Software Solutions Symposium 2017

March 20–23, 2017

Temporal Partitioning and Verification in Distributed Cyber-Physical Systems

Dionisio de Niz
Bjorn Andersson

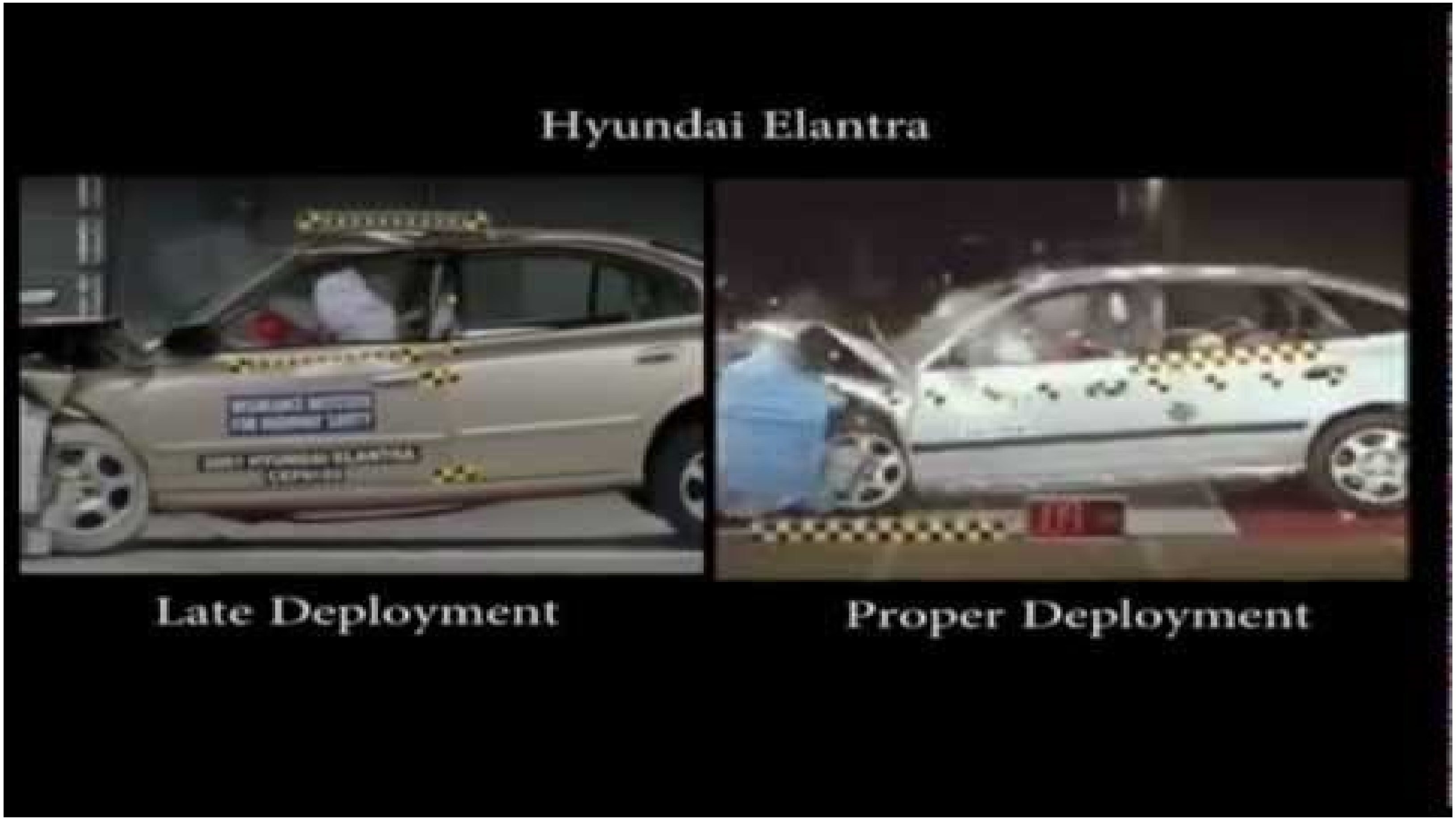
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

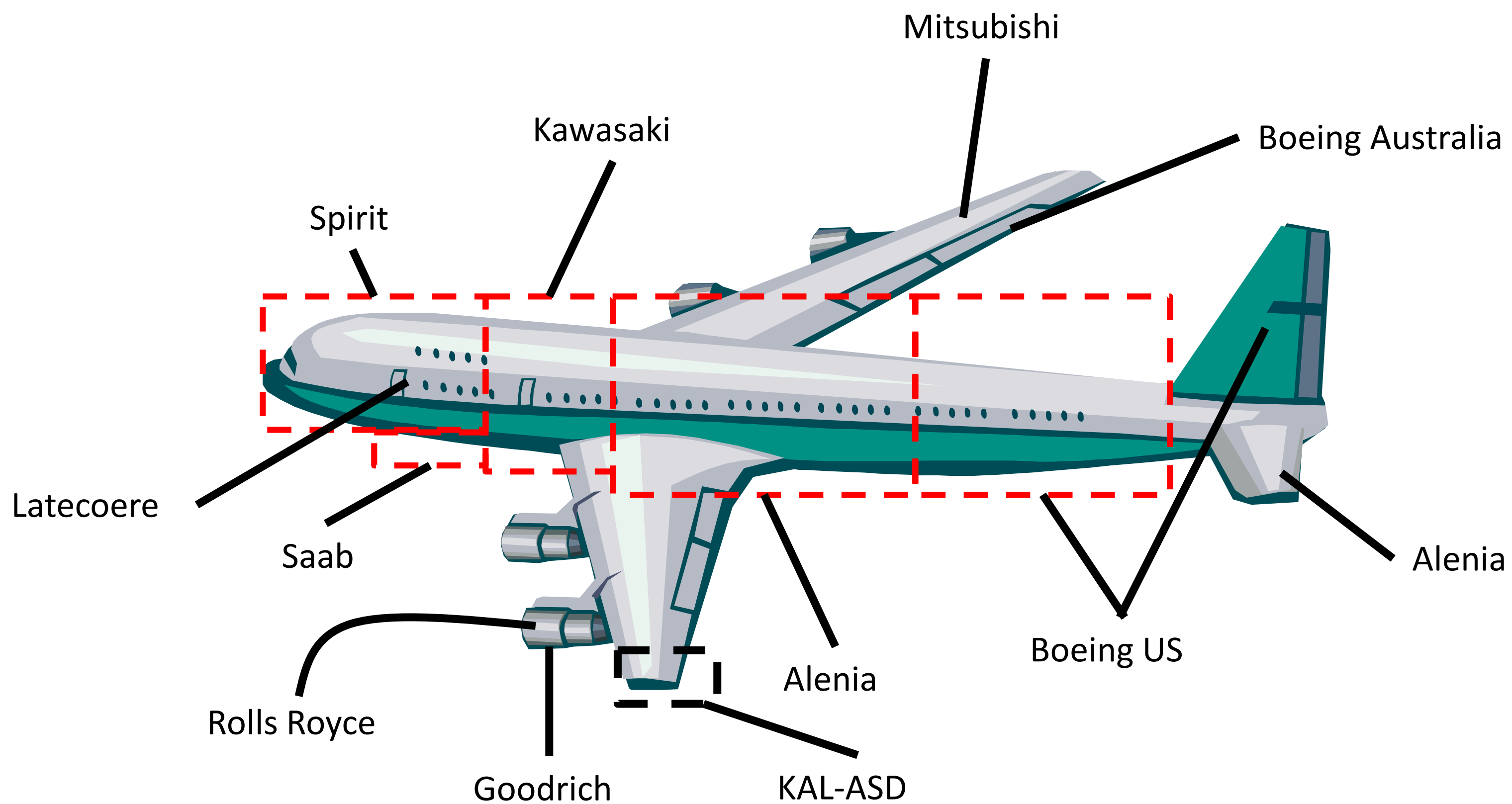
DM17-0005

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Why Is Timing Verification Important?



Why Temporal Protection?
Boeing 787 Suppliers



Source: Boeing / Reuters

FAA Needs Modular Certification

Temporal Partitioning in Distributed CPS

Critical to Verify Timing of CPS

- Late interaction with physical world can be catastrophic

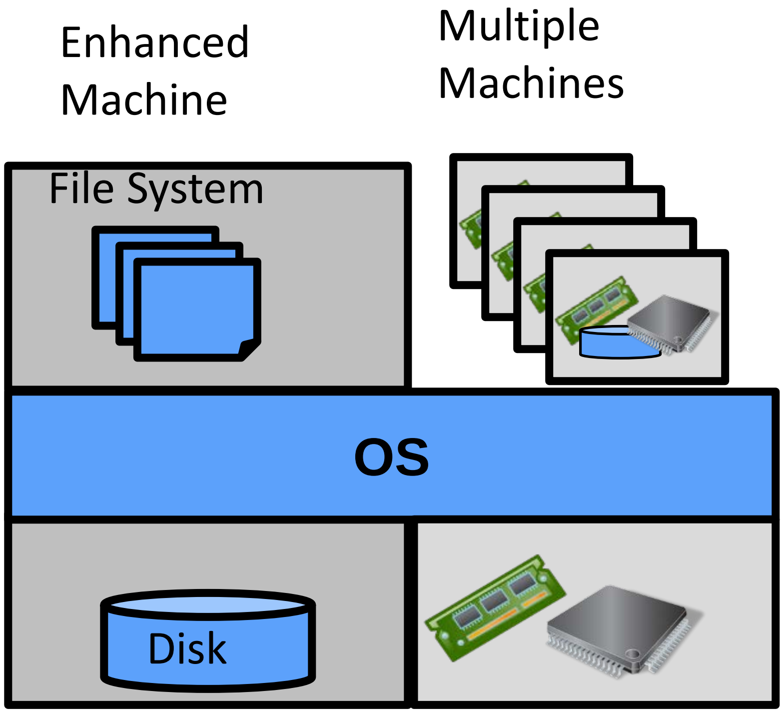
Complexity of CPS Requires Modularization

- Large CPS integrated from components from different organizations
- Prevent modification of component from affecting all system

Everything is Distributed

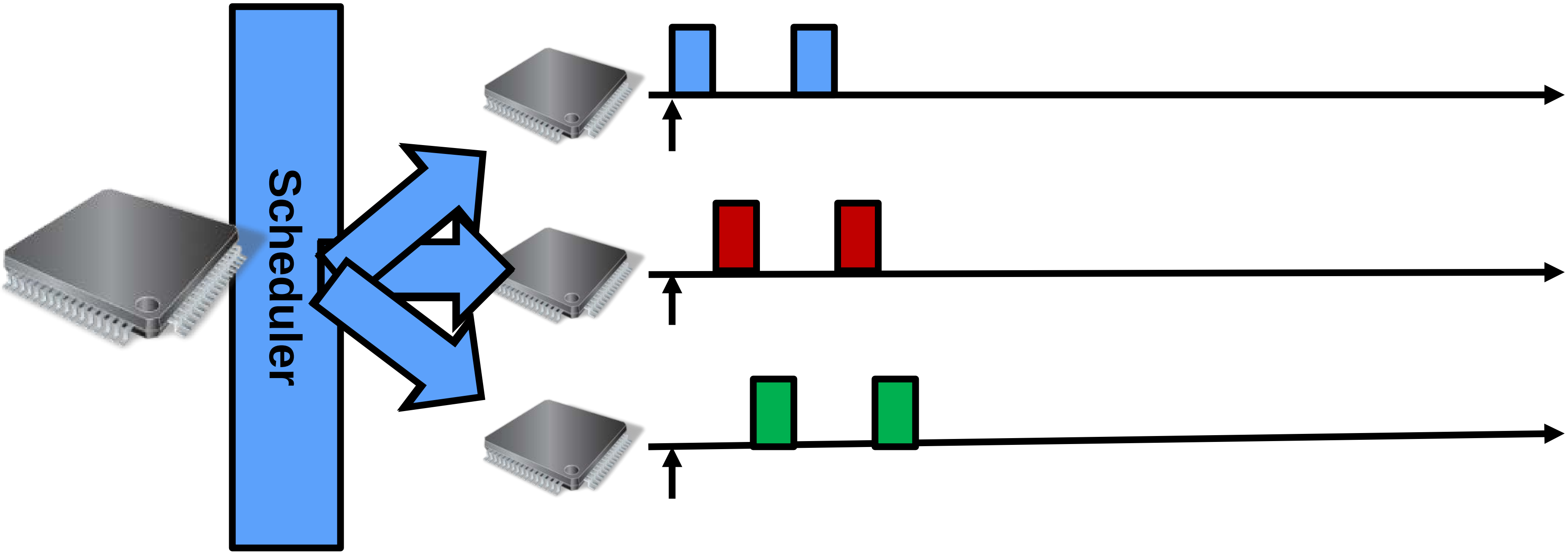
- Any medium-size CPS runs on a network of computers

