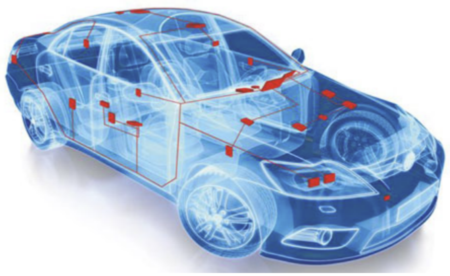
Physical-State-Aware Dynamic Slack Management for Mixed-Criticality Systems

Hoon Sung Chwa and Kang G. Shin
University of Michigan

Hyeongboo Baek and Jinkyu Lee
Sungkyunkwan University (SKKU)

Mixed-Criticality Systems

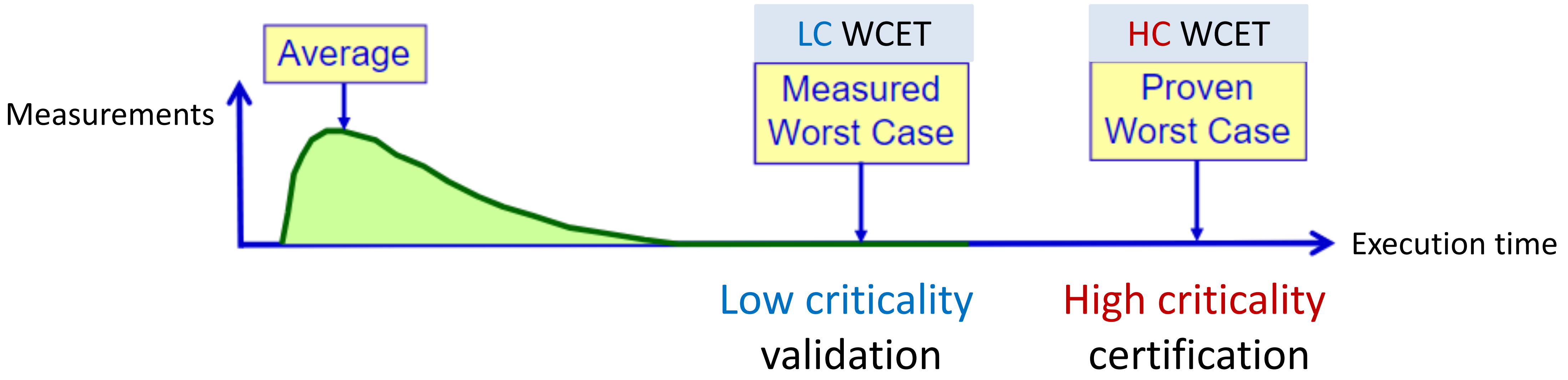
- Apps/systems with **different criticality levels**
  - Automotive systems
    - Automotive safety integrity level (ASIL) in ISO 26262 standard
    - Different levels of safety assurance



A	B	C	D
Navigation	Lane departure		Acceleration control (ACC)
Entertainment	Speedometer		Steering control (AVS)
Lighting	Rear camera		Braking control (ABS)

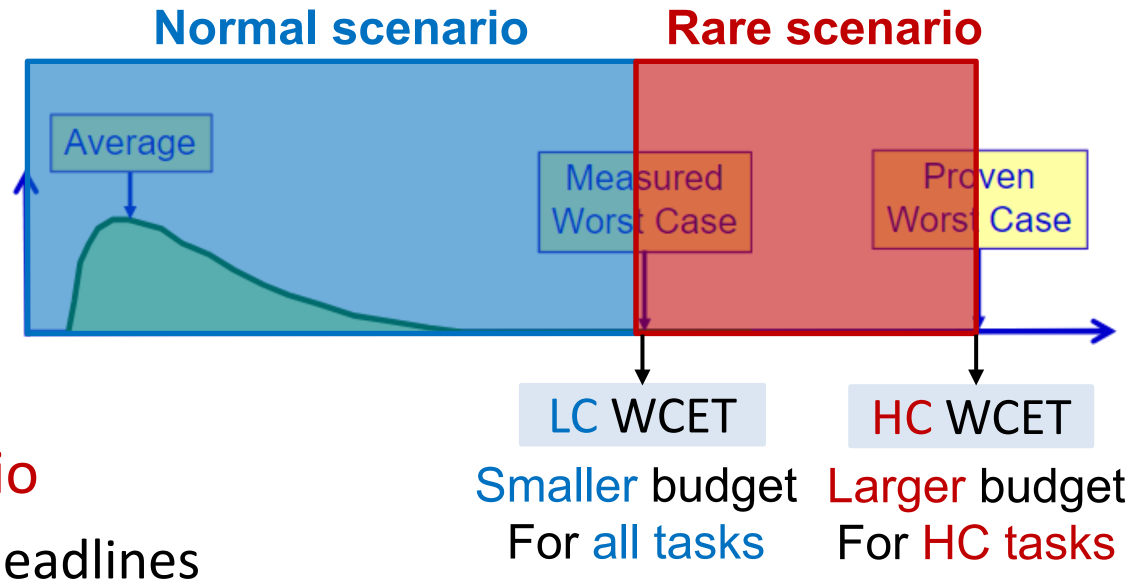
Mixed-Criticality Systems

- **Multiple** worst-case execution time (**WCET**) **estimates**
  - Different levels of confidence
    - Low criticality validation: extensive experimentation under normal scenarios
    - High criticality certification: cycle-counting/flow-analysis under pessimistic assumptions



