Theory and Practice of EDF Scheduling in Distributed Real-Time Systems

J. Javier Gutiérrez and Héctor Pérez

Software Engineering and Real-Time Group
Universidad de Cantabria

Funded in part by the Spanish Government (TIN2014-56158-C4-2-P)

23rd International Conference on Reliable Software Technologies
Ada-Europe 2018

Lisbon (Portugal), 18-22 June 2018

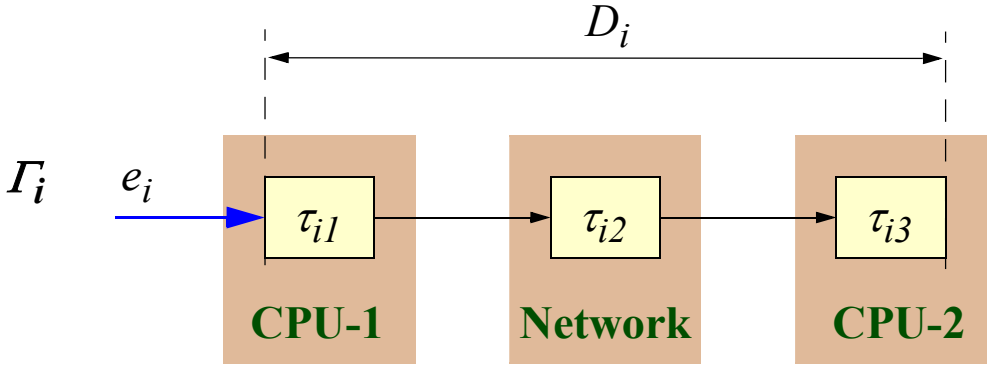
Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

The context

Our work on the schedulability analysis and optimization of distributed real-time systems for the EDF (Earliest Deadline First) policy

- event-driven systems
- end-to-end (e2e) flows
  - steps (tasks or messages)
  - periodic or sporadic external events
- processing resources (processors or networks)
- timing requirements: global end-to-end deadlines



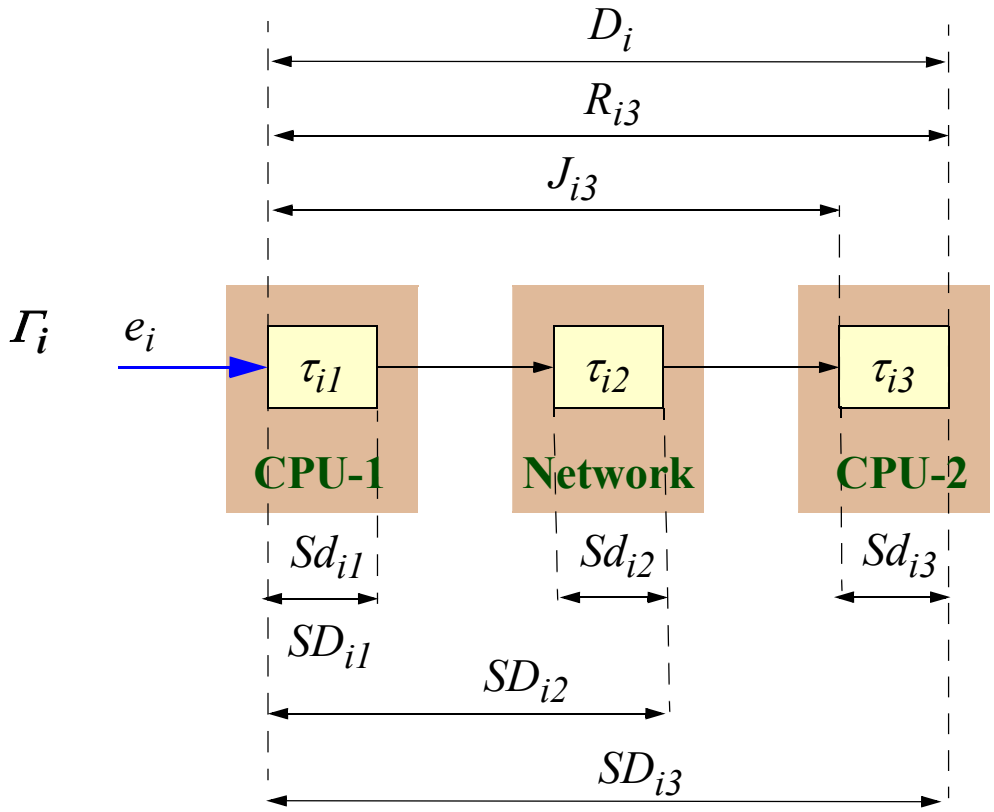
The context: response time analysis

Holistic schedulability analysis by Spuri (1996)

- based on *global* scheduling deadlines *SD* relative to the arrival of the external event
- a global clock reference is needed (*GC-EDF*)

We proposed a holistic schedulability analysis for *LC-EDF* (Rivas et at 2010)