OS Dual Objective



Icons credit: http://www.doublejdesign.co.uk

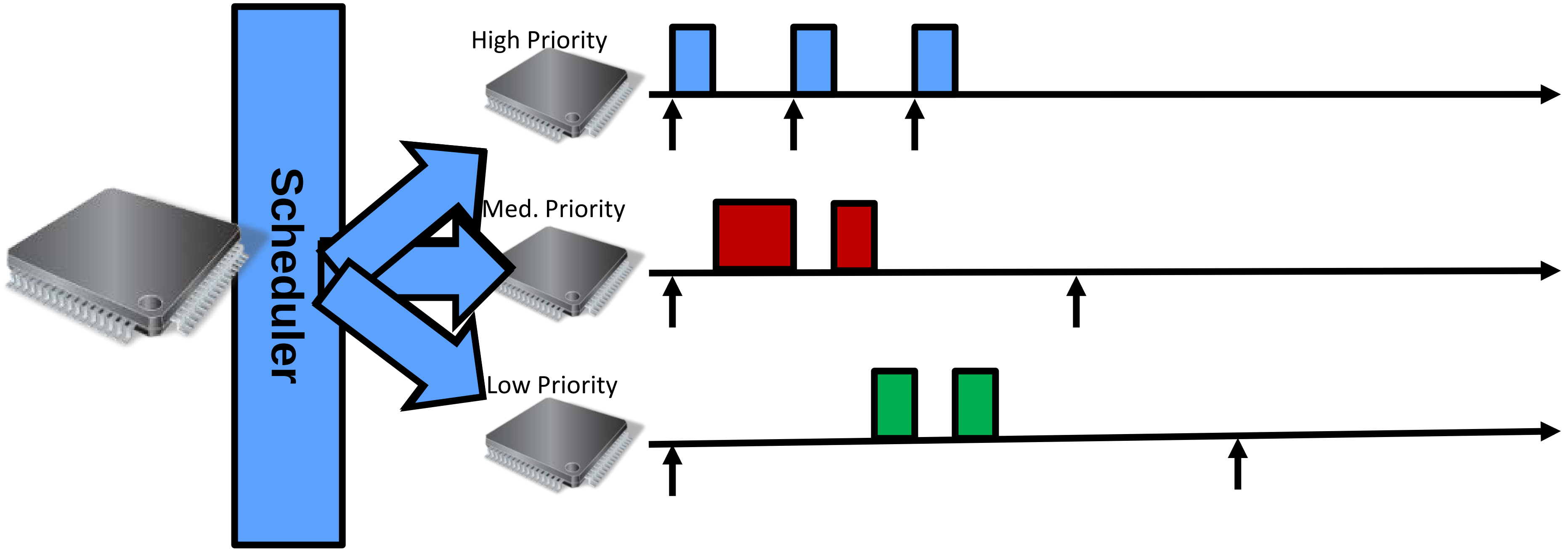
Time-Sharing CPU – Round robin



Same time requirement – Fair Scheduling

Icons credit: http://www.doublejdesign.co.uk

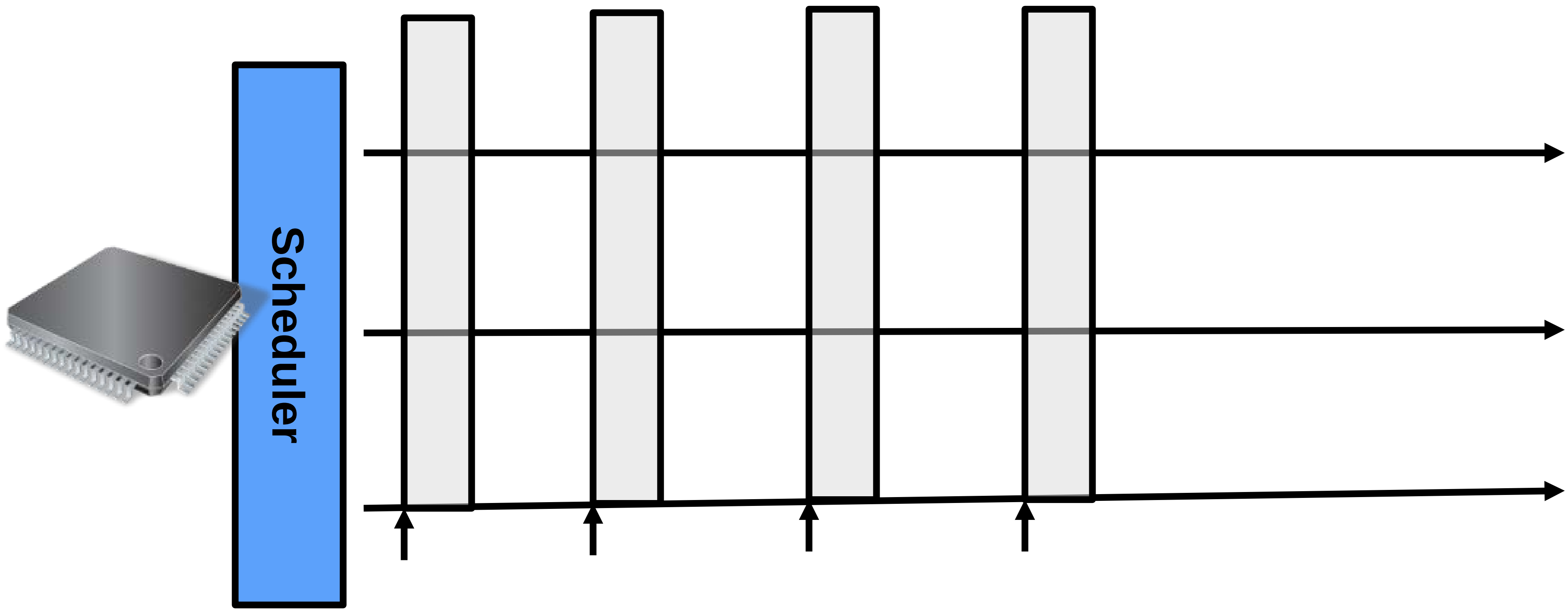
Fixed-Priority Scheduling + Rate Monotonic