Mixed-Criticality (MC) Task Model

- 2 different criticality levels
  - low-criticality (LC) and high-criticality (HC)
- Multiple execution budgets
  - Smaller budget for normal scenario
    - All tasks are required to meet deadlines
  - Larger (conservative) budget for rare scenario
    - High-critical tasks are still required to meet deadlines



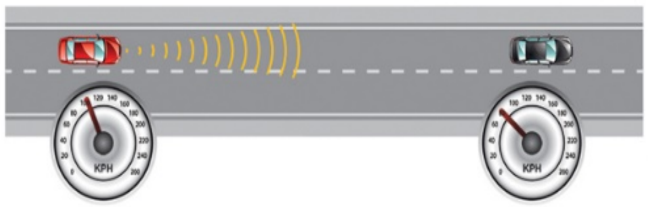
Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In RTSS, 2007.

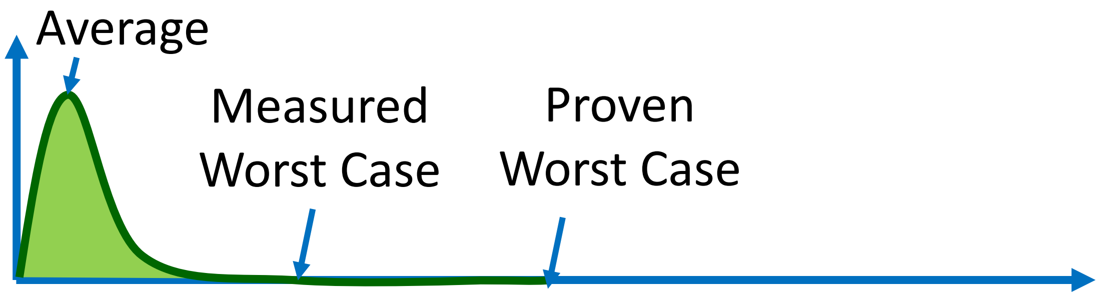
Motivation

- Assumption
  - WCET estimates do *not* change during runtime
    - *statically* derived independently of physical states

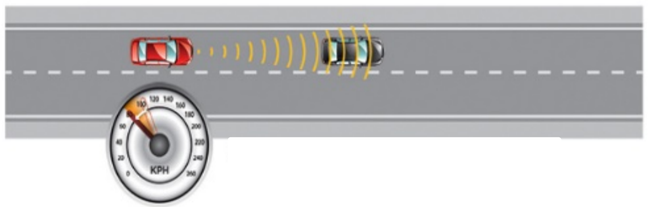
- In practice,

State 1: No vehicle in front





State 2: Approaching vehicles

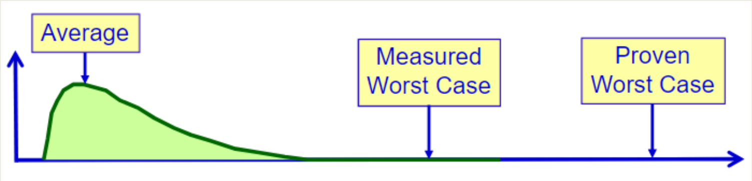


This Paper

Different criticality levels

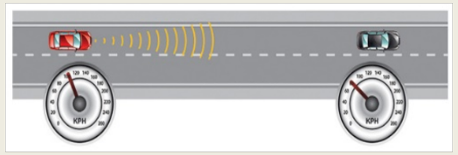
A	Low
B	Medium
C	High
D	Critical

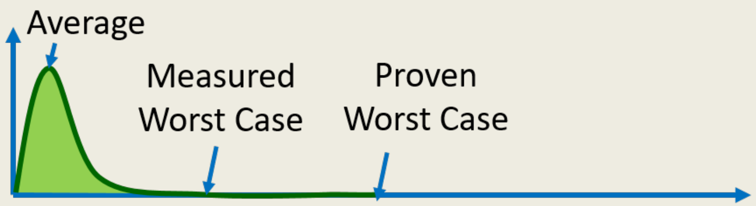
Multiple WCET estimates



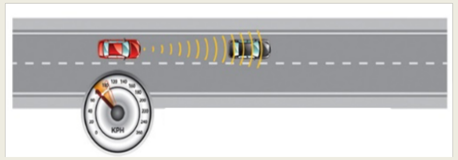
Dynamic execution behavior under varying physical states