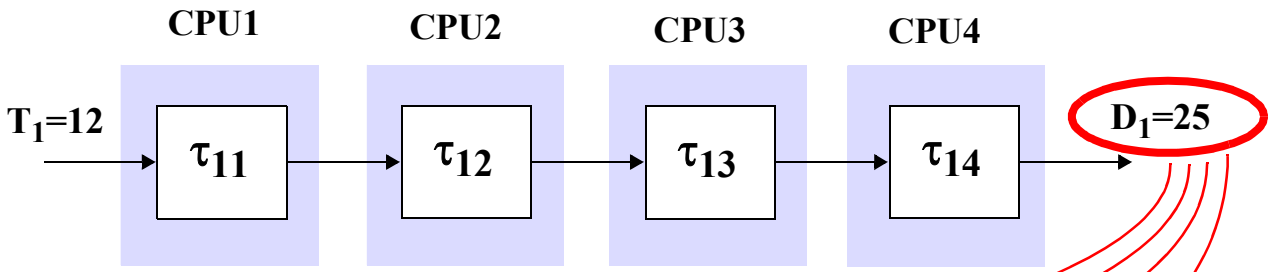
- based on *local* scheduling deadlines *Sd* relative to the arrival of the event triggering the step



The context: off-line scheduling deadline assignment

Transformation of the timing requirements of the e2e flow into individual scheduling deadlines of each step:

- e.g., by distributing the e2e deadline according to some criteria



Distribution of D_1	4	7	5	9

Sd_{ij}	4	7	5	9
SD_{ij}	4	11	16	25

The context: basic scheduling deadline assignment algorithms

Non-iterative algorithms proposed in the book by Jane Liu (2000):

- Ultimate Deadline (UD): assigns the e2e deadline of an e2e flow to each one of its steps
- Effective Deadline (ED): assigns the e2e deadline to each step minus the sum of the WCET of all the following steps in the same e2e flow
- Proportional Deadline (PD): distributes the e2e deadline proportionally to the WCET of each step

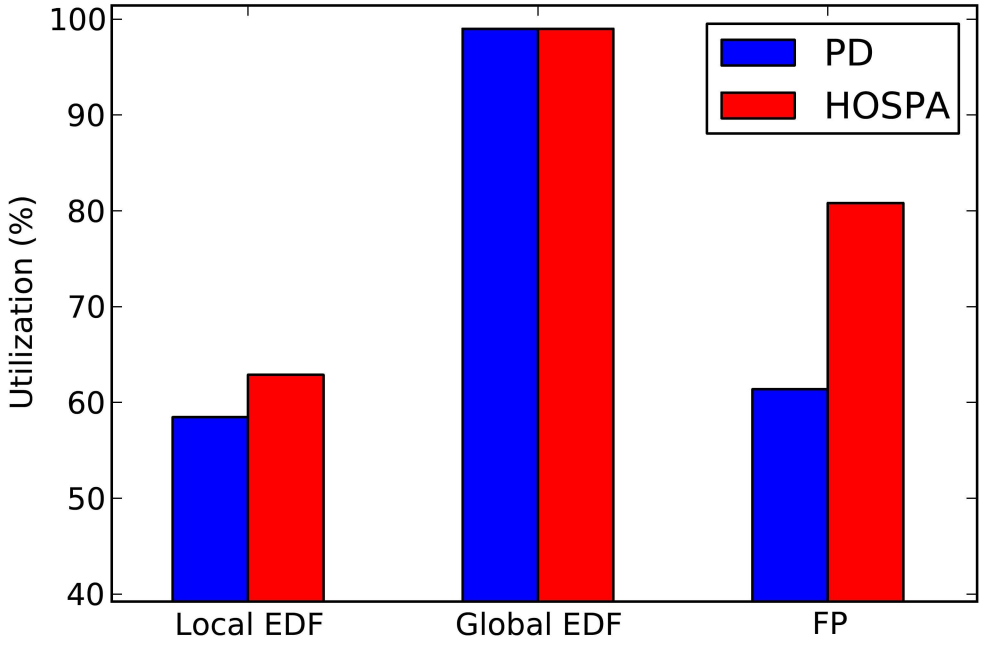
Iterative algorithm:

- HOSPA: heuristic algorithm for optimizing priorities or scheduling deadlines in distributed systems

The starting point

A preliminary comparison of the performance of FP, LC-EDF and GC-EDF in distributed systems:

- applying the scheduling parameter assignment algorithms that work fine for FP
- LC-EDF obtained poor results
  - breakdown utilization less than 65%
  - FP is greater than 80%
  - GC-EDF is greater than 95%