Different Requirements – Guaranteed Timing

Icons credit: http://www.doublejdesign.co.uk

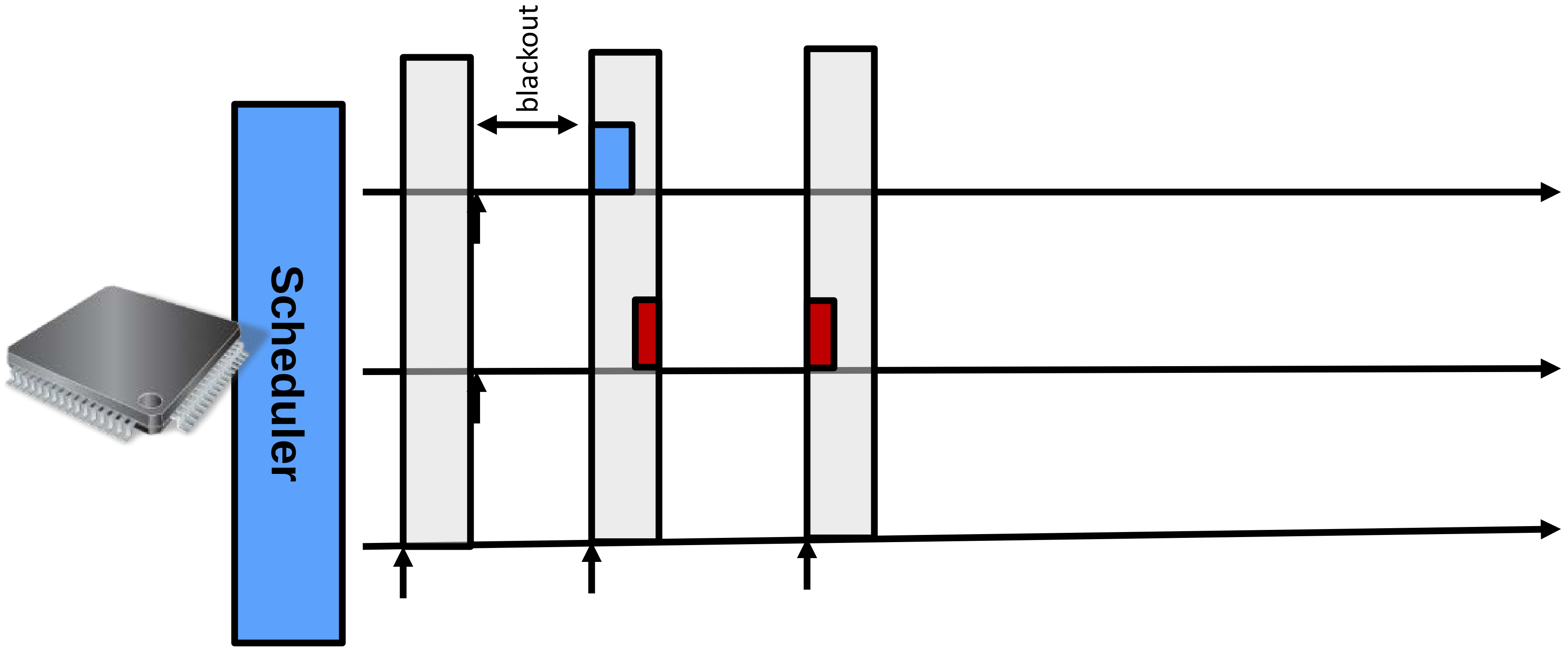
Periodic Server



“Shape” Execution

Icons credit: http://www.doublejdesign.co.uk

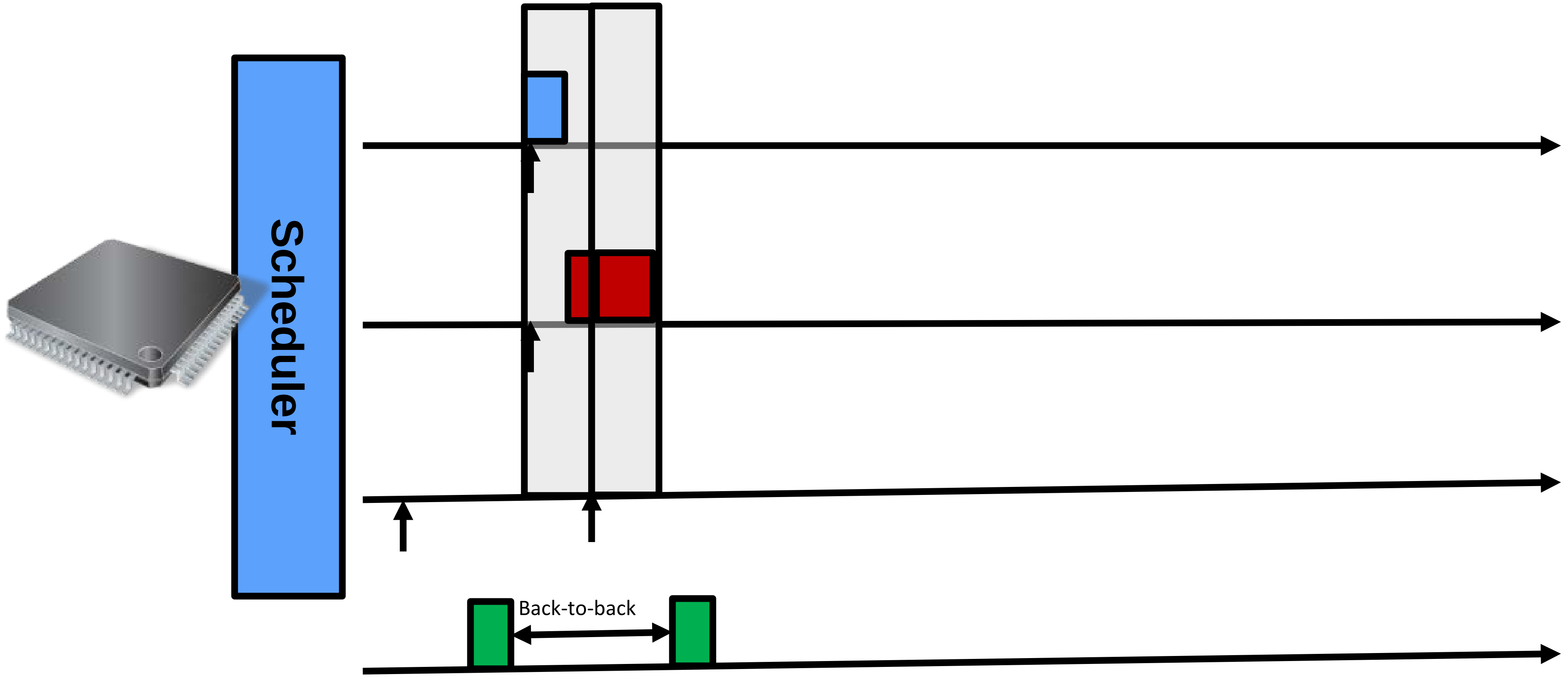
Periodic Server



Blackout Interval

Icons credit: http://www.doublejdesign.co.uk

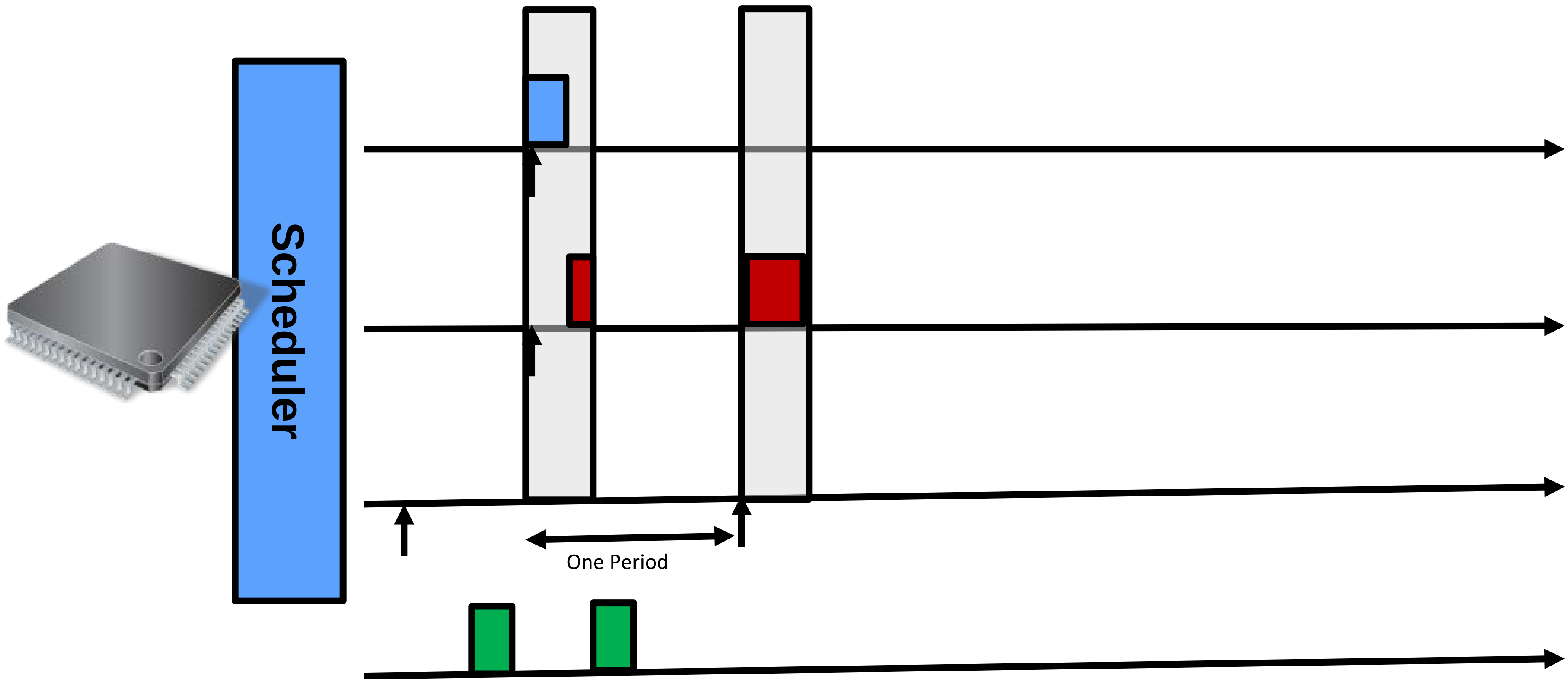
Deferrable Server



Back-to-Back Preemption

Icons credit: http://www.doublejdesign.co.uk

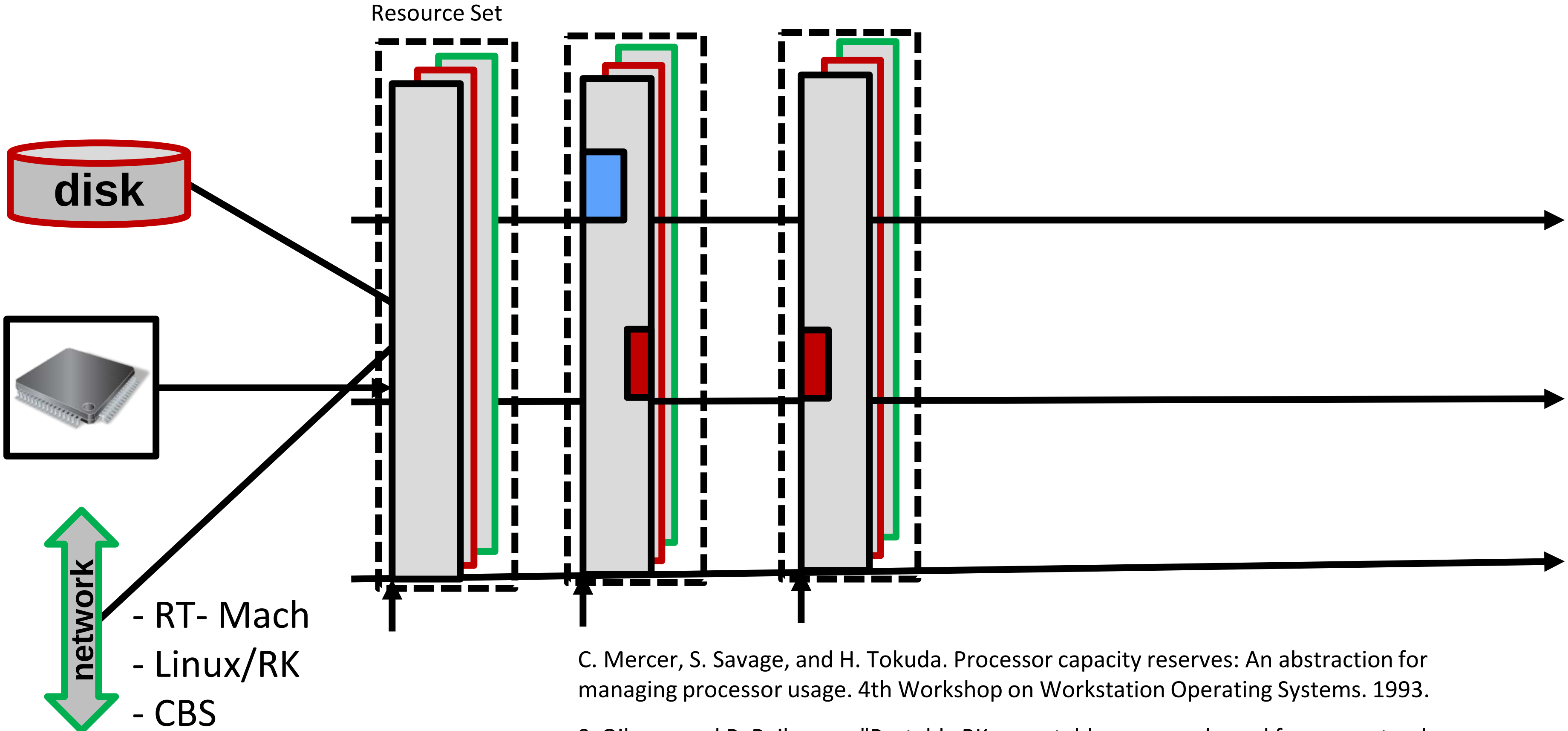
Sporadic Server



No back-to-back but multiple replenishment pieces

Icons credit: http://www.doublejdesign.co.uk

Resource Reserves OS Abstraction



- RT- Mach
- Linux/RK
- CBS
- ZSRM
- Time-Triggered (ARINC 653)

C. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: An abstraction for managing processor usage. 4th Workshop on Workstation Operating Systems. 1993.