State 1: No vehicle in front





State 2: Approaching vehicles

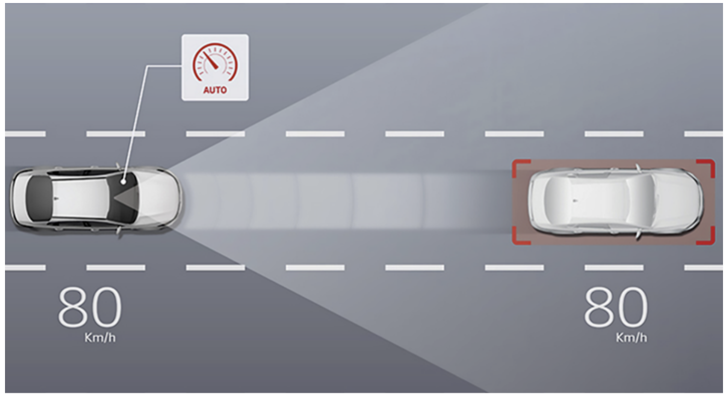


Organization of this talk

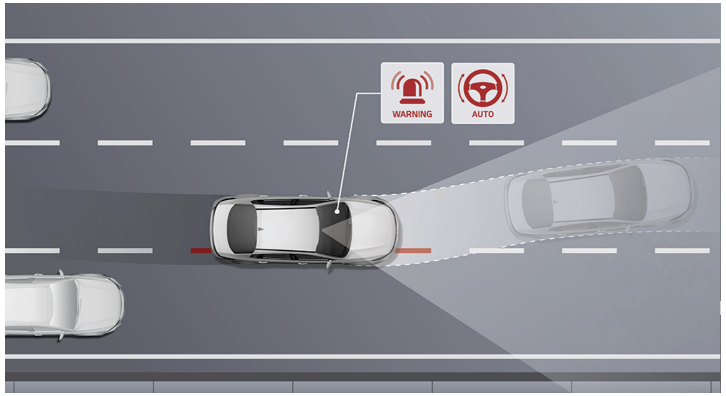
- Introduction
- Case study
  - Adaptive cruise control (ACC) & active vehicle steering (AVS)
  - Other applications
- Our approach
  - New MC task model
  - New slack concept
  - Dynamic slack management framework
- Evaluation

Case Study: ADAS system

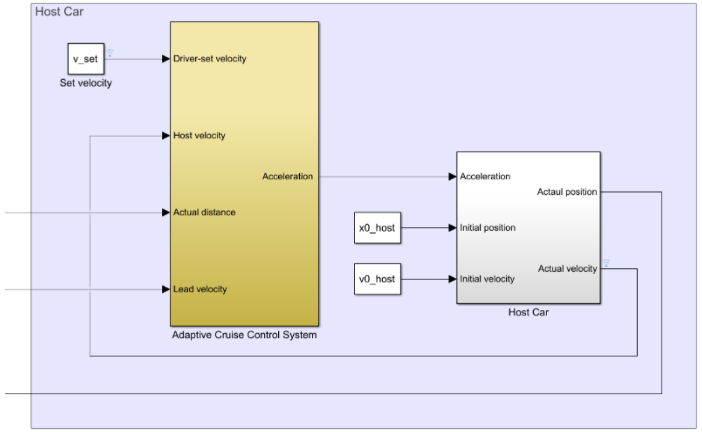
- Adaptive cruise control (ACC)
  - Speed control to maintain a safe distance



- Active vehicle steering (AVS)
  - Steering maneuver to avoid collision

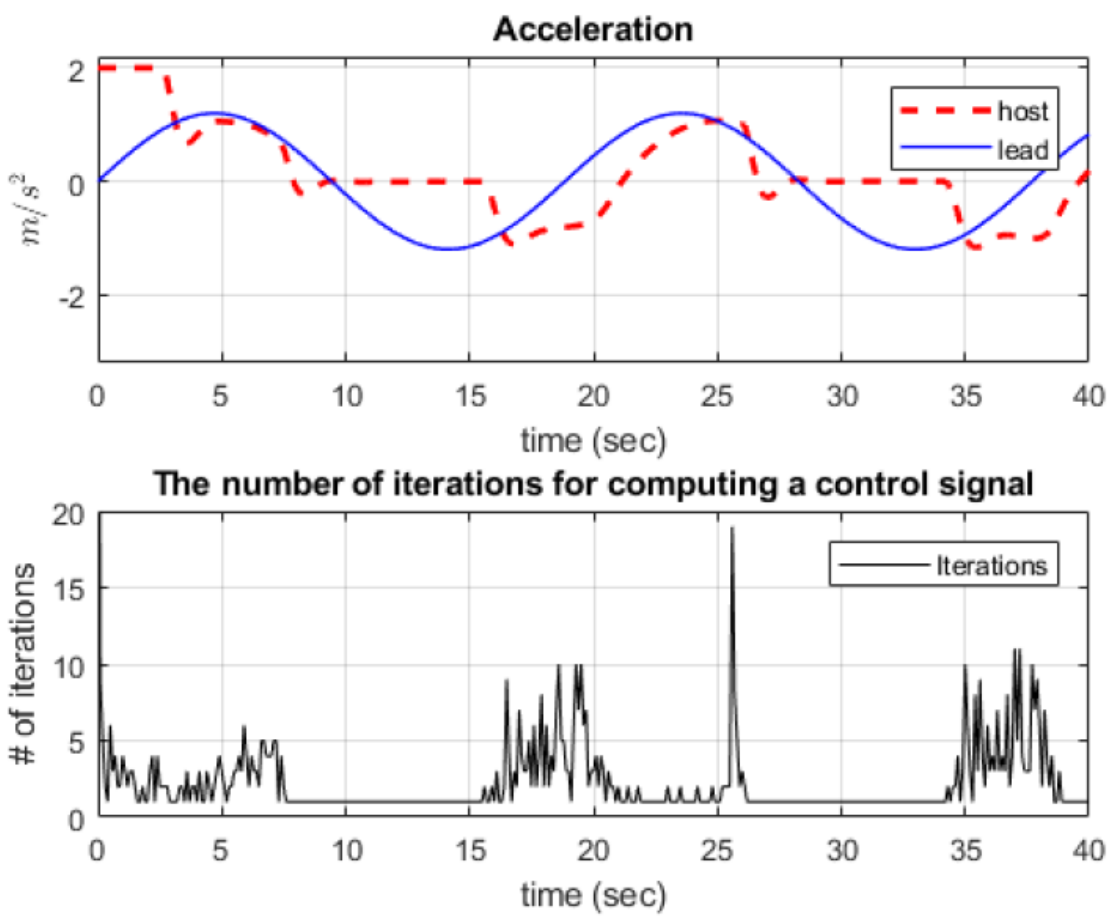


- Using *model predictive control* (MPC) in Matlab
  - Desired speed: 30m/sec
  - Sampling period: 0.1sec
  - Double lane change maneuver

