Department of Mathematics, University of Padua

Ravenscar-EDF

Comparative Benchmarking of an EDF Variant of a Ravenscar Runtime

Ada Europe 2017
22nd Int'l Conference on Reliable Software Technologies

Paolo Carletto: carletto.paolo@gmail.com
Tullio Vardanega: tullio.vardanega@math.unipd.it

June 13, 2017

Outline of the Talk

- **1973**: Liu and Layland address the question of which online (preemptive) scheduling policy for single-core processors is theoretically the best

- **1973**: Liu and Layland address the question of which online (preemptive) scheduling policy for single-core processors is theoretically the best
- Nonetheless, most existing real-time kernels use FPS
  - Easier to implement
  - Simpler to understand and supposedly more "stable"

- **1973**: Liu and Layland address the question of which online (preemptive) scheduling policy for single-core processors is theoretically the best
- Nonetheless, most existing real-time kernels use FPS
  - Easier to implement
  - Simpler to understand and supposedly more "stable"
- **2005**: Buttazzo sustains the superiority of EDF also from a practical standpoint

- **1973**: Liu and Layland address the question of which online (preemptive) scheduling policy for single-core processors is theoretically the best
- Nonetheless, most existing real-time kernels use FPS
  - Easier to implement
  - Simpler to understand and supposedly more "stable"
- **2005**: Buttazzo sustains the superiority of EDF also from a practical standpoint
- **2017**: We present an empirical, quantitative comparison between concrete implementations of EDF and FPS

The Benchmark

1. It makes a very fair comparison between runtimes
   - We changed *only* the scheduling operations

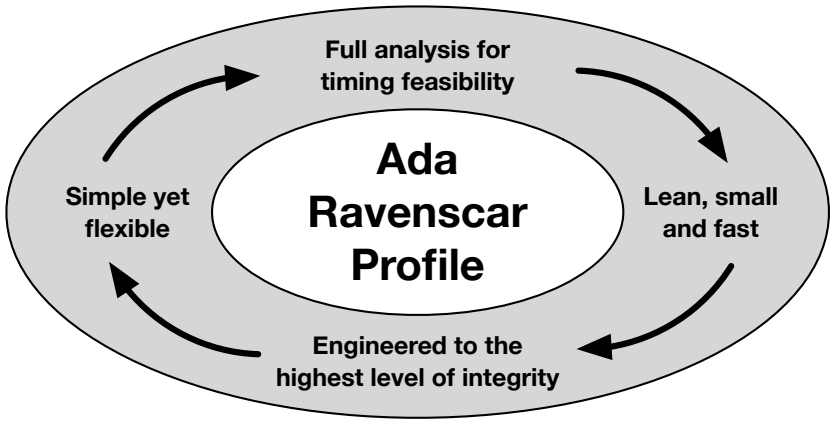
The Benchmark

1. It makes a very fair comparison between runtimes
   - We changed *only* the scheduling operations
2. The application stays unchanged
   - Any performance difference is directly ascribable to the scheduler

The Benchmark

1. It makes a very fair comparison between runtimes
   - We changed *only* the scheduling operations
2. The application stays unchanged
   - Any performance difference is directly ascribable to the scheduler
3. We stressed each system to the theoretical limit discussed in the literature
   - Using exactly the same, unchanged, application software
   - The switch of scheduling policy is completely transparent to it

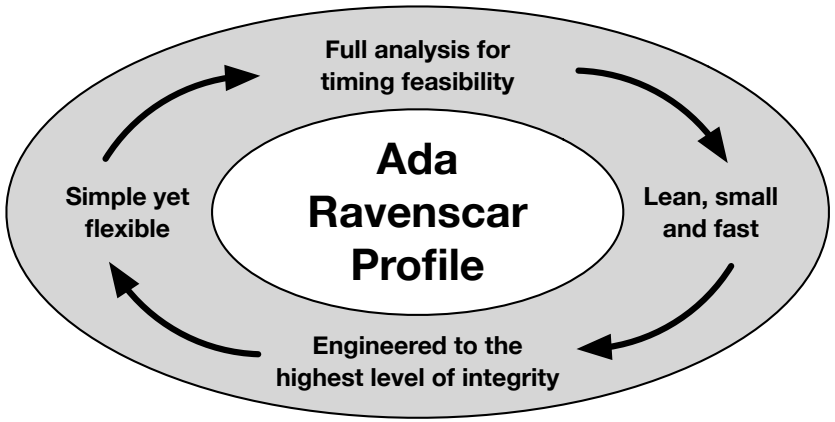
- The Ravenscar profile is an important asset of the Ada programming language



The RM-to-EDF Transformation Process

The Ada Ravenscar Profile

- The Ravenscar profile is an important asset of the Ada programming language

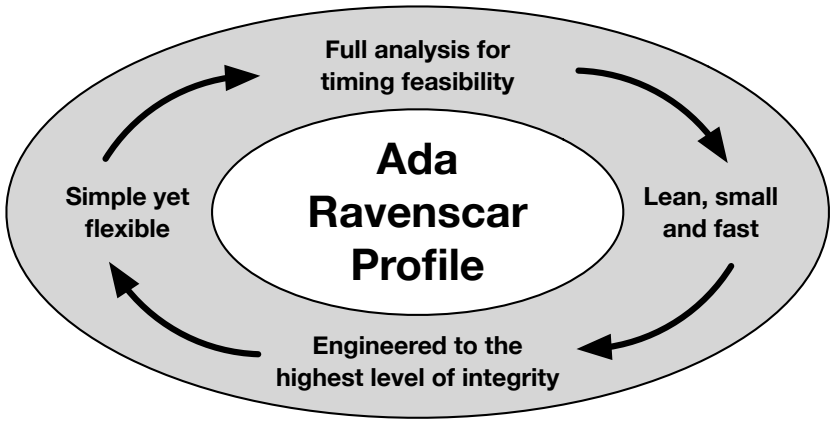


- The first-ever Ravenscar runtime to be released for industrial use was produced by AdaCore for the LEON processor family

The RM-to-EDF Transformation Process

The Ada Ravenscar Profile

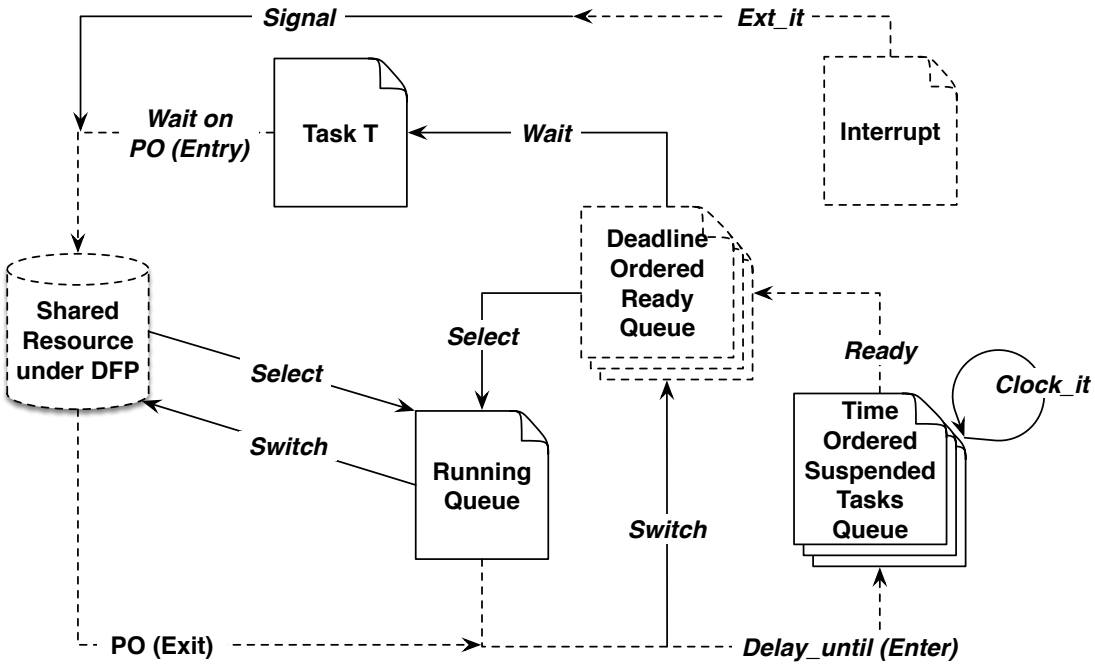
- The Ravenscar profile is an important asset of the Ada programming language