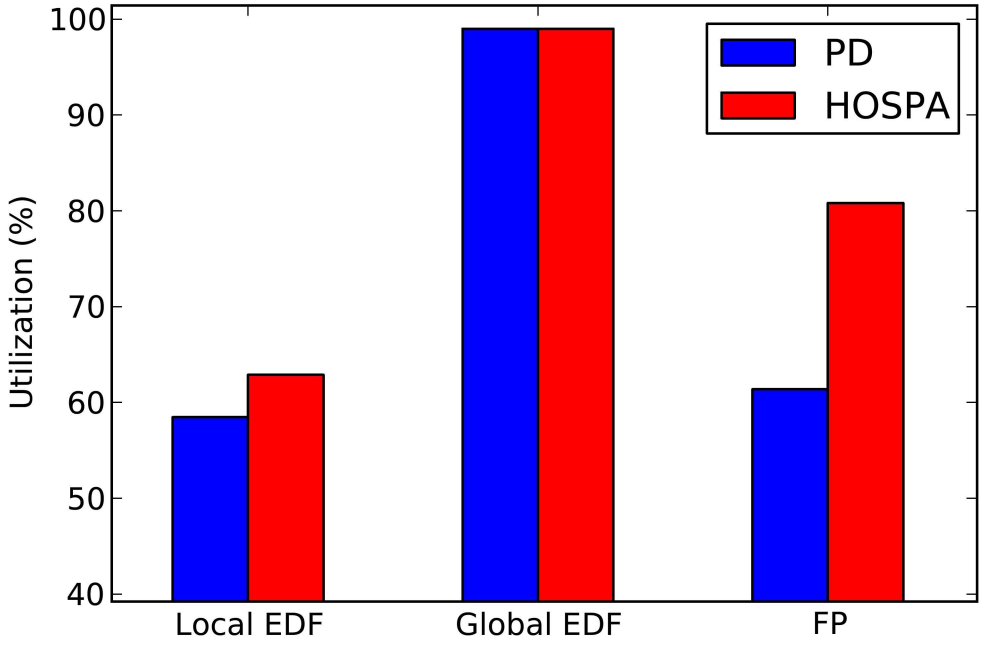


The starting point

A preliminary comparison of the performance of FP, LC-EDF and GC-EDF in distributed systems:

- applying the scheduling parameter assignment algorithms that work fine for FP
- LC-EDF obtained poor results
  - breakdown utilization less than 65%
  - FP is greater than 80%
  - GC-EDF is greater than 95%

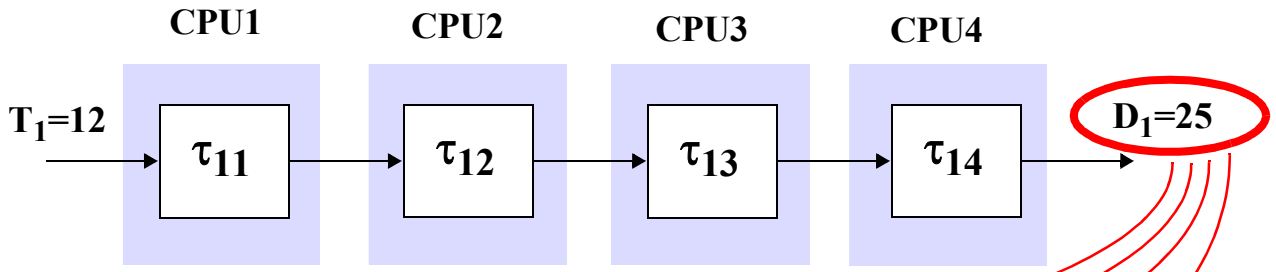
Why do these algorithms perform worse for LC-EDF?



A paradox

We tried other assignment algorithms removing the constraint of complying with the e2e deadline

- under certain conditions *UD* and *ED* obtain better results for LC-EDF

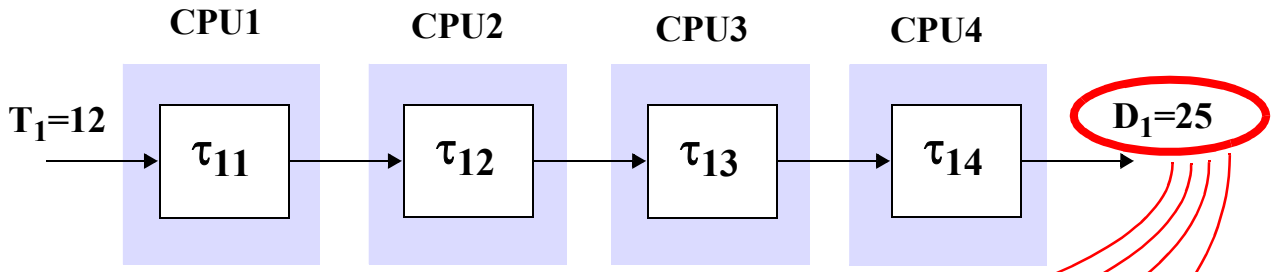


Example for PD	4	7	5	9
UD might be better!	25	25	25	25

A paradox

We tried other assignment algorithms removing the constraint of complying with the e2e deadline

- under certain conditions *UD* and *ED* obtain better results for LC-EDF



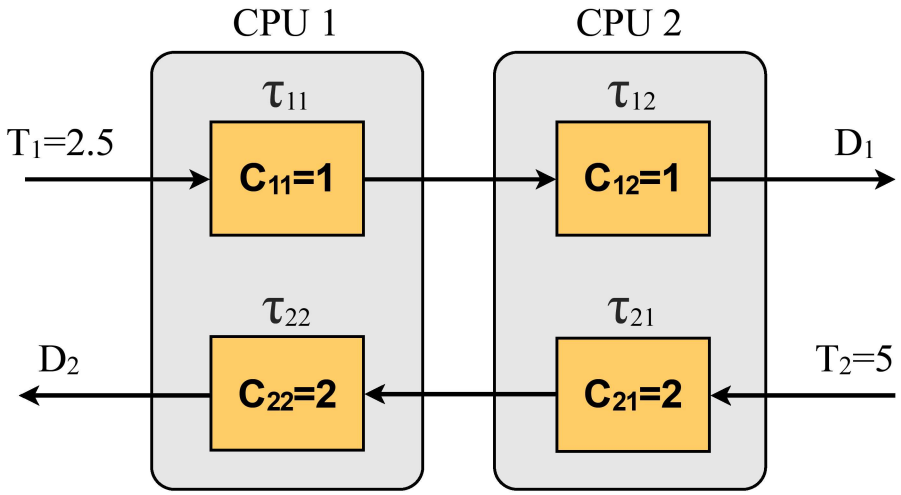
Example for PD	4	7	5	9
UD might be better!	25	25	25	25