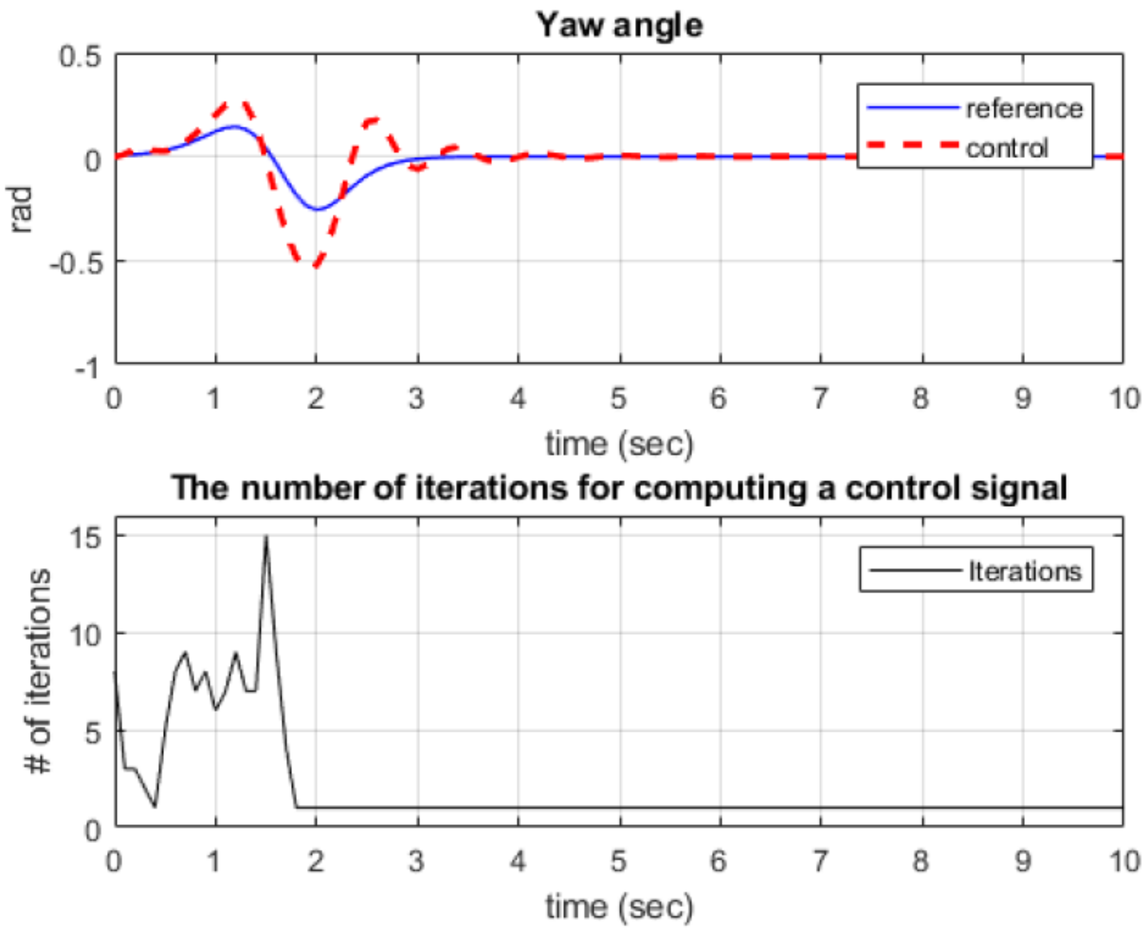


[Model predictive control toolbox, Matlab]