- The first-ever Ravenscar runtime to be released for industrial use was produced by AdaCore for the LEON processor family
- That technology originated from a fork of the Open Ravenscar Real-Time Kernel (ORK+) developed by the Technical University of Madrid for the European Space Agency (ca. 2000)

The RM-to-EDF Transformation Process

Turning Priorities into Deadlines



1. **Task Dispatching Policy**: from *"FIFO Within Priorities"* to *"EDF"*[1]
2. **Locking Policy**: from IPCP to DFP[2]

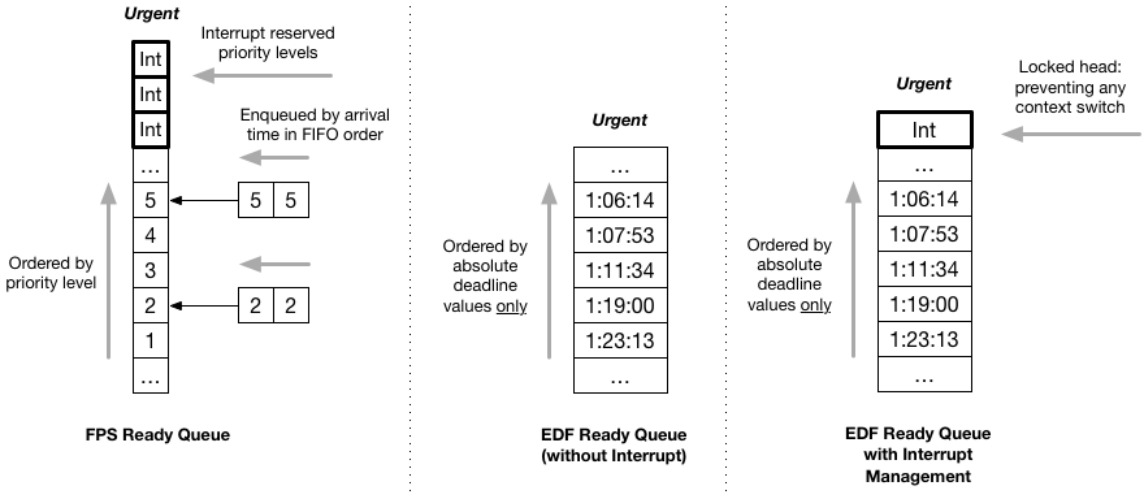
[1] A. Burns, An EDF Runtime Profile based on Ravenscar. Ada Lett. 33, 1 (June 2013)

[2] A. Burns and A. Wellings. The Deadline Floor Protocol and Ada. Ada Lett. 36, 1 (July 2016)

The RM-to-EDF Transformation Process

Implementation Challenges

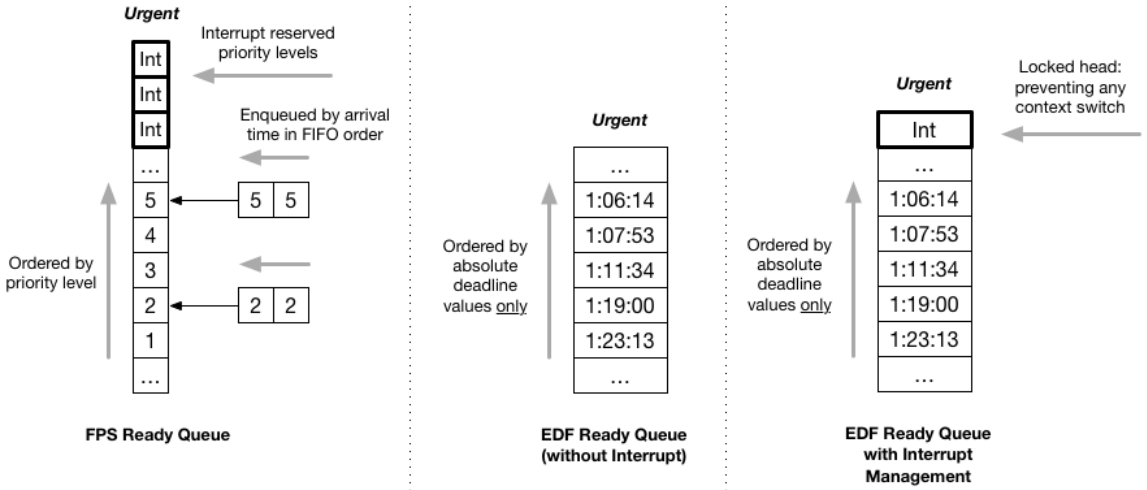
Interrupt handling intrinsically assumes priorities, which – in principle – do not belong in an EDF system



The RM-to-EDF Transformation Process

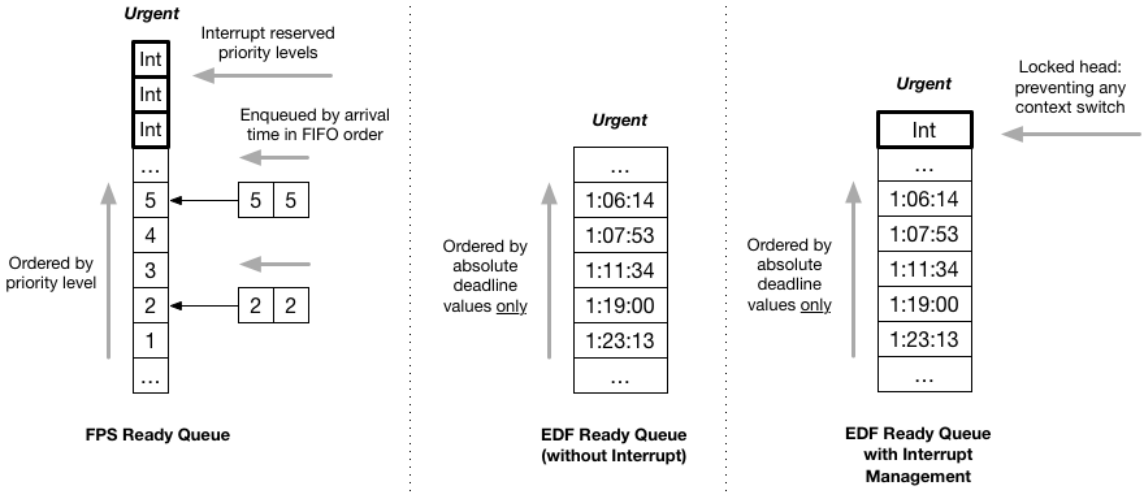
Implementation Challenges

Interrupt handling intrinsically assumes priorities, which – in principle – do not belong in an EDF system



- Our solution reserves a fictitious position at the top of the ready queue for the current interrupt handler
  - If an interrupt handler is active, that position is used and the deadline-based part of the queue is frozen