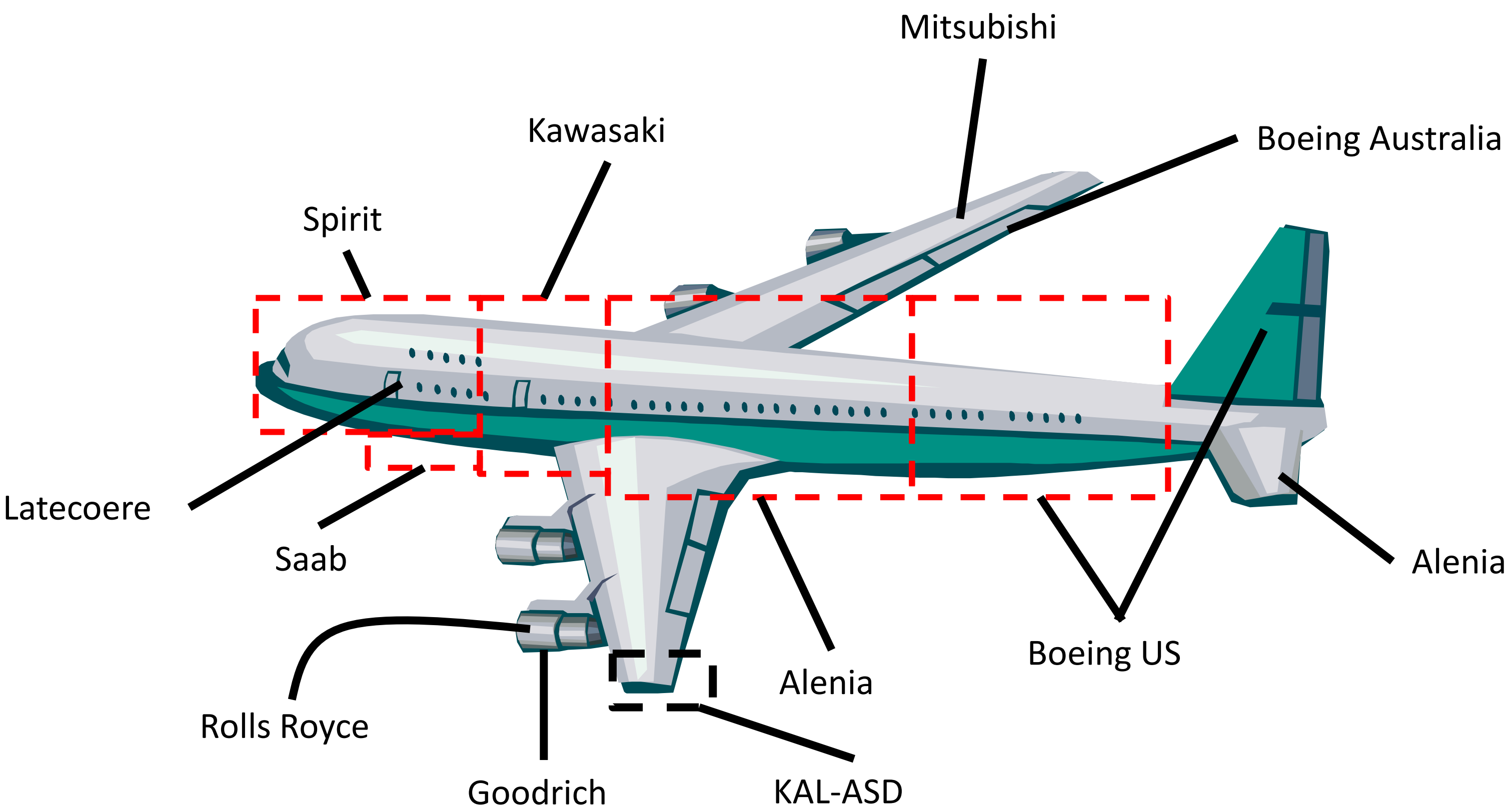
S. Oikawa and R. Rajkumar, "Portable RK: a portable resource kernel for guaranteed and enforced timing behavior," *RTAS, 1999.*

L. Abeni and G. Buttazzo.Resource Reservation in Dynamic Real-Time Systems. Journal of *Real-Time Systems.* 27, 2 (July 2004)

D. de Niz, K. Lakshmanan, and . Rajkumar. On the Scheduling of Mixed-Criticality Real-Time Tasksets. RTSS 09.

Icons credit: http://www.doublejdesign.co.uk

What Type of Temporal Protection?
Boeing 787 Suppliers



FAA Needs Modular Certification

Source: Boeing / Reuters

Mixed-Criticality

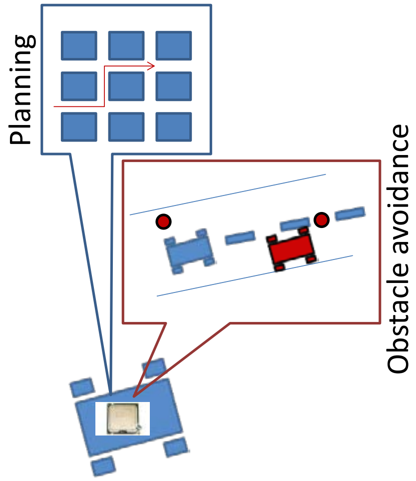
FAA / CO-178C

- Criticality ≡ Failure Condition Categories
  - Catastrophic
  - Hazardous
  - Major
  - Minor
  - No Safety Effects
- Higher The Failure Consequence Higher Level of Correctness Certainty
- Low-criticality allowed lower correctness certainty iff prevented from interfering with high-criticality

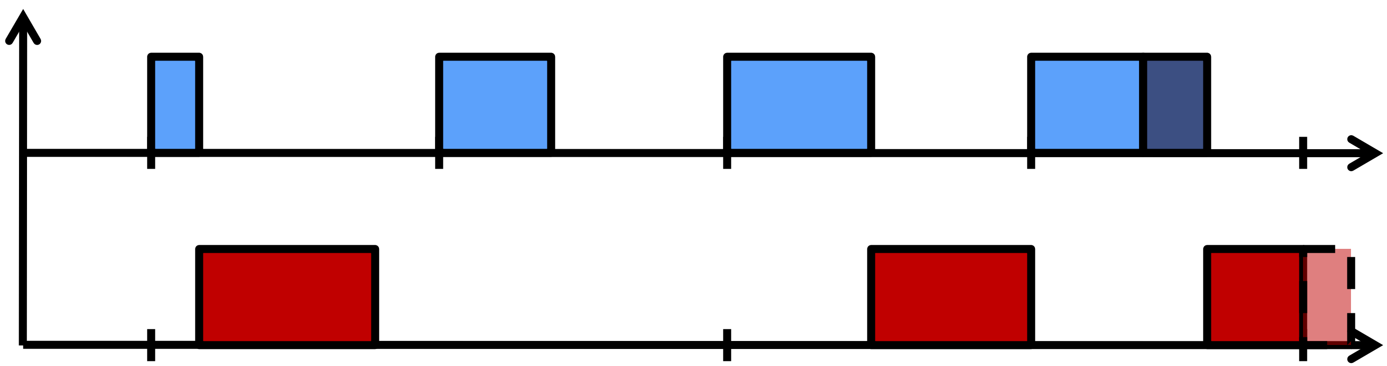
Scheduling (S. Vestal)

- Higher Estimate of Worst-Case Execution Time => Higher Confidence of Correctness
- Protect Higher-Criticality Tasks from Lower-Criticality Ones
  - ASYMMETRIC PROTECTION!

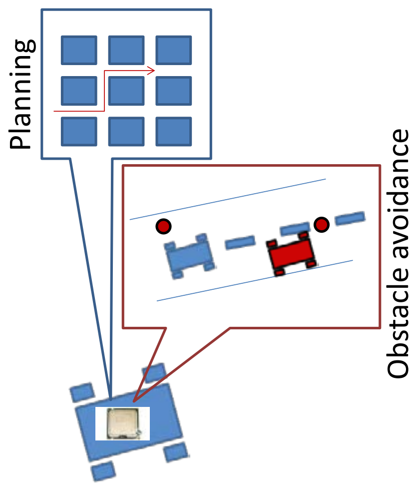
Consolidation of Mixed-Criticality Tasks



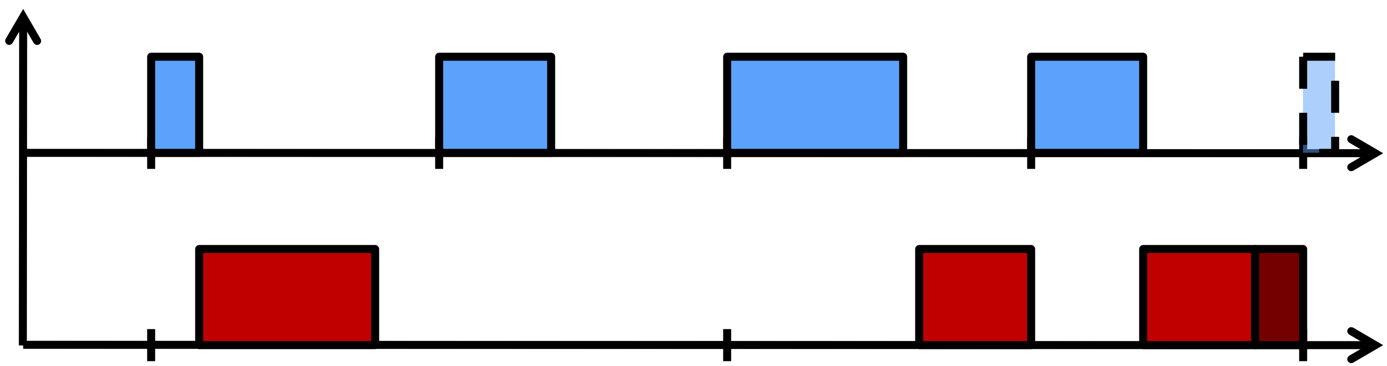
Shared Hardware
Can lead to cycle stealing



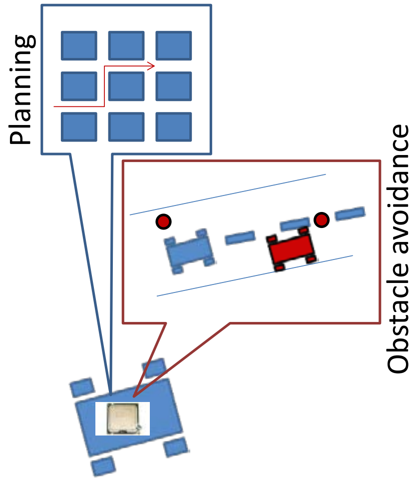
Consolidation of Mixed-Criticality Tasks



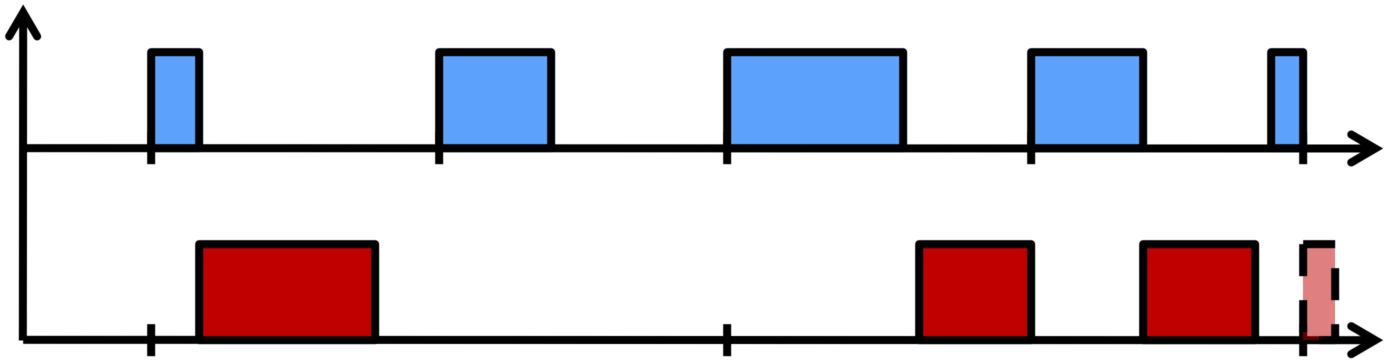
To avoid interference
add temporal protection