Motivational Simulation Results

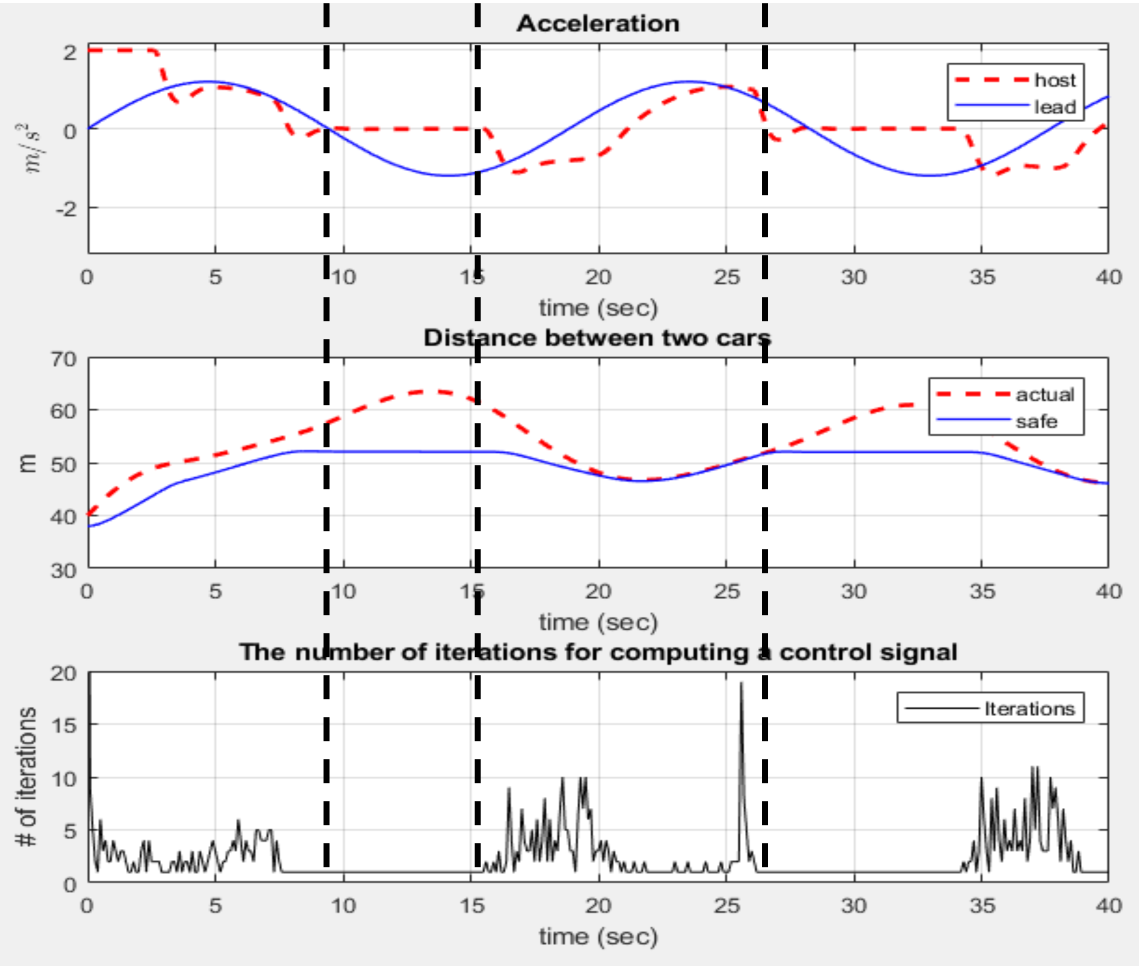


Adaptive Cruise Control (ACC)



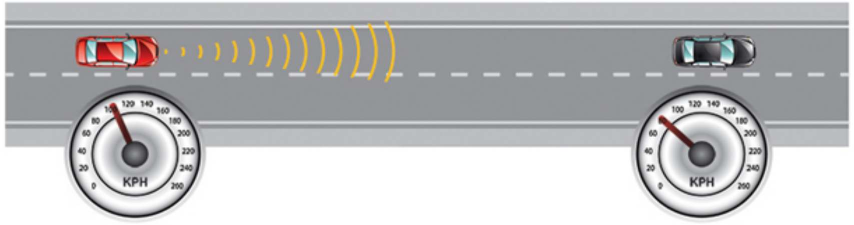
Active Vehicle Steering (AVS)

Motivational Simulation Results

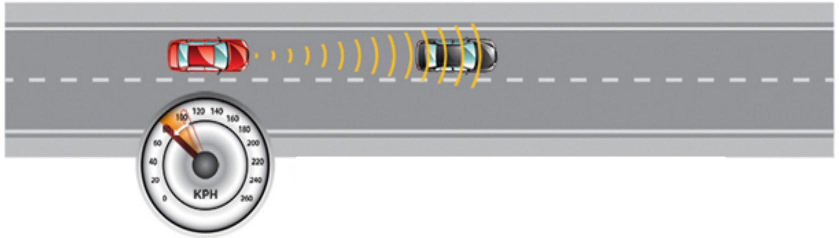


Adaptive Cruise Control (ACC)

- Execution time is strongly correlated with a physical state
  - Less exec. time (9—15 secs)