- strange behavior: *"More haste less speed"*
  - higher scheduling deadlines produce lower response times
  - scheduling deadlines are just numbers, not timing requirements

A paradox: simple example

Timing requirements

	Scn1	Scn2
D1	5	10
D2	10	20



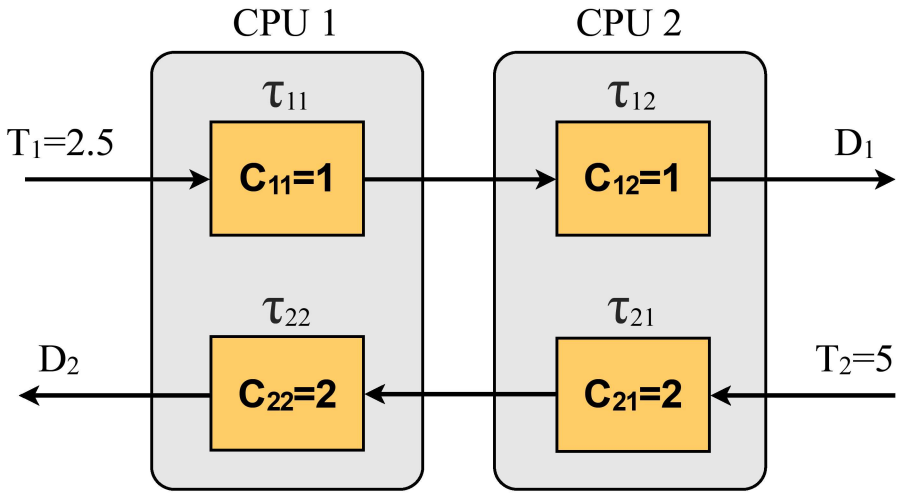
PD assignment
Analysis results

	Scn1		Scn2	
	Sd	R	Sd	R
τ_{11}	2.5		5	
τ_{12}	2.5		5	
τ_{21}	5		10	
τ_{22}	5		10	

A paradox: simple example

Timing requirements

	Scn1	Scn2
D1	5	10
D2	10	20

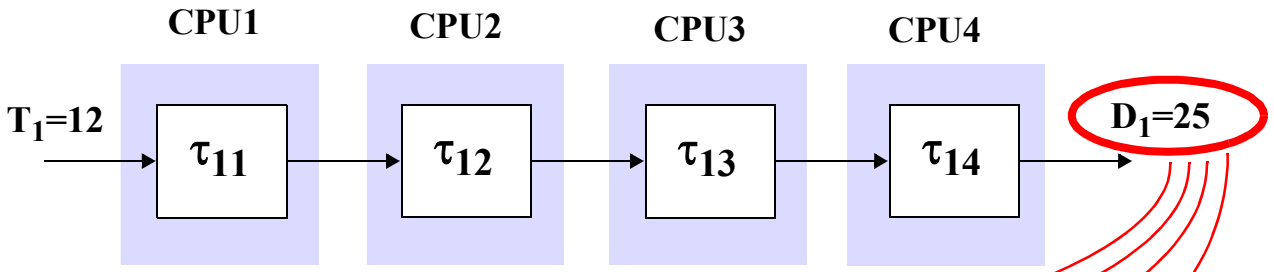


PD assignment
Analysis results

	Scn1		Scn2	
	Sd	R	Sd	R
τ_{11}	2.5	**3.5**	5	**1**
τ_{12}	2.5	**5**	5	**2**
τ_{21}	5	**5**	10	**4**
τ_{22}	5	**9**	10	**8**