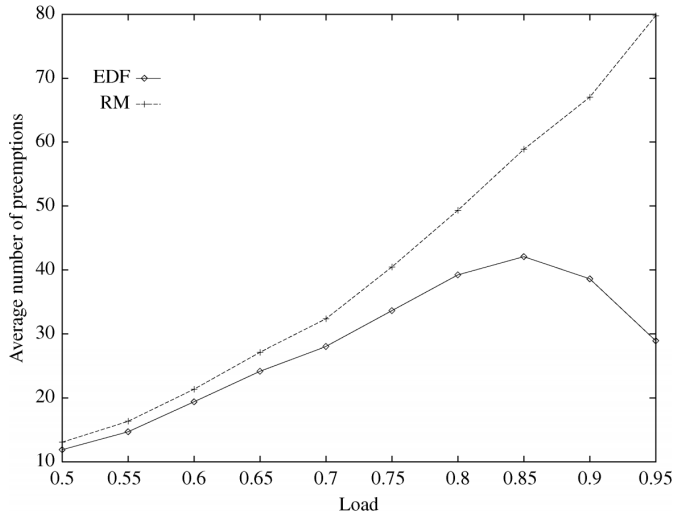
Interrupt handling intrinsically assumes priorities, which – in principle – do not belong in an EDF system



- Our solution reserves a fictitious position at the top of the ready queue for the current interrupt handler
  - If an interrupt handler is active, that position is used and the deadline-based part of the queue is frozen
  - If no interrupt is running, that position is not in use and cannot be contended

Buttazzo claimed EDF better than RM (FPS) in many respects

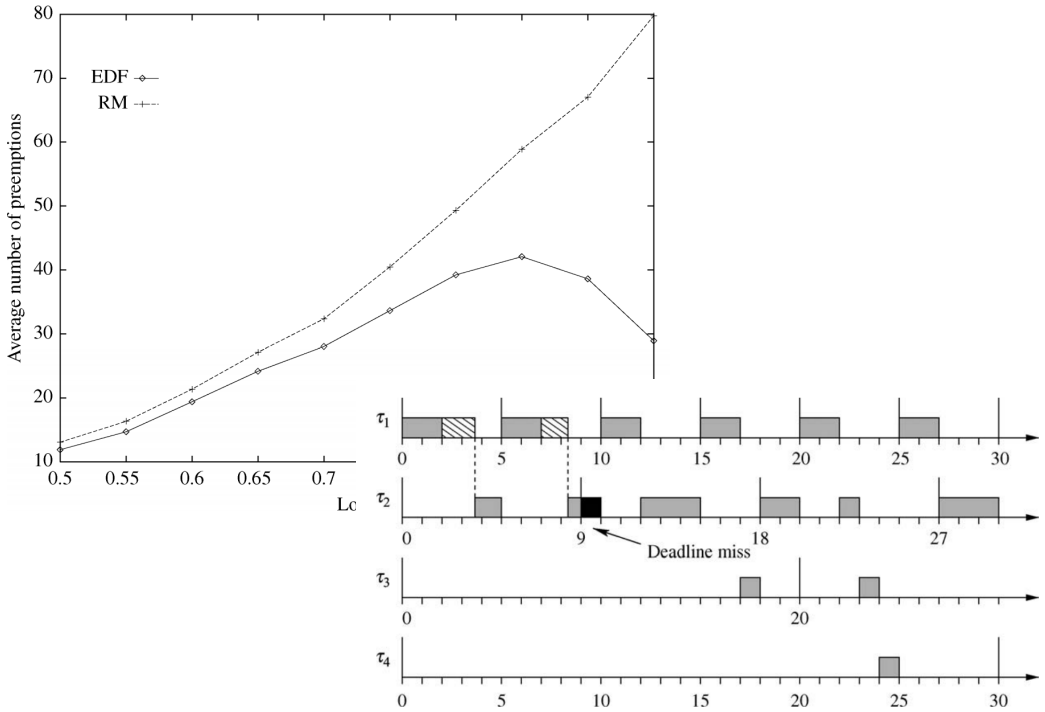
- Lower runtime overhead
  - Less preemptions
- Easier analysis
- More robust under overloads
  - Transient
  - Permanent



Evaluation Results
Buttazzo's Analysis

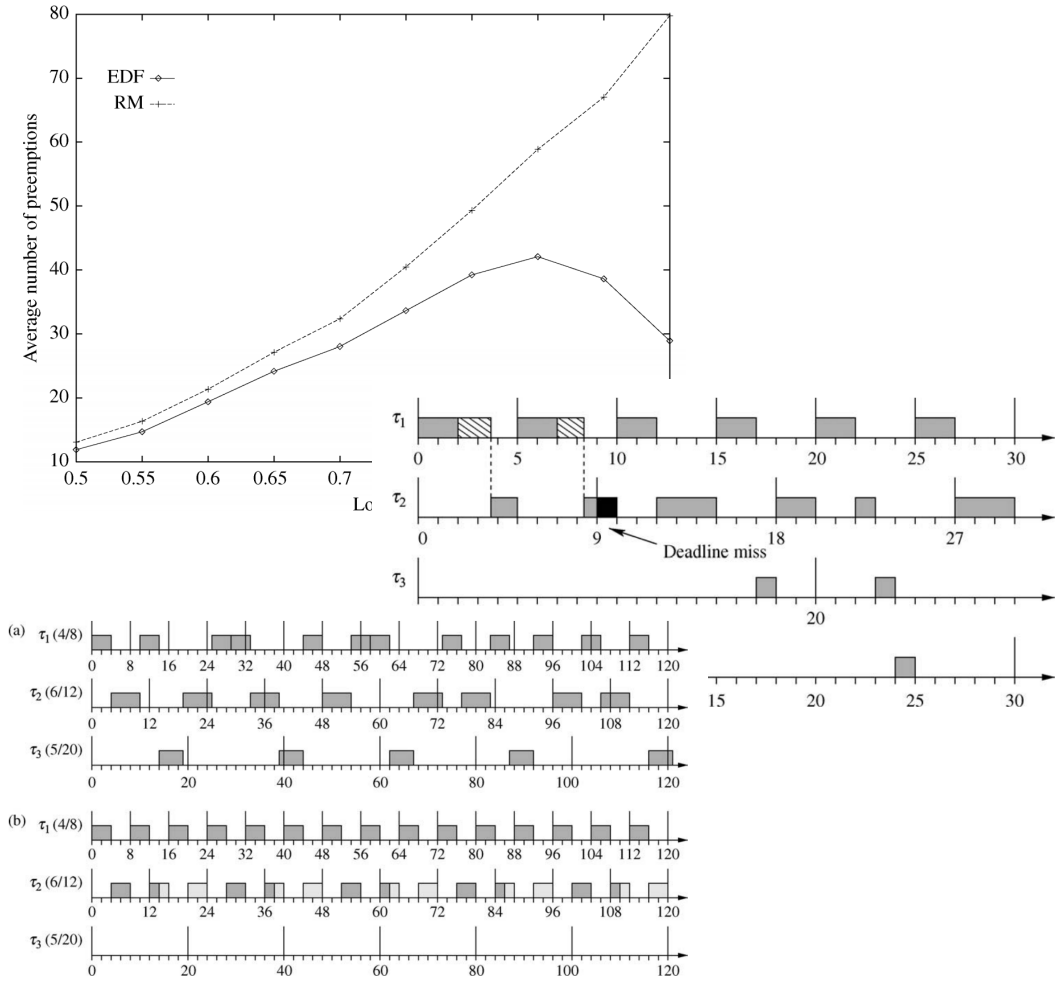
Buttazzo claimed EDF better than RM (FPS) in many respects

- Lower runtime overhead
  - Less preemptions
- Easier analysis
- More robust under overloads
  - Transient
  - Permanent



Buttazzo claimed EDF better than RM (FPS) in many respects

- Lower runtime overhead
  - Less preemptions
- Easier analysis
- More robust under overloads
  - Transient
  - Permanent



What is weak in Buttazzo's analysis?