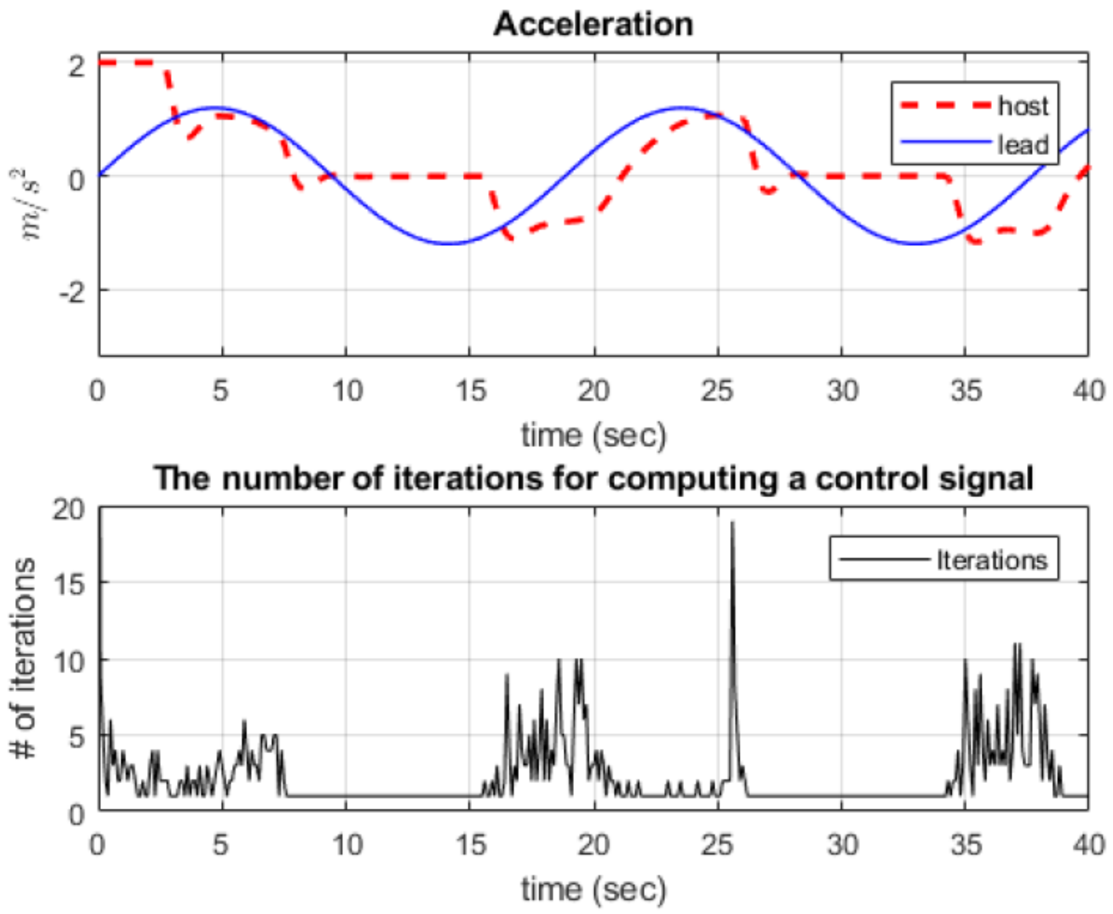


  - 20x more exec. time (15—28 secs)