A paradox: new assignment algorithm

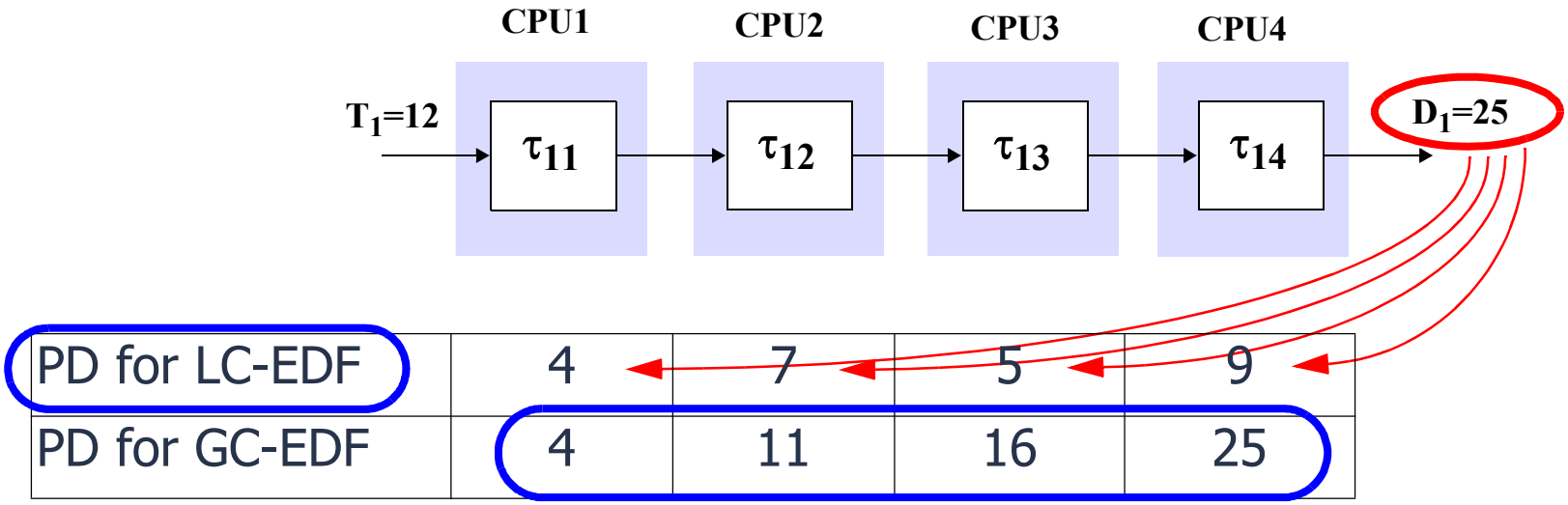
Another kind of algorithm should be explored



PD for LC-EDF	4	7	5	9
PD for GC-EDF	4	11	16	25

A paradox: new assignment algorithm

Another kind of algorithm should be explored



PD for LC-EDF	4	7	5	9
PD for GC-EDF	4	11	16	25

- Proportional Deadline with Global Scheduling Deadline (PD-GSD): uses the PD assignment obtained for GC-EDF
  - similar performance for LC-EDF to FP

Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

Objectives

To contrast whether the behavior of LC-EDF predicted by theory under different scheduling strategies can be observed in practice:

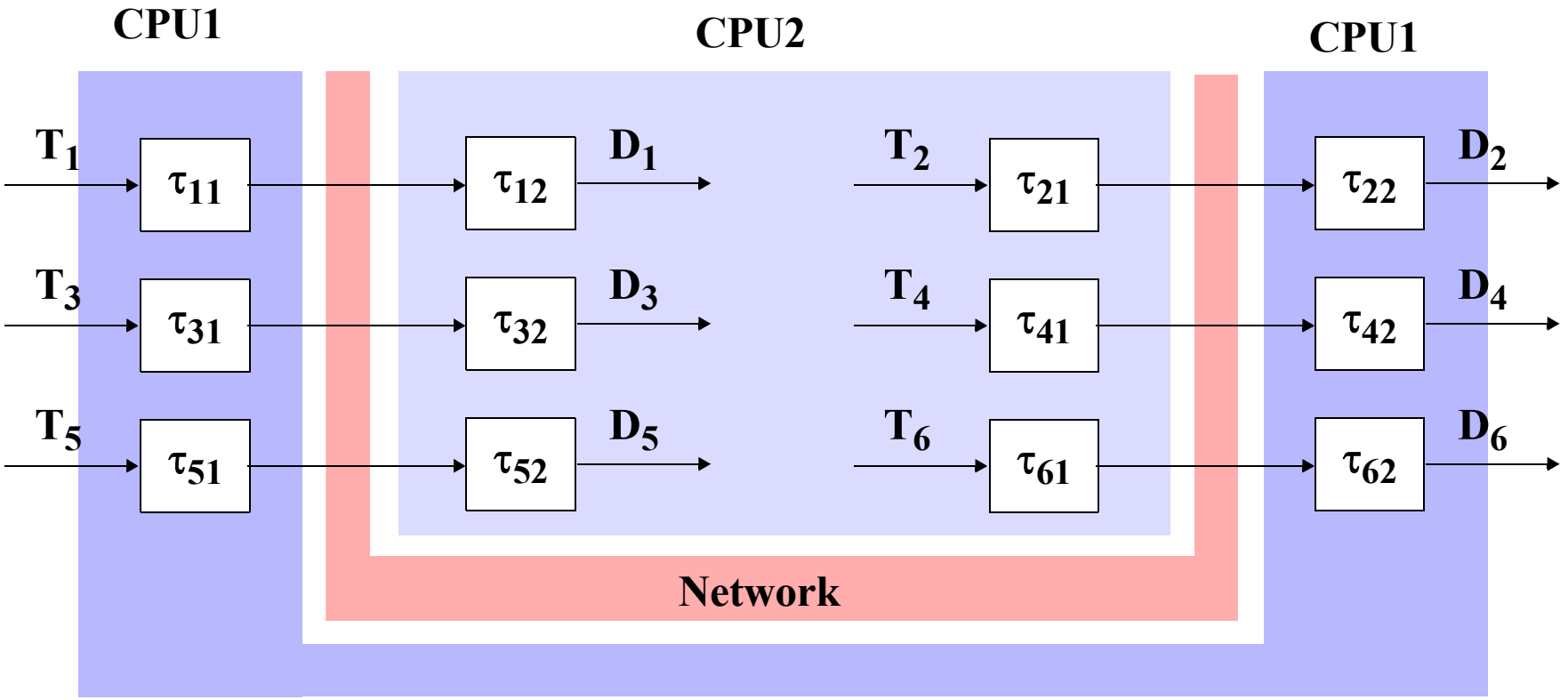
- we do not have a formal demonstration of the LC-EDF behavior
- we want to estimate trends of response times in a practical implementation
- scheduling deadline assignment: UD, ED, PD and PD-GSD