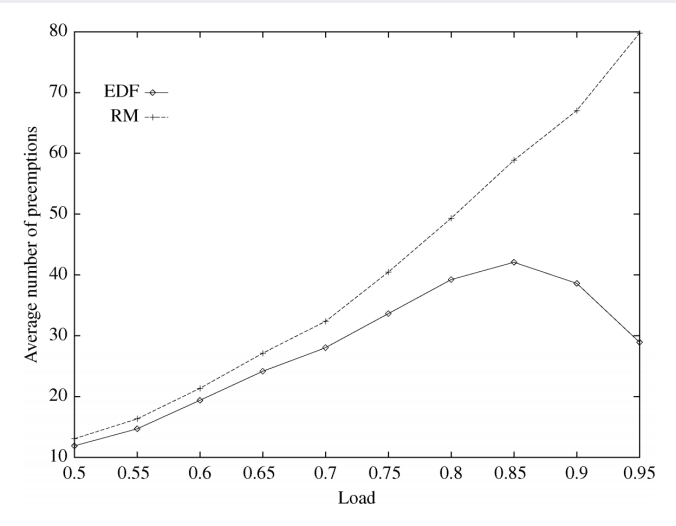
- Task cardinality too small (10-30 tasks) to be significant

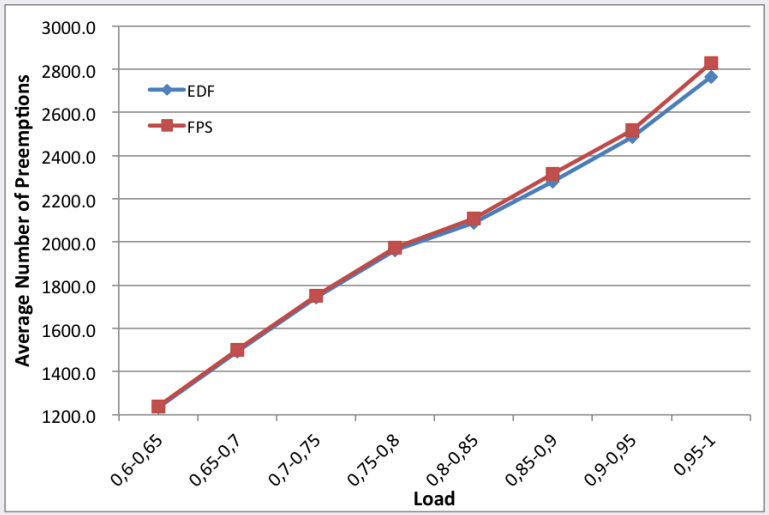
What is weak in Buttazzo's analysis?

- Task cardinality too small (10-30 tasks) to be significant
- Overload analysis confined to specific cases and not sufficiently general

What is weak in Buttazzo's analysis?

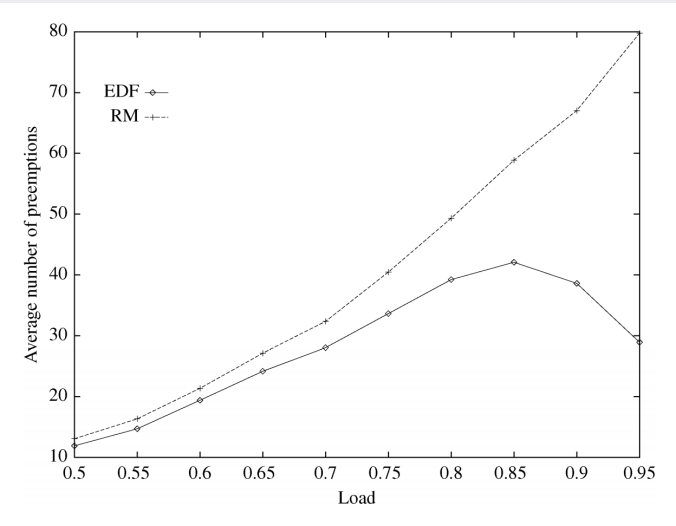
- Task cardinality too small (10-30 tasks) to be significant
- Overload analysis confined to specific cases and not sufficiently general
- Different preemption behavior observed under 100%

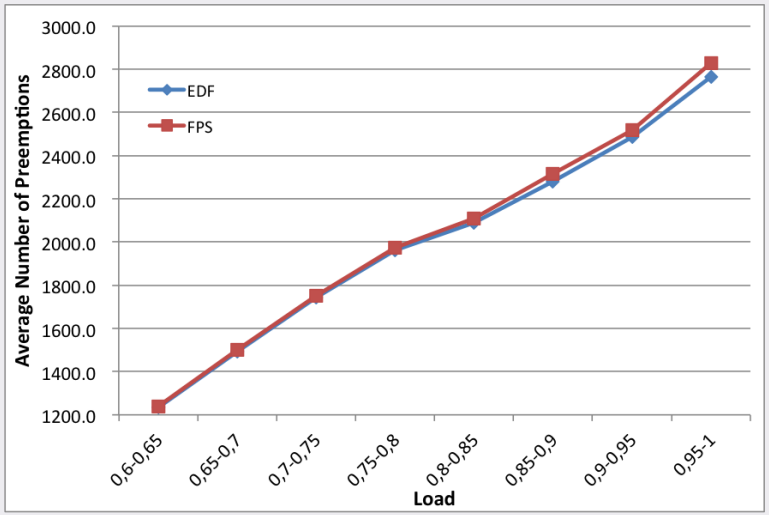




What is weak in Buttazzo's analysis?

- Task cardinality too small (10-30 tasks) to be significant
- Overload analysis confined to specific cases and not sufficiently general
- Different preemption behavior observed under 100%





- Lack of practical implementation and analysis of resource sharing protocols

Evaluation Results
Highest Schedulable Utilization

Which tasksets achieved the highest schedulable utilization in each runtime variant?

Evaluation Results
Highest Schedulable Utilization

Which tasksets achieved the highest schedulable utilization in each runtime variant?

Taskset Type	Task Types	Delta Schedulable Utilization	Max CPU Load	EDF			FPS		
				RC	DM	PR	RC	DM	PR
Constrained	Short & Mid	**2,89%**	105,50%	30.714	0	3.637	29.850	415	6.202
Implicit	Mid Only	**3,72%**	102,63%	18.691	0	837	18.021	673	2.040
Constrained	All	**0,05%**	104,06%	24.398	0	5.131	24.409	0	5.211
Implicit	All	**5,22%**	100,85%	24.935	953	6.309	26.236	0	5.715

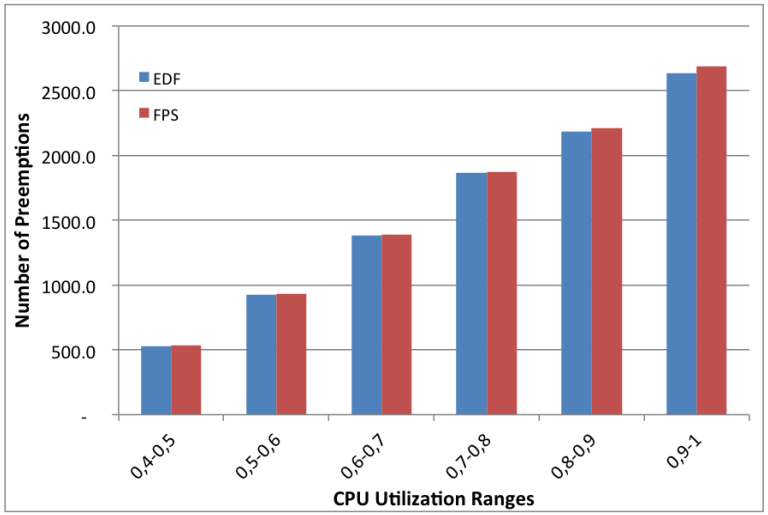
- **RC**: count of regular completions
- **DM**: count of deadline misses
- **PR**: count of preemptions

Evaluation Results
Runtime Overhead

Do the less preemptions and context switches that EDF incurs justify the higher costs of its scheduling operations?