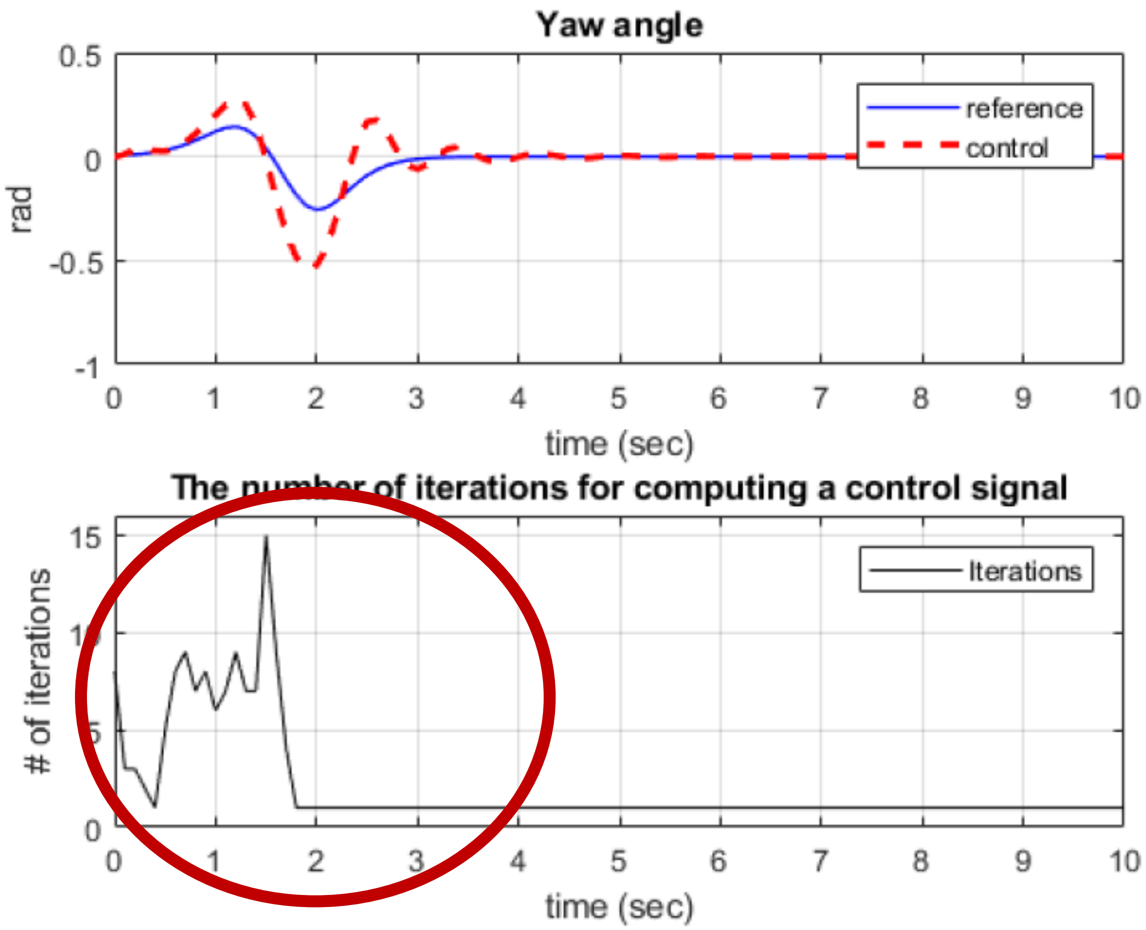


- Highly dynamic over a wide range

Motivational Simulation Results



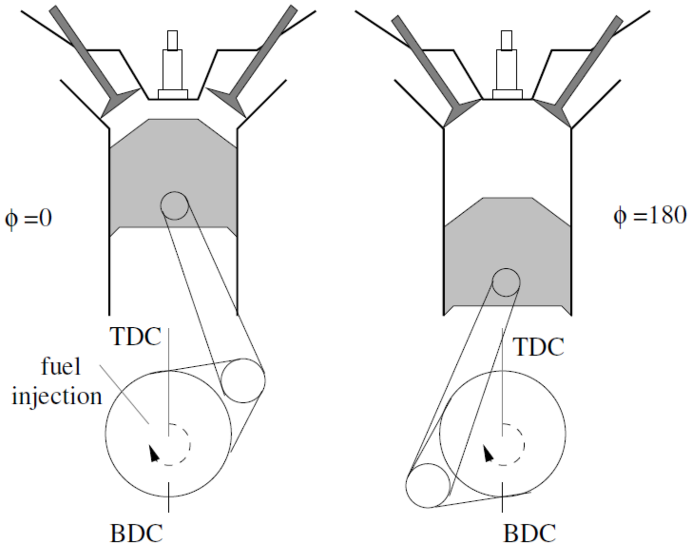
Adaptive Cruise Control (ACC)



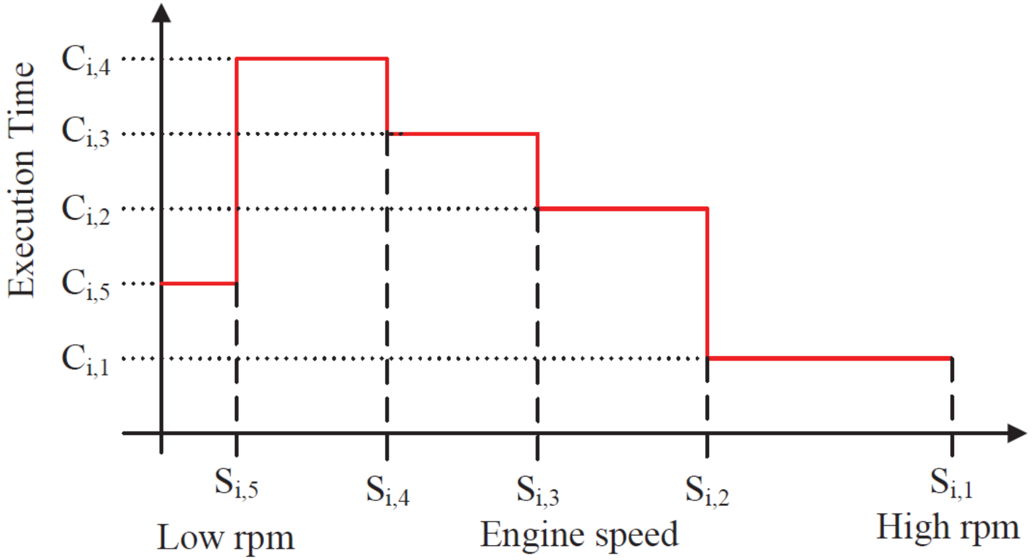
Active Vehicle Steering (AVS)

Other Applications

- Engine Control Module
  - Strong correlation between physical state and resource demand
  - Speed of the engine crankshaft's rotation



[Biondi *et al.*,14]