Consolidation of Mixed-Criticality Tasks



BUT
Symmetric protection
leads to *criticality inversion*



Rate-Monotonic Priority

Shorter Period → Higher Priority

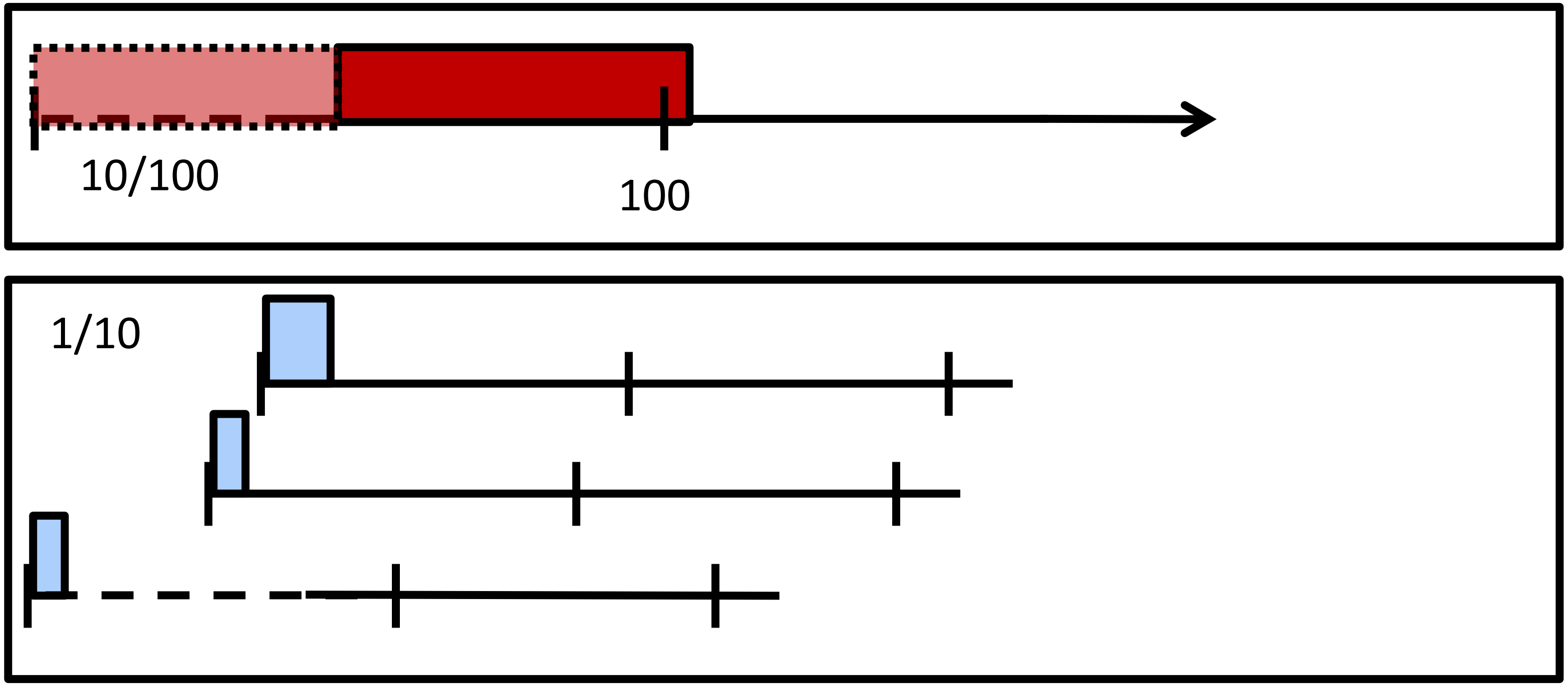
- Ideal utilization

BUT: Poor Criticality Protection Due to Criticality Inversion

- If criticality order is opposite to rate-monotonic priority order

High Criticality

Low Criticality



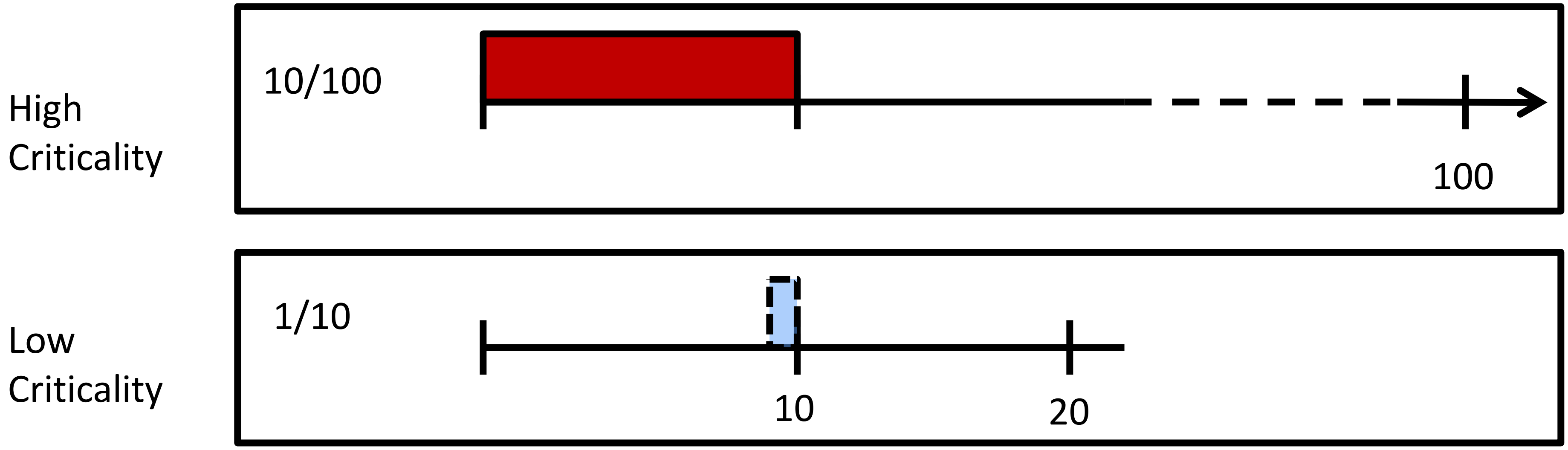
Criticality As Priority Assignment (CAPA)

Higher Criticality → Higher Priority

- Ideal criticality protection:
  - lower criticality cannot interfere with higher criticality

BUT: Poor Utilization Due to Priority Inversion

- If criticality order is opposite to rate-monotonic priority order



Zero-Slack Scheduling

Tasks with:

- Period (T), Normal (C), and Overloaded (C^o) Execution Times

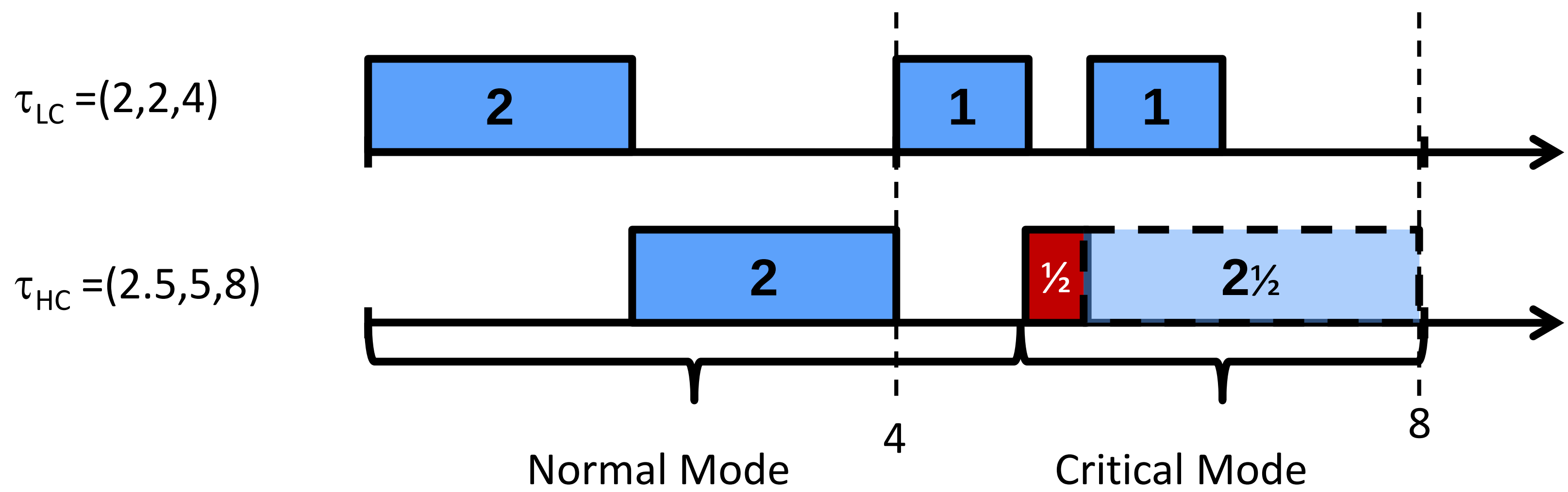
Start with rate-monotonic scheduling

Calculate the last instant before τ_{HC} misses its deadline

- this is called the ***zero-slack*** instant

Switch to criticality-as-priority

- Splits the execution window into
  - Normal mode (RM)
  - Critical mode (CAPA)



D. de Niz, K. Lakshmanan, and R. Rajkumar. On the Scheduling of Mixed-Criticality Real-Time Tasksets. RTSS 09.

Zero-Slack Scheduling

Tasks with:

- Period (T), Normal (C), and Overloaded (C^o) Execution Times

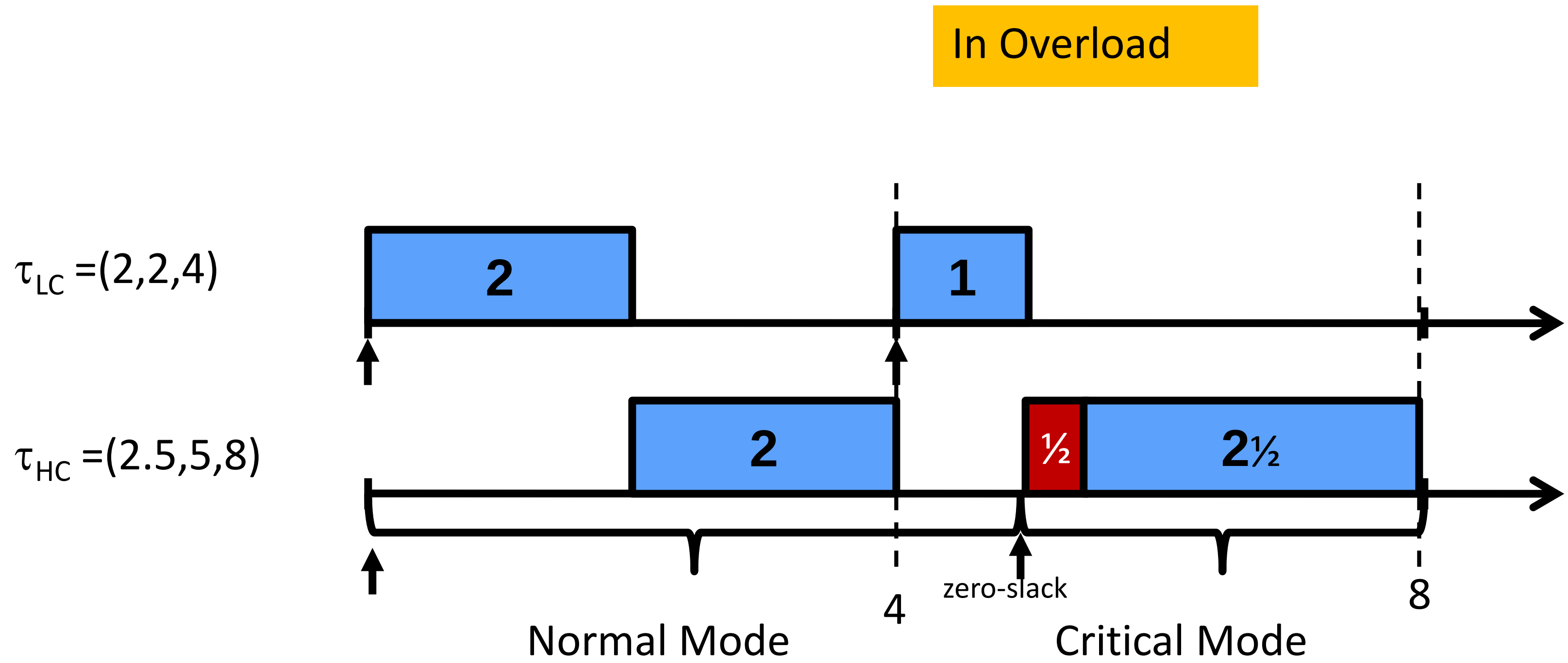
Start with rate-monotonic scheduling

Calculate the last instant before τ_{HC} misses its deadline

- this is called the ***zero-slack*** instant

Switch to criticality-as-priority

- Splits the execution window into
  - Normal mode (RM)
  - Critical mode (CAPA)