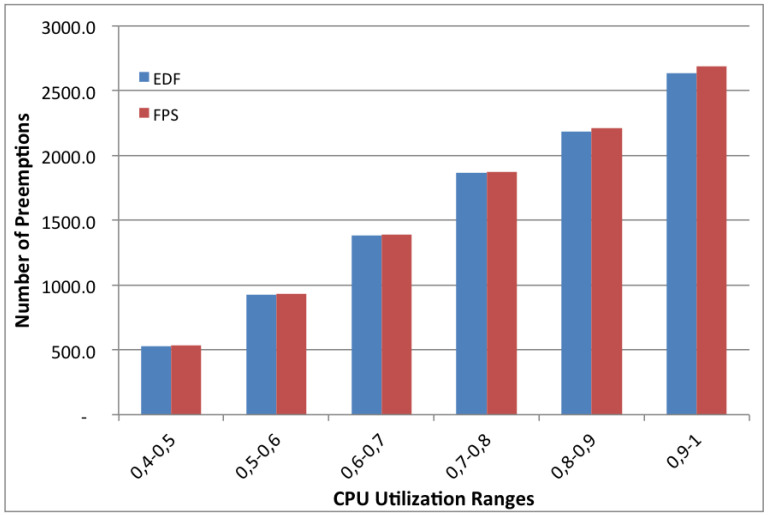
Evaluation Results
Runtime Overhead

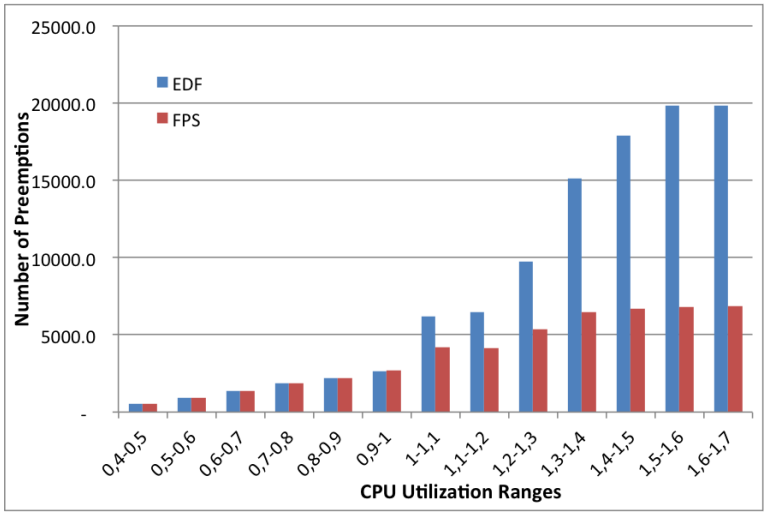
Do the less preemptions and context switches that EDF incurs justify the higher costs of its scheduling operations?



Evaluation Results
Runtime Overhead

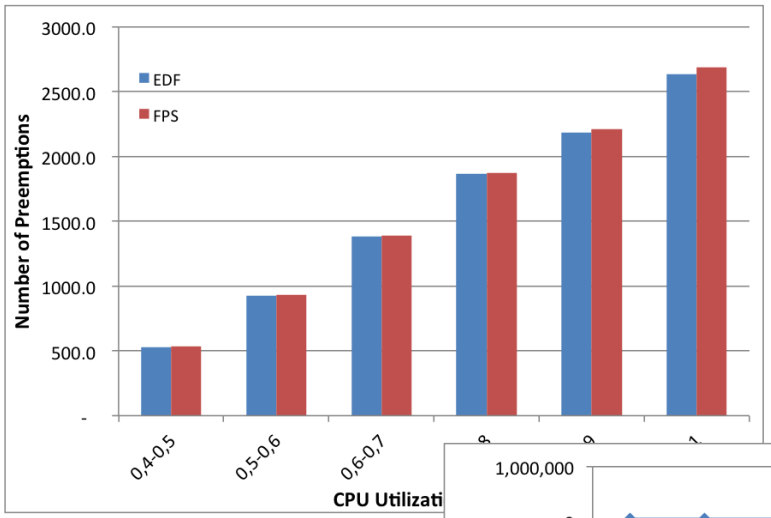
Do the less preemptions and context switches that EDF incurs justify the higher costs of its scheduling operations?

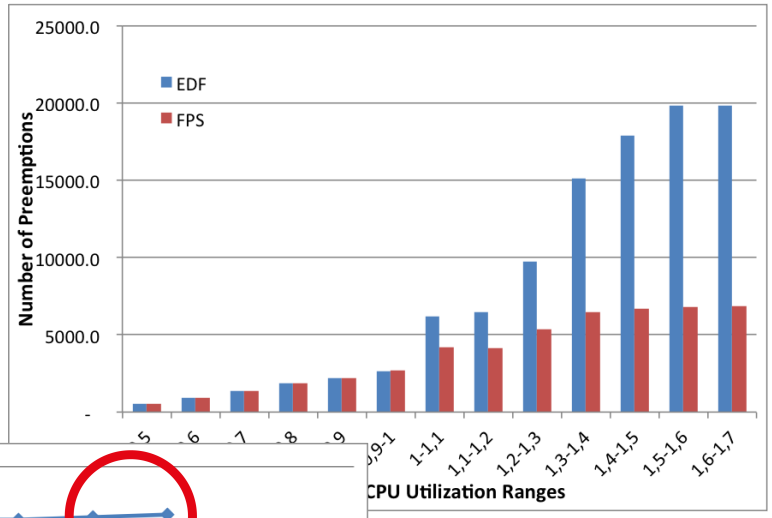


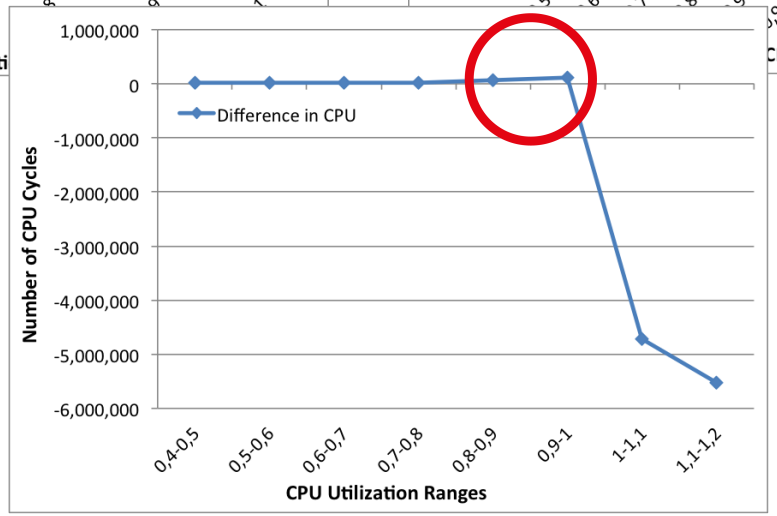


Evaluation Results
Runtime Overhead

Do the less preemptions and context switches that EDF incurs justify the higher costs of its scheduling operations?