To verify whether Ada is able to support the EDF policy for distributed real-time systems when a global clock reference is not available

Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

A reference example



A simple application to enable understanding of the results obtained

- complex enough for these results to be representative

A reference example (cont'd)

Execution times, periods, deadlines and message sizes chosen to minimize the impact of overheads:

- operating system, timing control and communications

Two configurations with different CPU utilizations:

- 85% and 90%
- the network utilization is less than 0.2%
  - it will enable simplification of the modeling, the schedulability analysis and the time measurements

Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

The Ada implementation

Ada tasks over MaRTE OS for x86

- emulation of CPU load by using CPU clocks

Two kinds of application tasks:

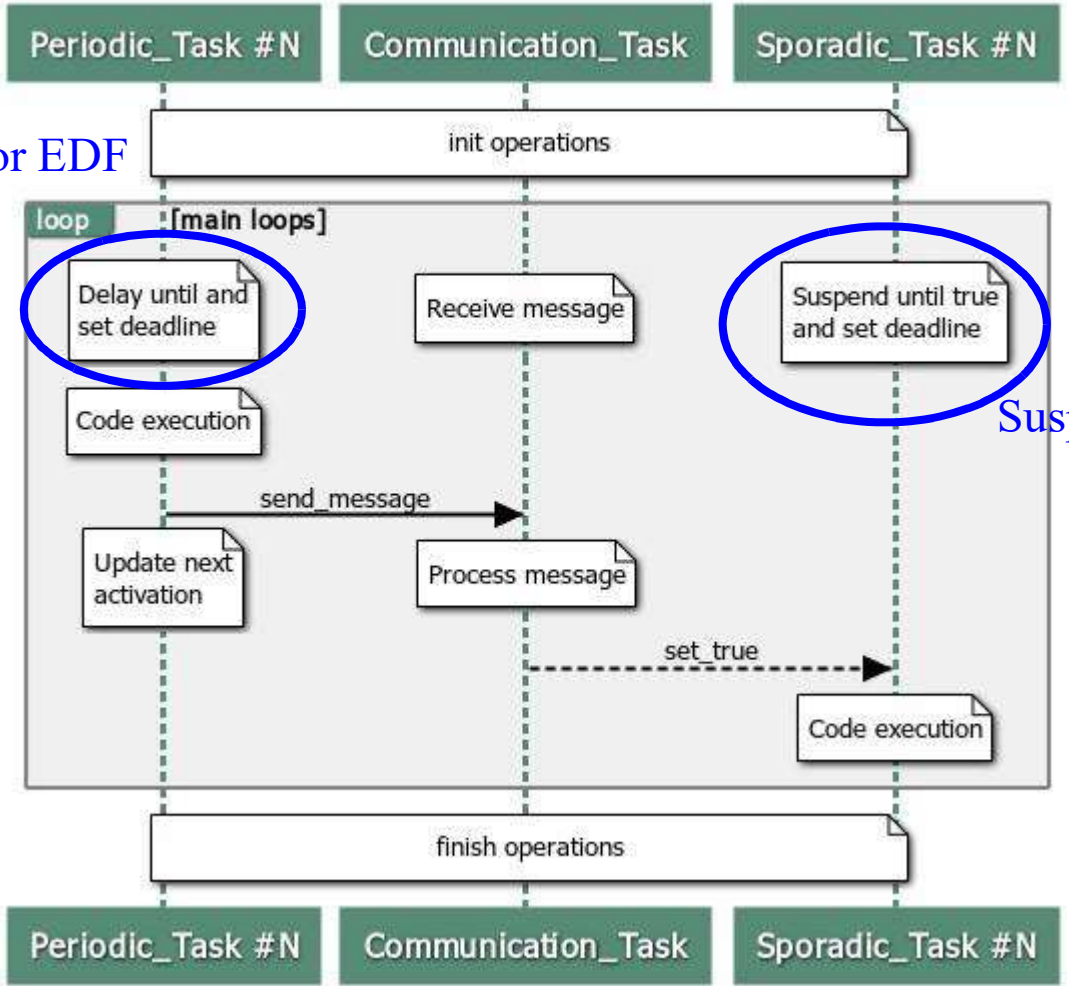
- Periodic tasks: first part of the e2e flow; execute the emulated code and send a message to activate the remote task
- Sporadic tasks: wait for the arrival of the events that start the execution of the remote part of the e2e flow