D. de Niz, K. Lakshmanan, and R. Rajkumar. On the Scheduling of Mixed-Criticality Real-Time Tasksets. RTSS 09.

Zero-Slack Scheduling

Tasks with:

- Period (T), Normal (C), and Overloaded (C^o) Execution Times

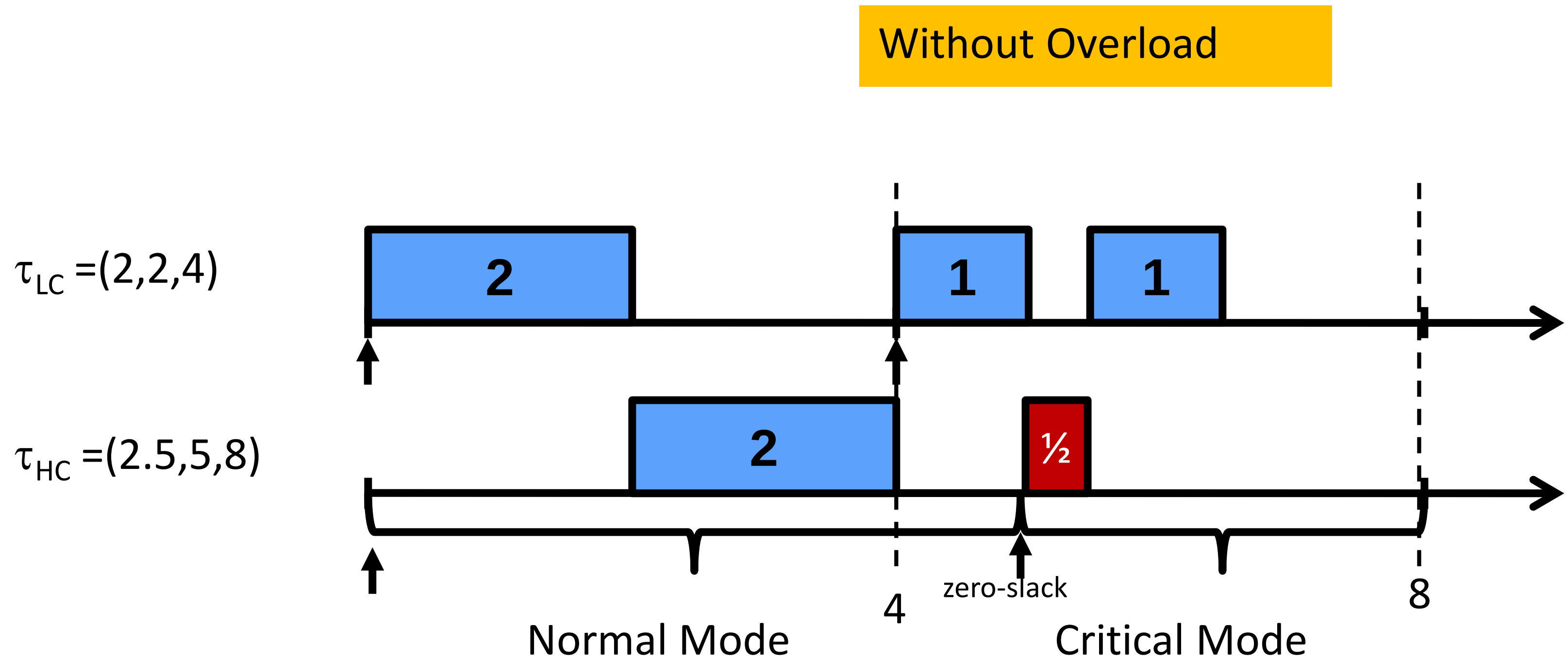
Start with rate-monotonic scheduling

Calculate the last instant before τ_{HC} misses its deadline

- this is called the ***zero-slack*** instant

Switch to criticality-as-priority

- Splits the execution window into
  - Normal mode (RM)
  - Critical mode (CAPA)



D. de Niz, K. Lakshmanan, and R. Rajkumar. On the Scheduling of Mixed-Criticality Real-Time Tasksets. RTSS 09.

ZSRM Properties

Subsumes RM

- If criticalities are aligned to priorities
- No critical mode

Subsumes CAPA

- If not enough slack, only critical mode

Graceful Degradation

- In overloads, deadlines are missed in reverse criticality order

Implementation

ZSRM

Scheduling algorithm calculates zero-slack instants offline

Linux/ RK

- Resource reservation in Linux
  - CPU, Net, Mem, Disk
- Bundled into resource sets that provide a form of virtual machine
- Multiple implementations
  - Nano/RK for sensor networks

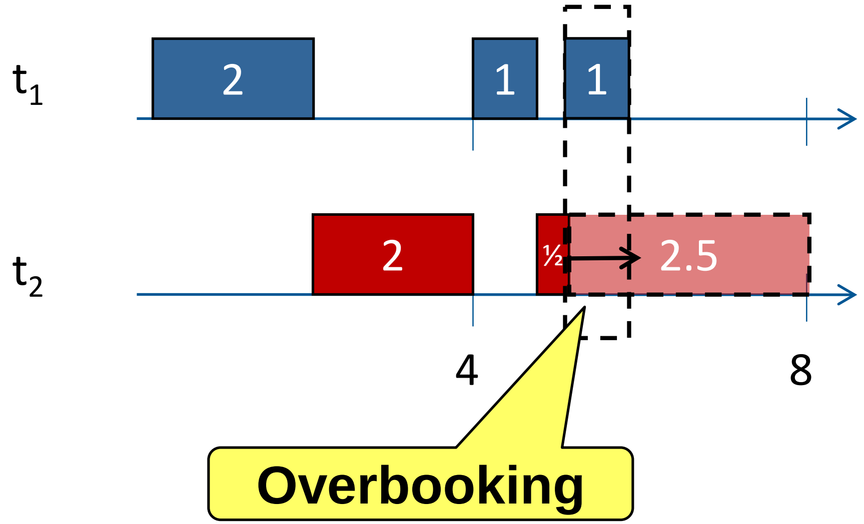
Special Zero-Slack Reserves

- Switch to critical mode
  - Stop lower-criticality tasks on zero-slack instant
- Tasks in critical mode in stack

Library prototype implementation for VxWorks

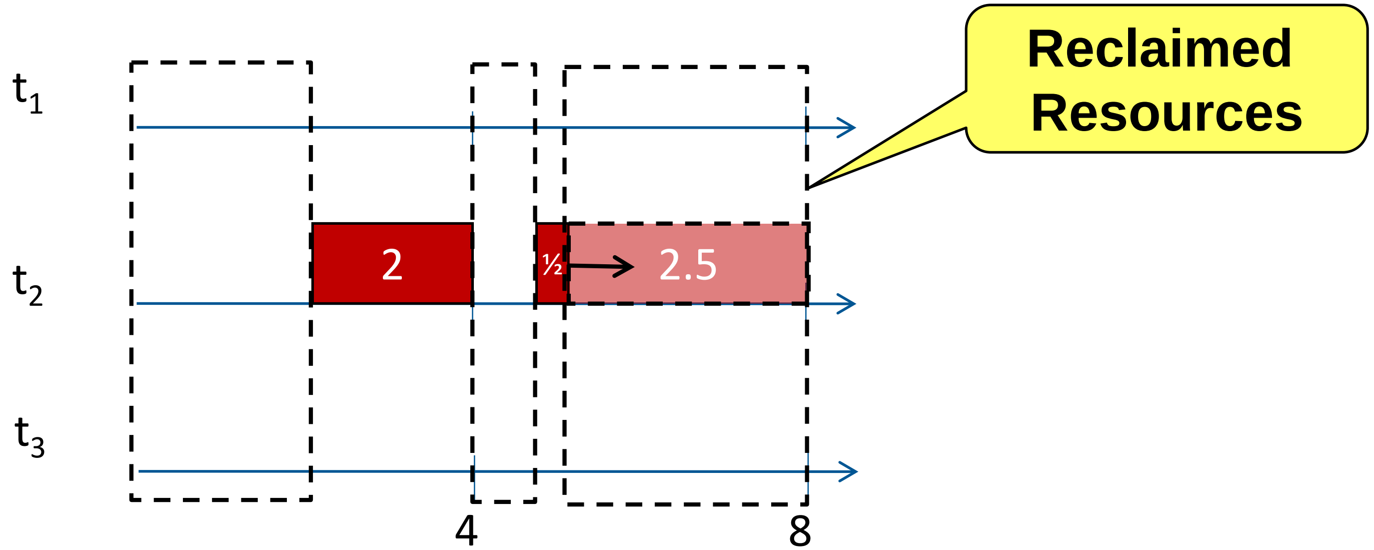
What if criticality exhibits diminishing returns?

Task	Period	Criticality	WCET	NCET
t_1 Surveillance Cov.	4	Mission	2	2
t_2 Collision Avoid.	8	Safety	5	2.5



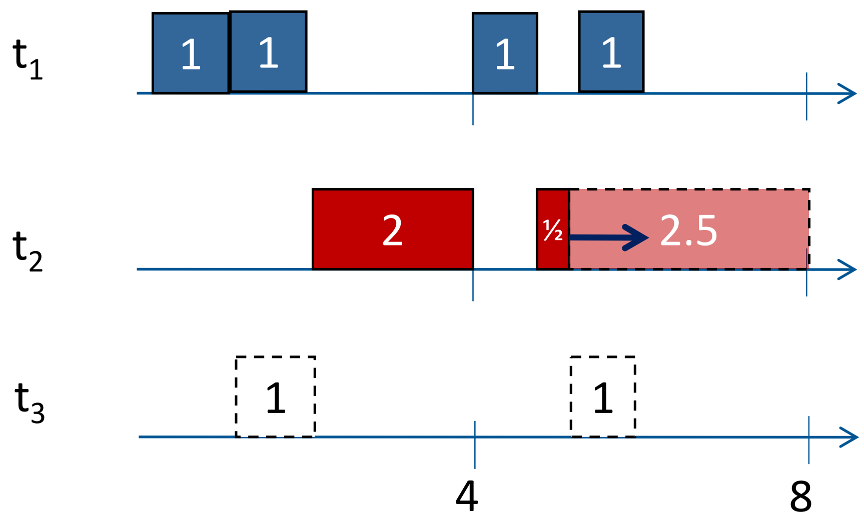
Reclaiming Resources in Mixed-Criticality Systems