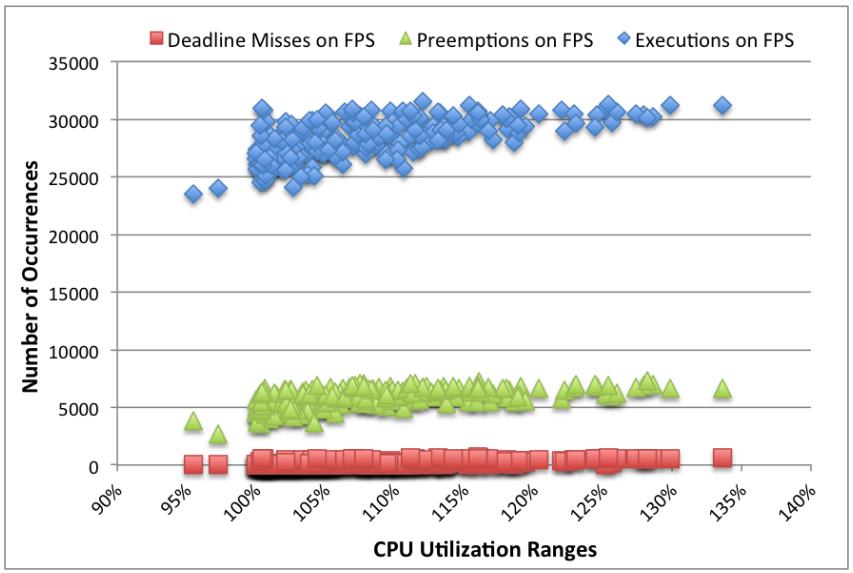


Evaluation Results
Resilience to Overload Situations

What happens to EDF and FPS under overload conditions, when the CPU utilization exceeds 100%?

Evaluation Results
Resilience to Overload Situations

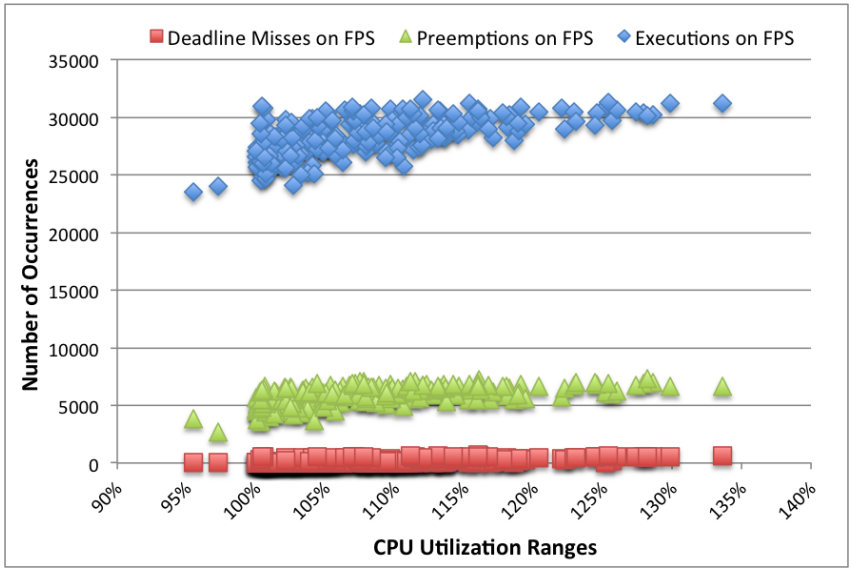
What happens to EDF and FPS under overload conditions, when the CPU utilization exceeds 100%?

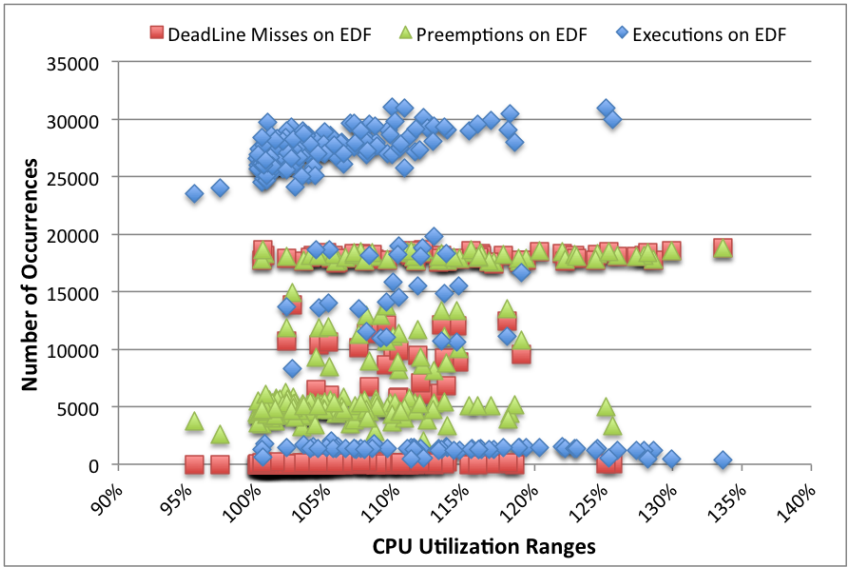


- FPS presents a linear behavior
- EDF's behaviour varies dramatically depending on the nature of the overload situation
  - Transient vs permanent

Evaluation Results
Resilience to Overload Situations

What happens to EDF and FPS under overload conditions, when the CPU utilization exceeds 100%?

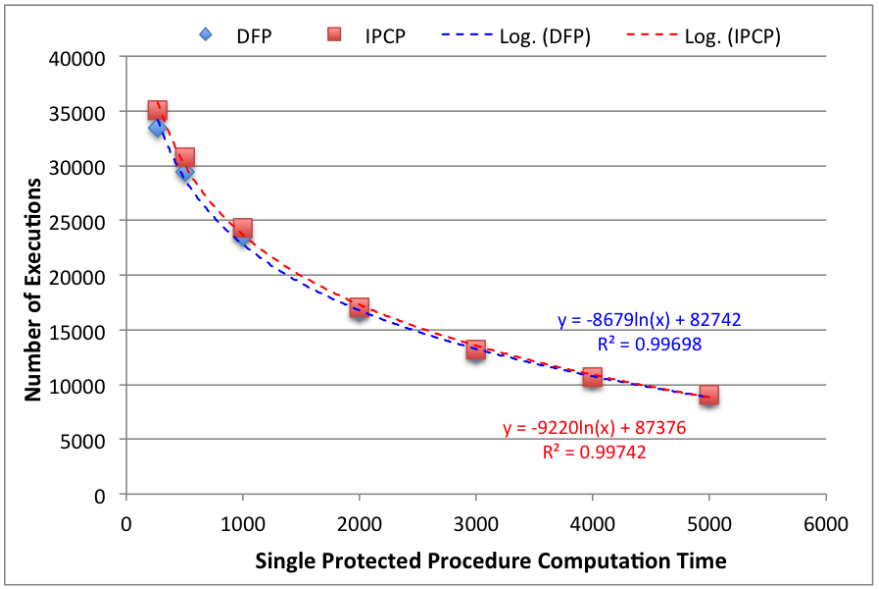




- FPS presents a linear behavior
- EDF's behaviour varies dramatically depending on the nature of the overload situation
  - Transient vs permanent