Communication task waiting for the arrival of messages to:

- signal the corresponding sporadic task which is waiting on a suspension object

Sequence diagram of an e2e flow

Ada dispatching facilities for EDF
- EDF_Across_Priorities



Suspension object

- Suspension objects do not allow nested activations

Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

Theoretical results

We apply MAST (Modeling and Analysis Suite for Real-Time Applications) to the reference example

We assume that the messages sent through the network are instantaneous (we do not have an LC-EDF-compliant network available):

- we will not model them
  - as execution times and periods have been chosen to minimize the effect of communications

Worst-case response times are calculated for the e2e flows in both configurations and for all the scheduling deadline assignment schemes

Theoretical results (cont'd)

The analytical results agree with those obtained in previous works:

- Only the PD-GSD technique enables the e2e deadlines to be met for all the e2e flows in both configurations
- PD-GSD outperforms the worst-case response times obtained by PD for the end-to end flows with larger deadlines, without jeopardizing those with shorter deadlines
- PD obtains higher response times in general than the other techniques
- UD and ED do not allow the schedulability of both configurations, but they obtain the lowest worst-case response times for the end-to end flows with shorter deadlines

Practical results

The objective is to measure the best, average and worst response times for the e2e flows

- worst-case response times can be compared to the analytical results
- best and average response times can give a different view of the performance of the different algorithms

We wanted to obtain these measurements on the emulated Ada application, but

- we could not get a stable version of MaRTE OS and the GNAT-GPL-2014 compiler with the EDF facilities needed

Practical results

The objective is to measure the best, average and worst response times for the e2e flows

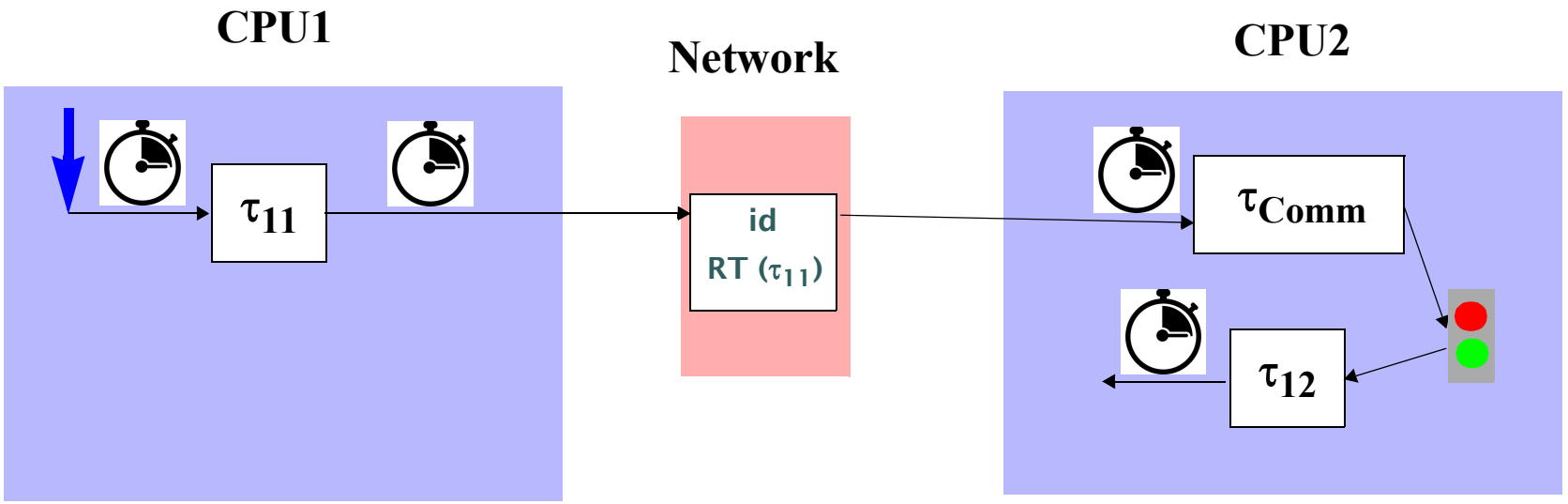
- worst-case response times can be compared to the analytical results
- best and average response times can give a different view of the performance of the different algorithms

We wanted to obtain these measurements on the emulated Ada application, but

- we could not get a stable version of MaRTE OS and the GNAT-GPL-2014 compiler with the EDF facilities needed

An alternative (less elegant) C program has been used

Time measurements



Times are measured separately with the local clocks:

- the response time of the periodic task is sent within the message to be added to the response time of the remote task
- the communication time is not included in the response time of the e2e flow
  - response times might be underestimated