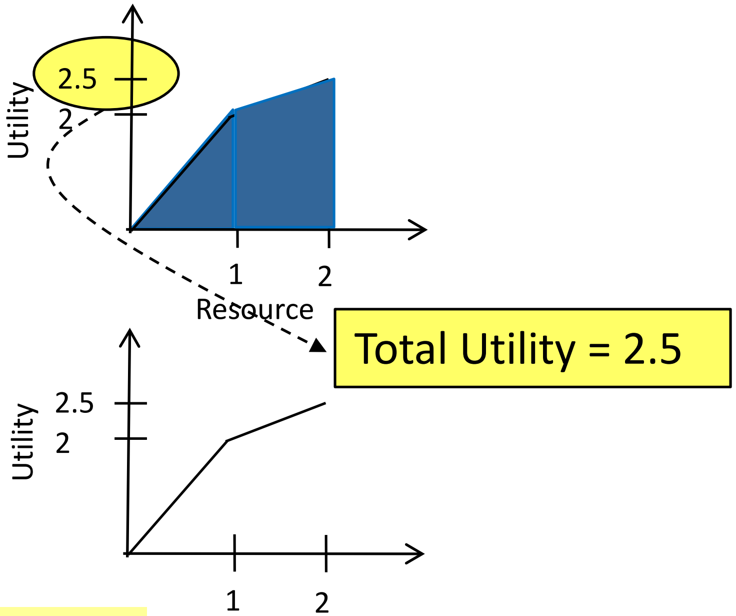
Task	Period	Criticality	WCET	NCET	Utility
t_1 Surveillance Cov.	4	Mission	2	2	{2,2.5}
t_2 Collision Avoid.	8	Safety	5	2.5	
t_3 Amount of Intelligence	4	Mission	2	2	{2,2.5}



Using Reclaimed Resources to Maximized Utility

Task	Period	Criticality	WCET	NCET	Utility Levels
t_1 Surveillance Cov.	4	Mission	2	2	{2,2.5}
t_2 Collision Avoid.	8	Safety	5	2.5	
t_3 Amount of Intelligence	4	Mission	2	2	{2,2.5}

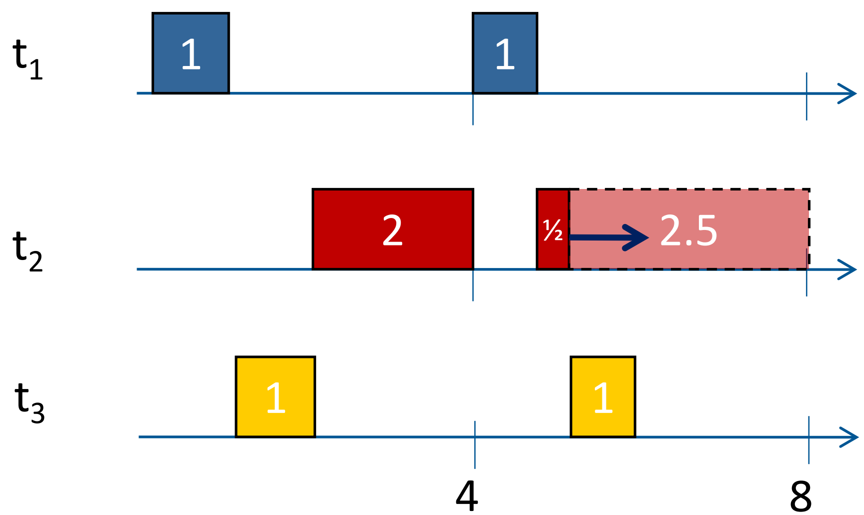


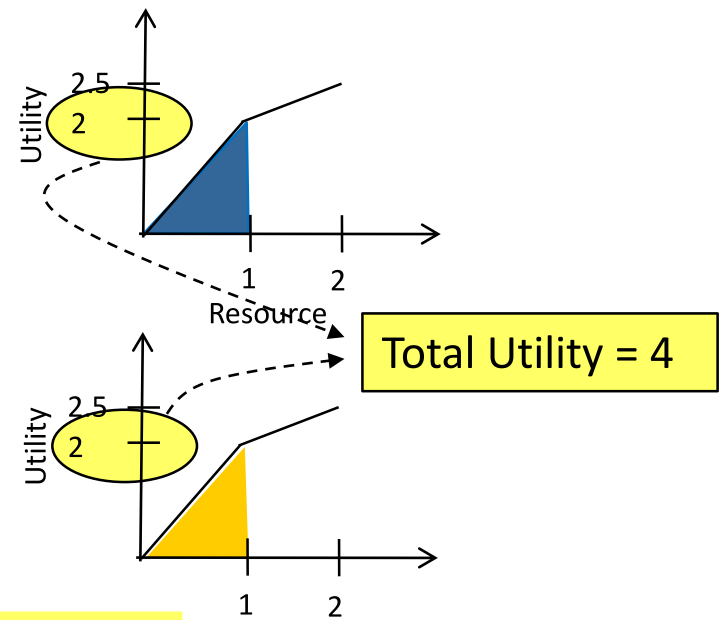


Utility Diminishes: Utility ≠ Criticality

Using Reclaimed Resources to Maximized Utility

Task	Period	Criticality	WCET	NCET	Utility Levels
t_1 Surveillance Cov.	4	Mission	2	2	{2,2.5}
t_2 Collision Avoid.	8	Safety	5	2.5	
t_3 Amount of Intelligence	4	Mission	2	2	{2,2.5}

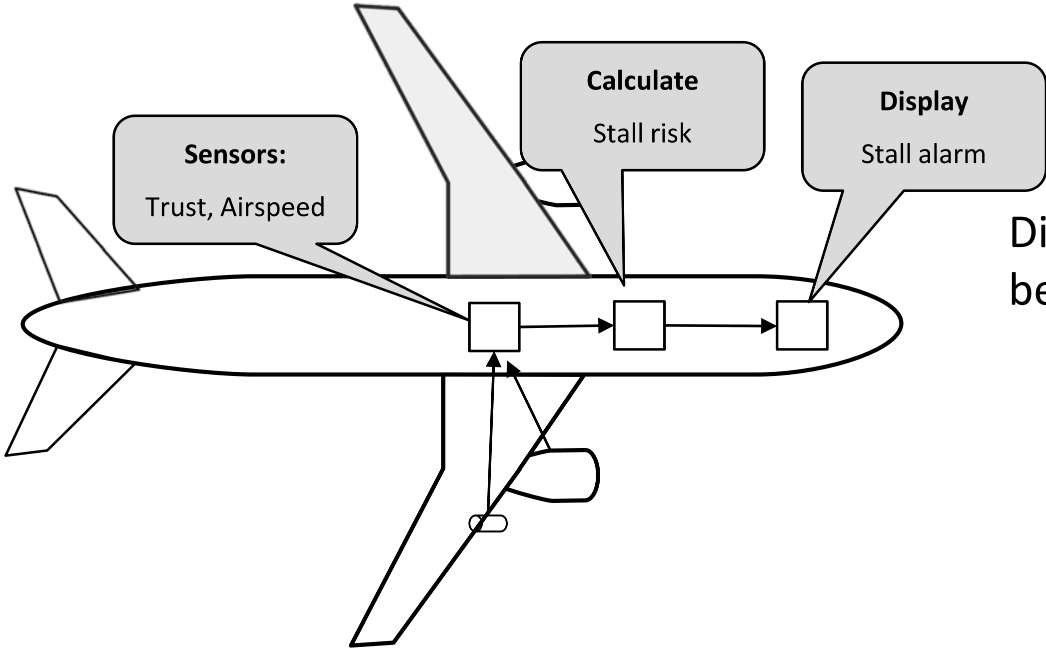




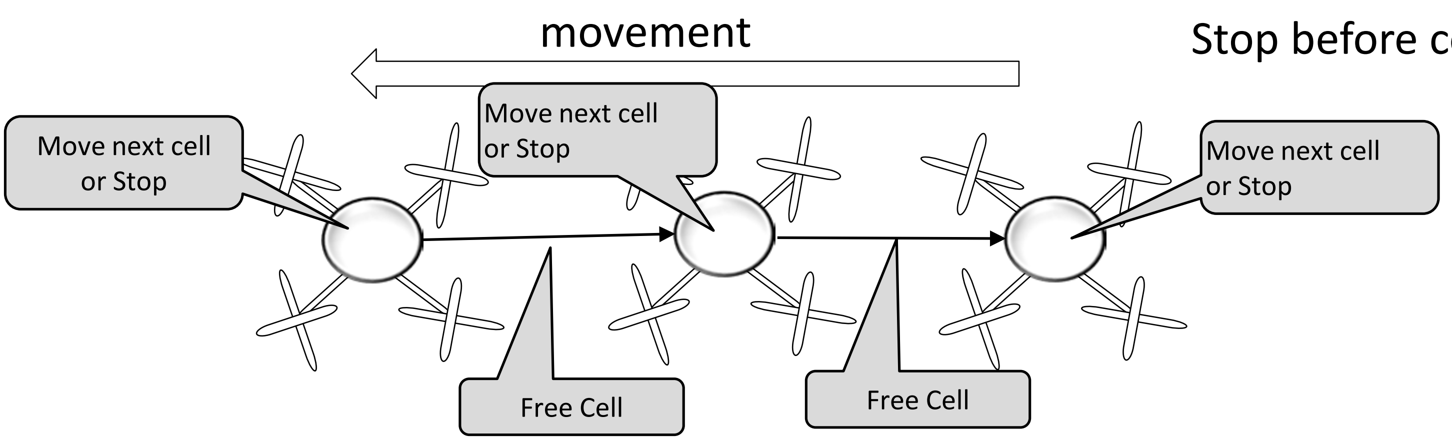
ZS-QRAM: More mission-critical utility from same resources

D. de Niz, L. Wrage, N. Storer, A. Rowe, and R. Rajkumar.
On Resource Overbooking in an Unmanned Aerial Vehicle. ICCPS. 2012.