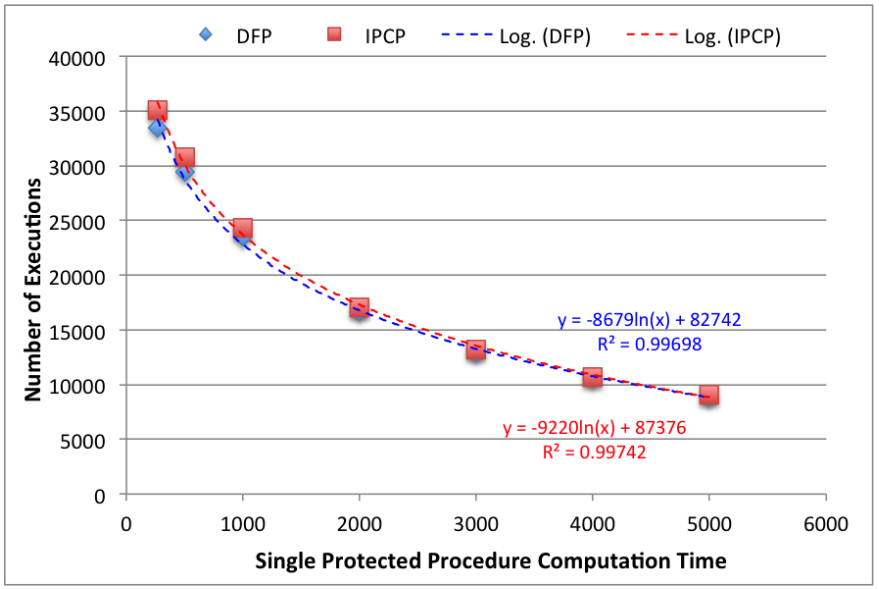
How does DFP perform compared to IPCP?

How does DFP perform compared to IPCP?



- It presents a logarithmic converging progression as the computation time of the protected procedure increases

How does DFP perform compared to IPCP?



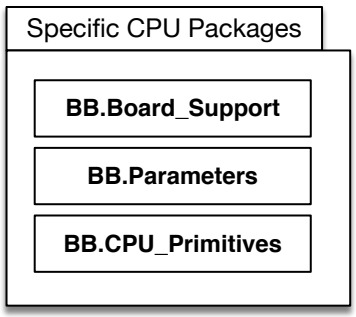
- It presents a logarithmic converging progression as the computation time of the protected procedure increases
- DFP incurs more cumulative overhead than IPCP
  - Due to the need to read the clock in checking absolute deadlines

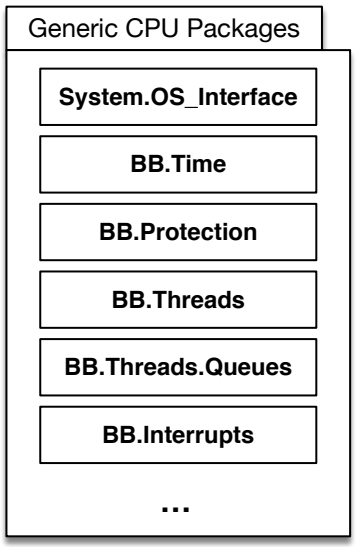
Is it feasible to migrate Ravenscar-EDF from LEON to ARM technology?

- We analyzed the ARM Cortex-M version of the AdaCore runtime in the GAP-2016 offering

Is it feasible to migrate Ravenscar-EDF from LEON to ARM technology?

- We analyzed the ARM Cortex-M version of the AdaCore runtime in the GAP-2016 offering

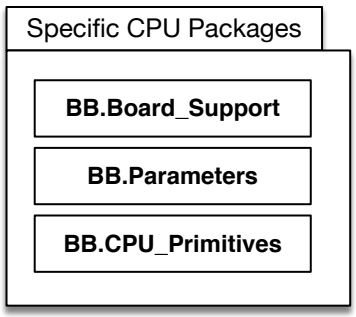


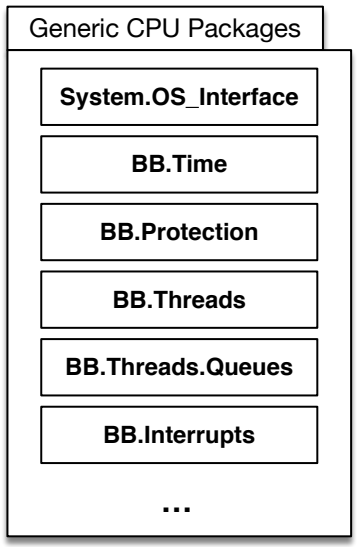


- The FPS version can be easily converted to EDF one modifying only a small part of the original packages

Is it feasible to migrate Ravenscar-EDF from LEON to ARM technology?

- We analyzed the ARM Cortex-M version of the AdaCore runtime in the GAP-2016 offering





- The FPS version can be easily converted to EDF one modifying only a small part of the original packages
- This should yield a runtime with more widespread use than for the LEON processor family

How can we take benefit of the best of both?

- EDF generates a feasible schedule (if any exists) within 100% CPU utilization
- FPS has more resilience beyond 100% CPU utilization

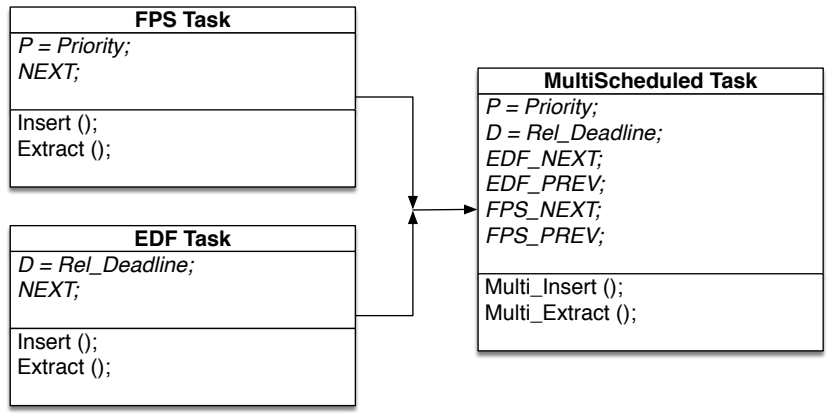
How can we take benefit of the best of both?

- EDF generates a feasible schedule (if any exists) within 100% CPU utilization
- FPS has more resilience beyond 100% CPU utilization

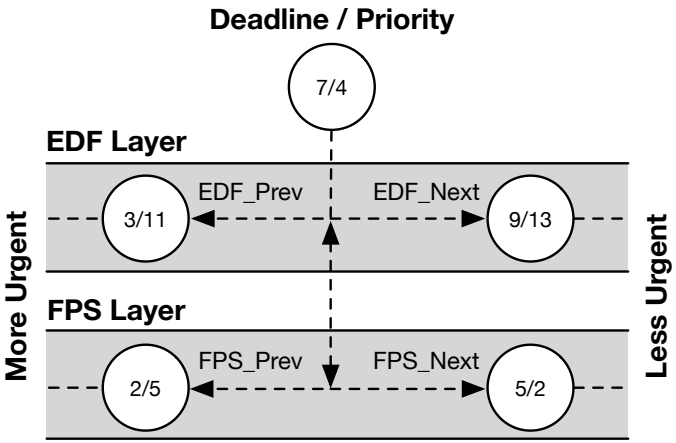
The Solution

A co-existence of both algorithms should yield the best of both worlds: EDF "becomes" FPS above 100% load