Observations on the results

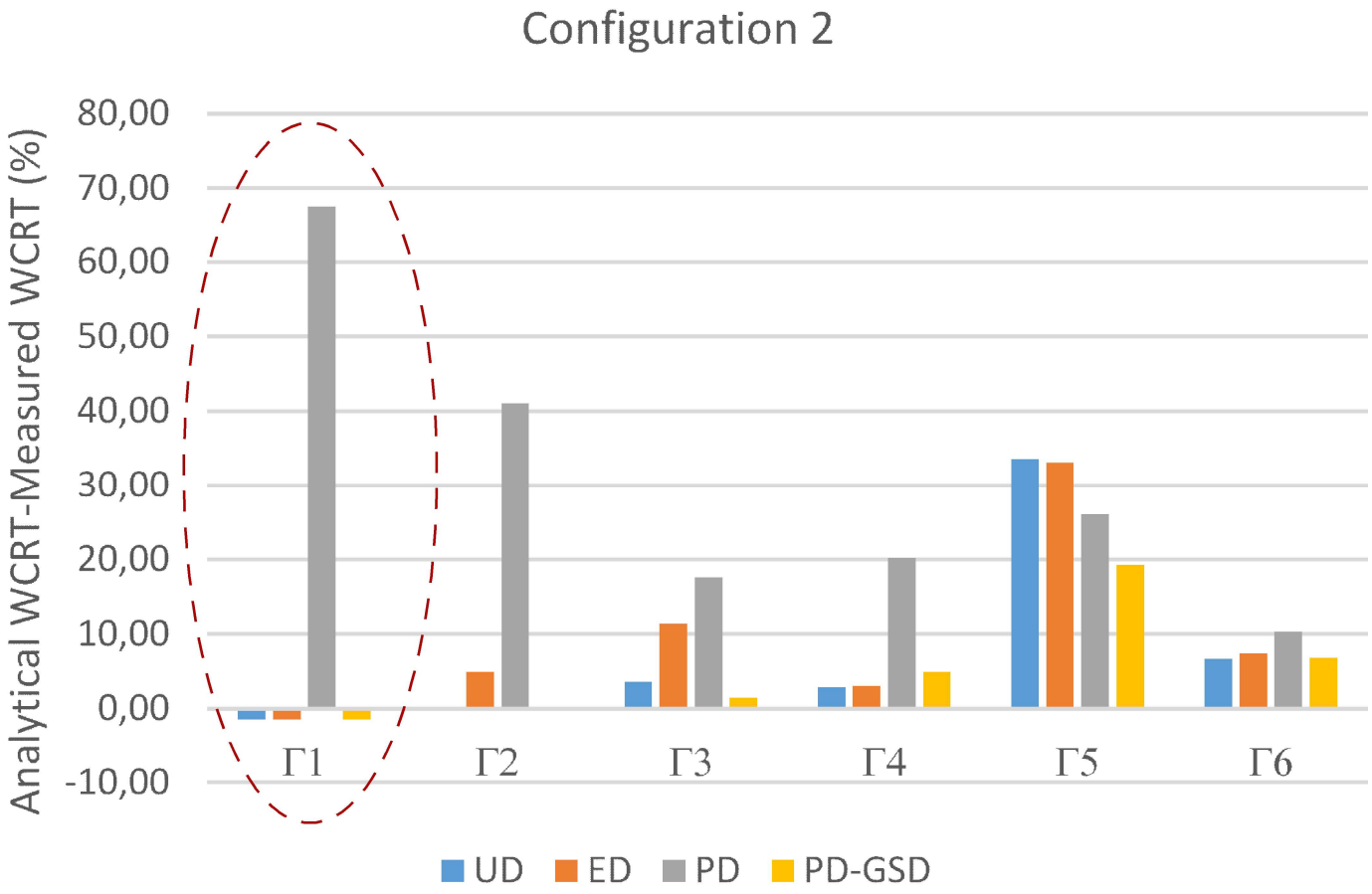
We cannot obtain a general behavior pattern for the average case

Similar best-case response times have been obtained by all the assignment techniques in both configurations

The response time analysis technique used is pessimistic (upper bounds of the worst-case response times), and it is also probable that we have not found the worst-case situation in the execution of the application

- ED and UD assignments do not make the system schedulable in Configuration 2 as predicted by theory
- PD-GSD is the technique that produces tighter worst-case response times and all of them are within the deadlines
- the difference is low for the e2e flows with shorter deadlines
  - except for PD in Configuration 2: particularly pessimistic results that cannot be observed in the real execution

Observations on the results (cont'd)



Outline

1. A short story
   - The context
   - The starting point
   - A paradox
2. Objectives
3. A reference example
4. The Ada implementation
5. Some results
6. Concluding remarks

Concluding remarks

As a general conclusion about the results on the worst-case response times:

- *PD-GSD* balances the worst-case response times of all the end-to end flows allowing deadlines to be met --> *it is the best technique*
- *UD* and *ED* benefit e2e flows with shorter deadlines to the detriment of the other ones --> *they are not a good option in general*
- PD performs worse than PD-GSD but it does not show the radically different behavior shown by theory

Concluding remarks

As a general conclusion about the results on the worst-case response times:

- *PD-GSD* balances the worst-case response times of all the end-to end flows allowing deadlines to be met --> *it is the best technique*
- *UD* and *ED* benefit e2e flows with shorter deadlines to the detriment of the other ones --> *they are not a good option in general*
- PD performs worse than PD-GSD but it does not show the radically different behavior shown by theory

Working with the EDF policy in distributed real-time systems

- a little bit like a nightmare

Thanks for your attention!