End-to-End Timing Requirements

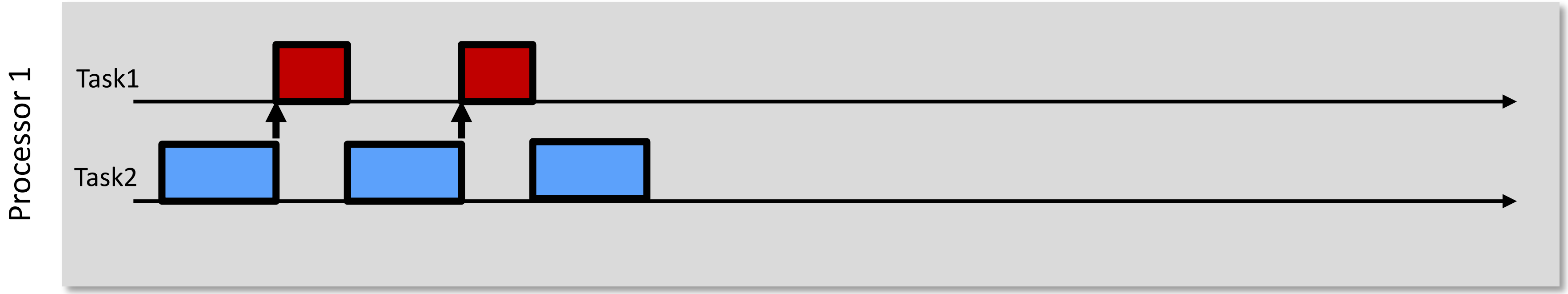


Display stall alarm
before too late to correct

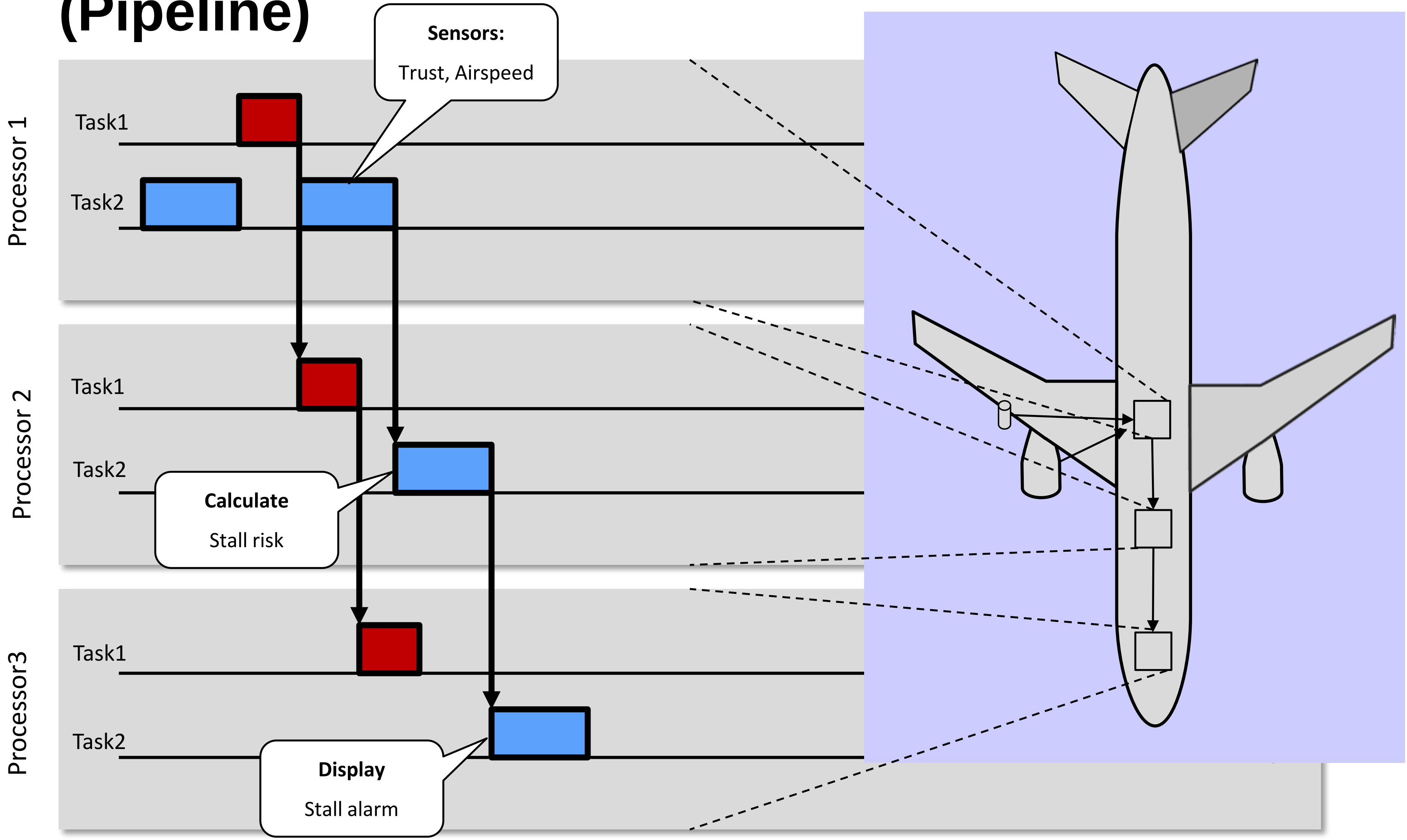


Move continuously or
Stop before collision

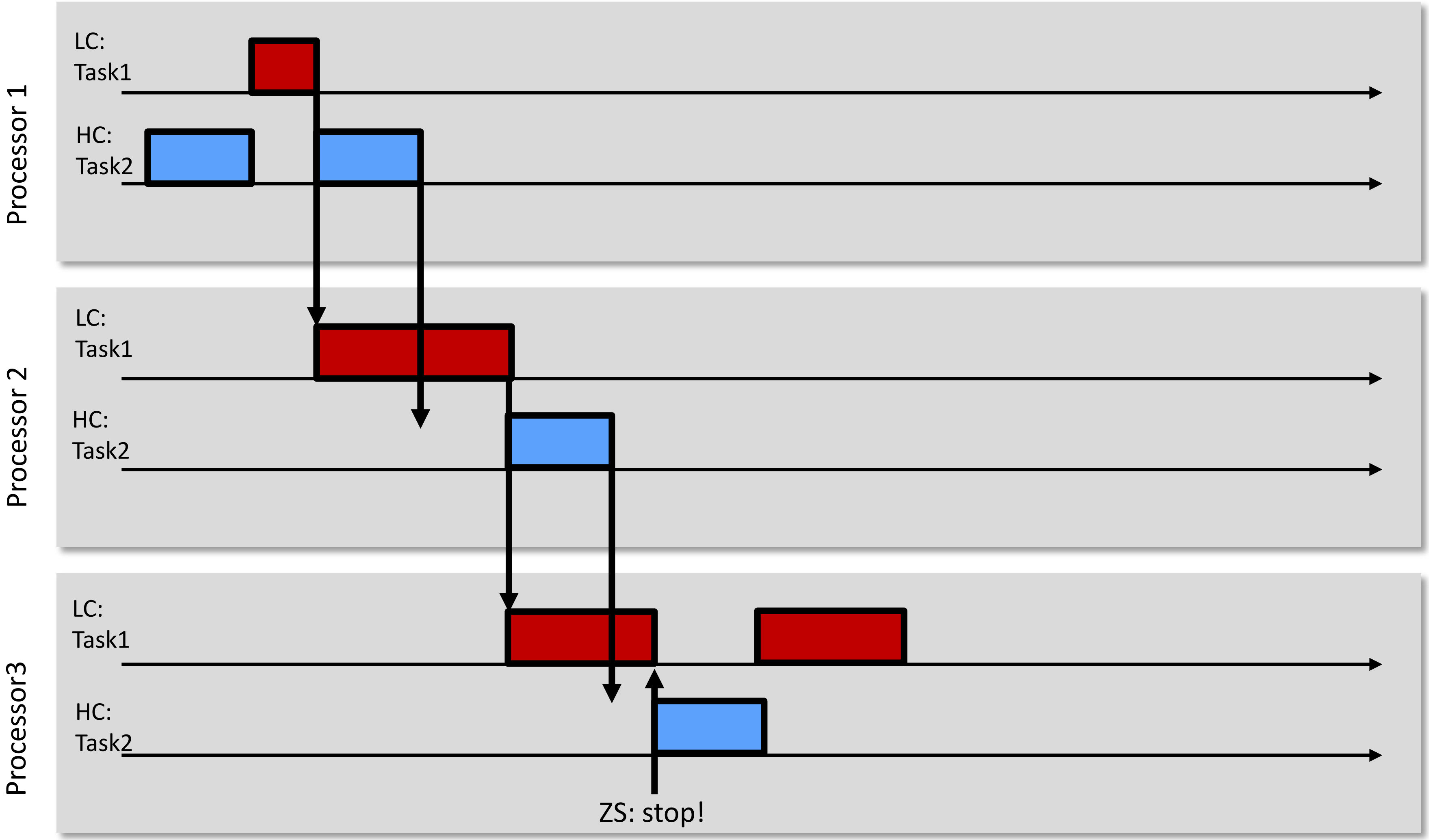
From Single Processor Preemptions



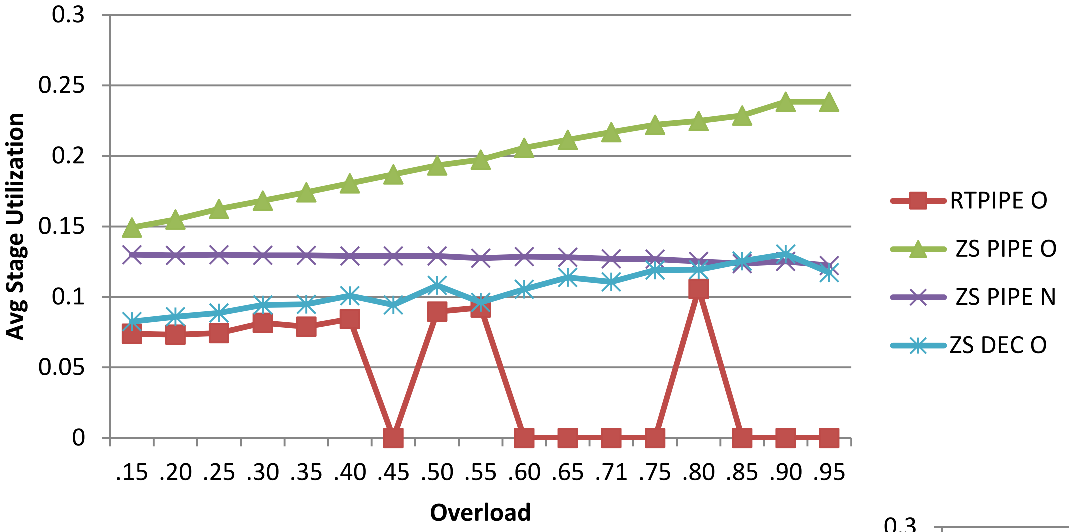
To Preemptions in Different Processors (Pipeline)

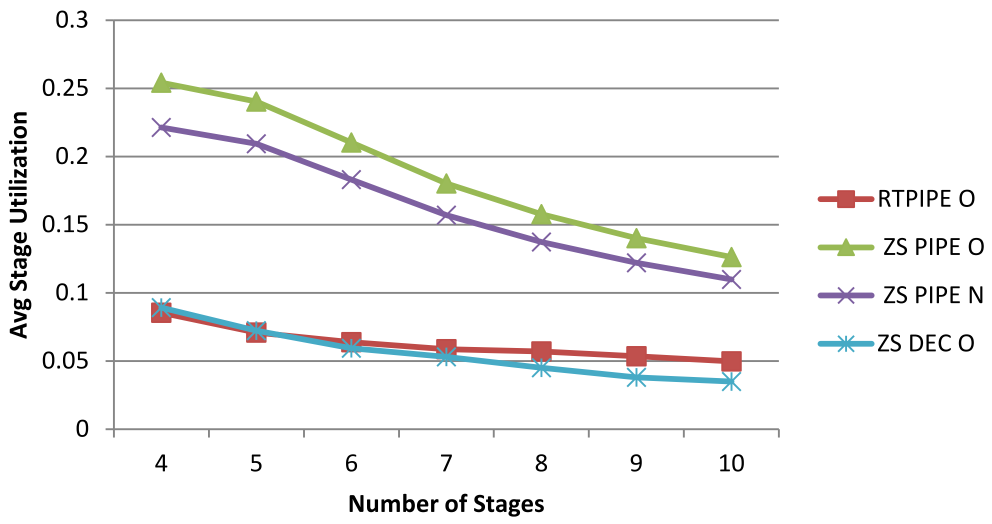


ZSRM Pipeline



ZSRM Pipeline Performance





Future Challenges

Increasingly Complex Hardware

- Massive multicores: Tile processors
- Heterogeneous multicores: hardware accelerators
- Adaptive Power: Dynamic Thermal Management

Increasingly Complex Software

- Machine Learning
  - Learning while executing could lead to unpredictable WCET
  - How to bound unpredictable behavior On Time (before crash)
- Complex distributed systems
  - Coordinated autonomy: smart highways
  - Global Internet of Things

Increasing Development Complexity

- Components from Different Suppliers
- Potential Safety Enforcement from Safety/Certification Authorities