- A double linkedlist could offer a quick switch mechanism
- It should be based on a threshold value computed dynamically by the runtime on the idle time

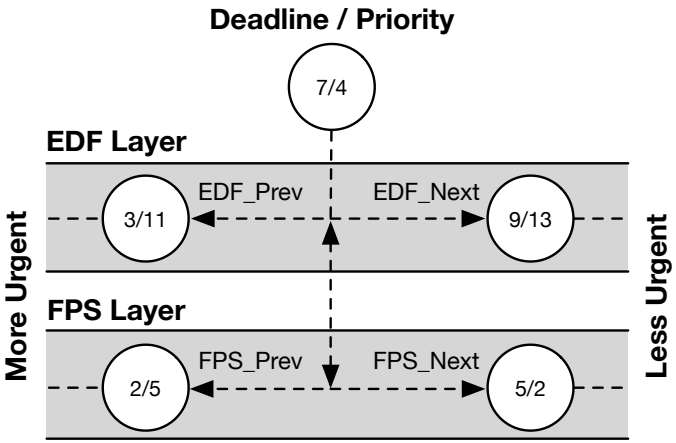


Protocol complexity

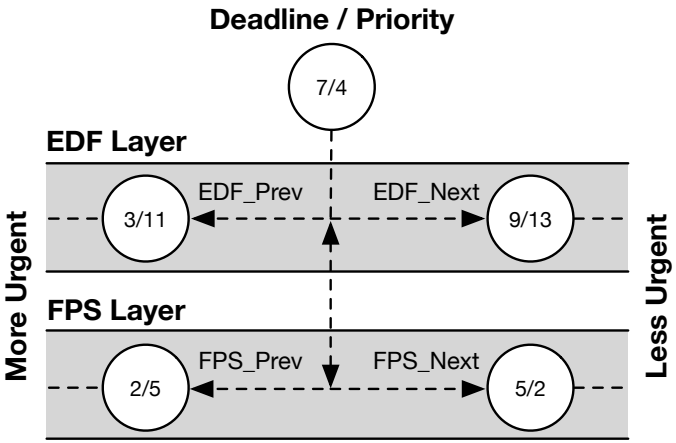


Protocol complexity

- $O(n)$ asymptotically

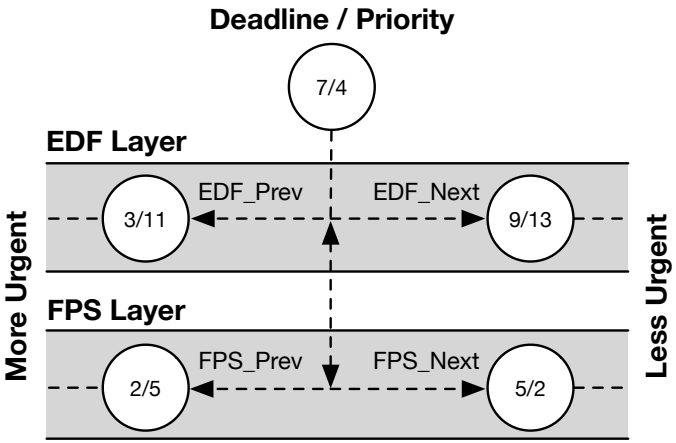


Protocol complexity



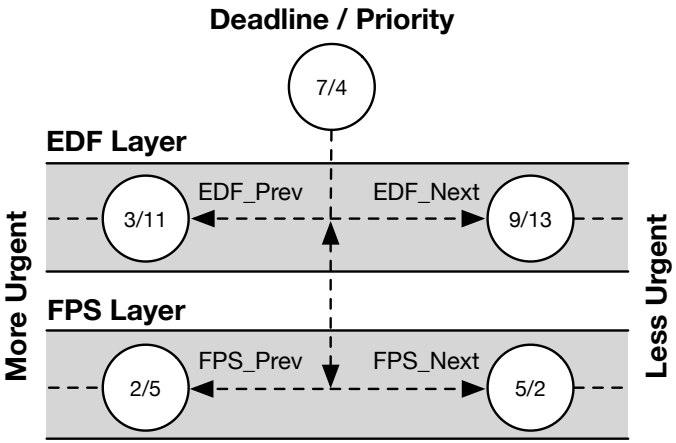
- $O(n)$ asymptotically
- The same limit of a common linked list

Protocol complexity



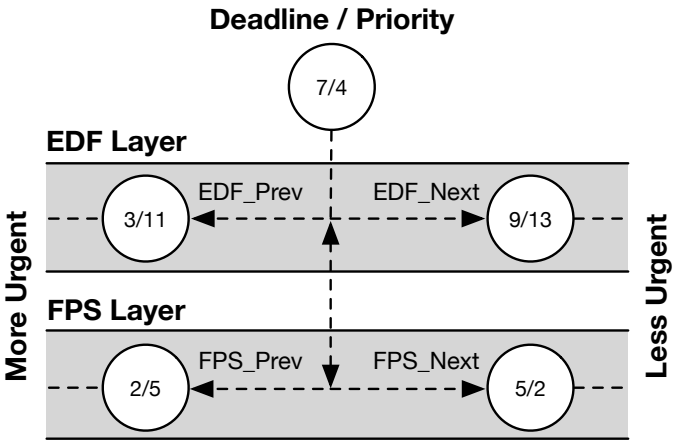
- $O(n)$ asymptotically
- The same limit of a common linked list
- Our Multilayered Scheduling does not increase runtime complexity

Protocol complexity



- $O(n)$ asymptotically
- The same limit of a common linked list
- Our Multilayered Scheduling does not increase runtime complexity
- It earns the benefits of both protocols
  - Max efficiency with CPU load $\leq 100\%$
  - Max stability with CPU load $> 100\%$

Protocol complexity



- $O(n)$ asymptotically
- The same limit of a common linked list
- Our Multilayered Scheduling does not increase runtime complexity
- It earns the benefits of both protocols
  - Max efficiency with CPU load $\leq 100\%$
  - Max stability with CPU load $> 100\%$
- Yet, interrupts could be difficult to manage in a dual scheme

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version
  - Our tests confirmed the theoretical conclusions of earlier works

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version
  - Our tests confirmed the theoretical conclusions of earlier works
  - Yet, we showed that the actual gain of EDF over FPS is far lower than anticipated even for CPU loads very close to 100%, where EDF was due to reap the best of its benefit

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version
  - Our tests confirmed the theoretical conclusions of earlier works
  - Yet, we showed that the actual gain of EDF over FPS is far lower than anticipated even for CPU loads very close to 100%, where EDF was due to reap the best of its benefit
  - We also experimentally observed the fragility of EDF in contrast to the resilience of FPS under overload conditions

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version
  - Our tests confirmed the theoretical conclusions of earlier works
  - Yet, we showed that the actual gain of EDF over FPS is far lower than anticipated even for CPU loads very close to 100%, where EDF was due to reap the best of its benefit
  - We also experimentally observed the fragility of EDF in contrast to the resilience of FPS under overload conditions
- We provided a baseline technology to further investigate this matter

Conclusions

- In this work, we built an experimental framework based on an EDF variant of an Ada Ravenscar runtime and compared it with the original FPS version
  - Our tests confirmed the theoretical conclusions of earlier works
  - Yet, we showed that the actual gain of EDF over FPS is far lower than anticipated even for CPU loads very close to 100%, where EDF was due to reap the best of its benefit
  - We also experimentally observed the fragility of EDF in contrast to the resilience of FPS under overload conditions
- We provided a baseline technology to further investigate this matter
- We are contemplating some hypothesis to combine the best of EDF and FPS in a single runtime