/S	min	0.05	0.03	0.03	0.03	0.06
	med	0.09	0.05	0.06	0.34	0.09
	max	1.50	0.23	0.09	0.80	0.31
MAST	min	-	< 0.5	< 0.5	< 0.5	< 0.5
	med	-	< 0.5	< 0.5	< 0.5	< 0.5
	max	-	< 0.5	< 0.5	< 0.5	< 0.5
Timed aut.	min	18.0	< 0.5	< 0.5	< 0.5	< 0.5
	med	34.5	< 0.5	1.0	< 0.5	< 0.5
	max	60.5	< 0.5	52.0	5.5	< 0.5
Simulation	min	1.0	< 0.5	0.5	0.5	< 0.5
	med	1.0	< 0.5	0.5	0.5	< 0.5
	max	1.0	< 0.5	0.5	0.5	< 0.5

Outline

- Motivation
- Abstractions
- Benchmarks
- Conclusions

Conclusions

- The analysis accuracy and the analysis time depend highly on the specific system characteristics
- None of the analysis methods performed best in all benchmarks
- The analysis results of the different approaches are remarkable different even for apparently basic systems
- The choice of an appropriate analysis abstraction matters
- The problem to provide accurate performance predictions for general systems is still far from solved

Discussion

- Approximation of complex event streams with standard event models can lead to poor performance predictions at local level
- Holistic approaches better in the presence of correlations among task activations (e.g. data dependencies)
- Cyclic dependencies represent a serious pitfall for the accuracy of compositional analysis methods
- Holistic methods less appropriate for timing properties referred to the *actual* release time of an event within a large jitter interval

Thank you!

Simon Perathoner

perathoner@tik.ee.ethz.ch