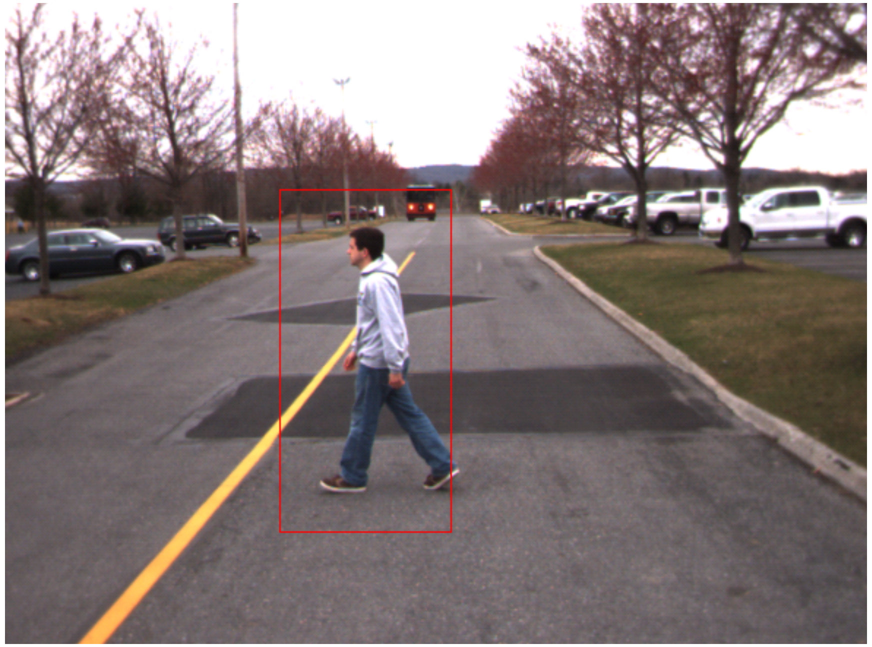


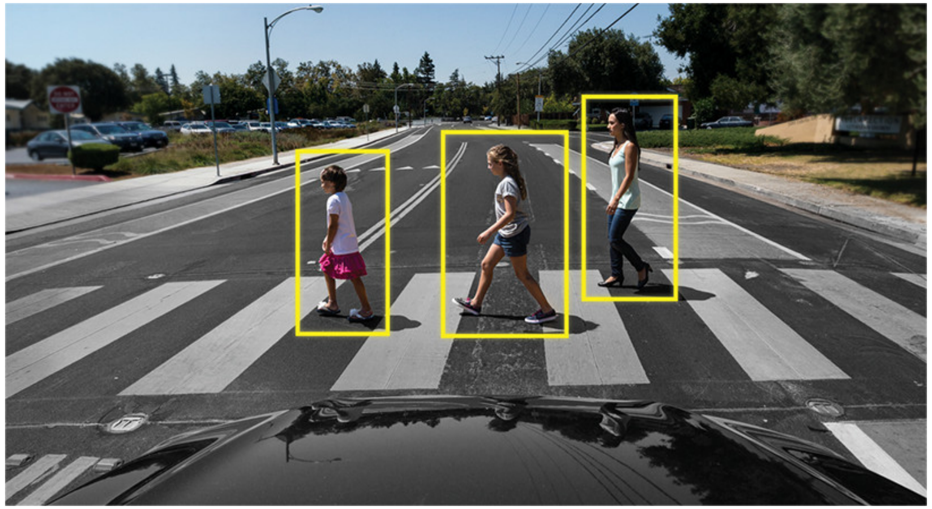
[Davis *et al.*,14]

Biondi *et al*. Exact interference of adaptive variable-rate tasks under fixed-priority scheduling. In ECRTS, 2014.
Davis *et al*. Schedulability tests for tasks with variable rate-dependent behavior under fixed priority scheduling. In RTAS, 2014.

Other Applications

- Vision-based Object Detection
  - Strong correlation between physical state and resource demand
  - The number of objects in the camera's field-of-view

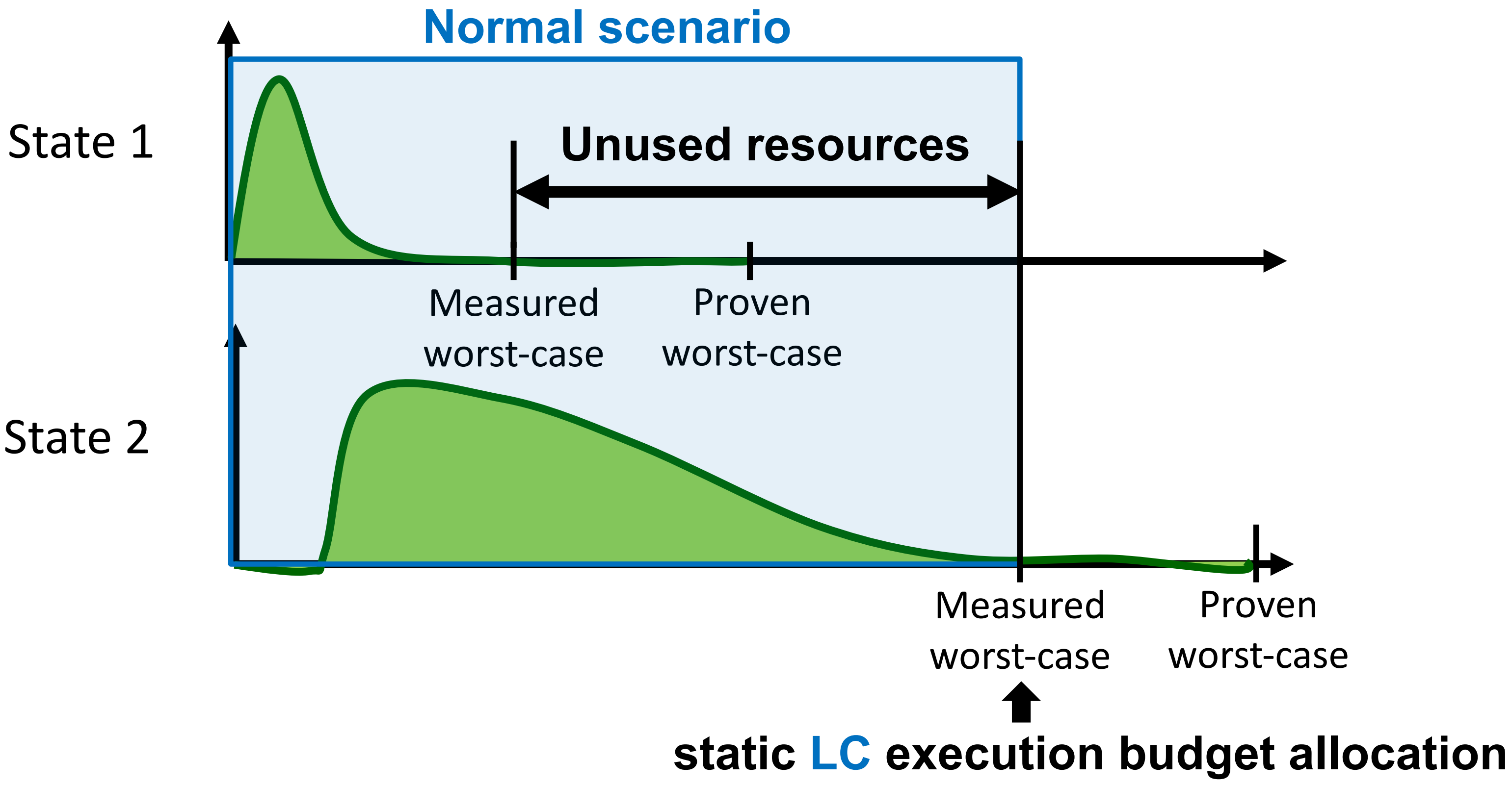




Niz *et al*. On resource overbooking in an unmanned aerial vehicle. In ICCPS, 2012.

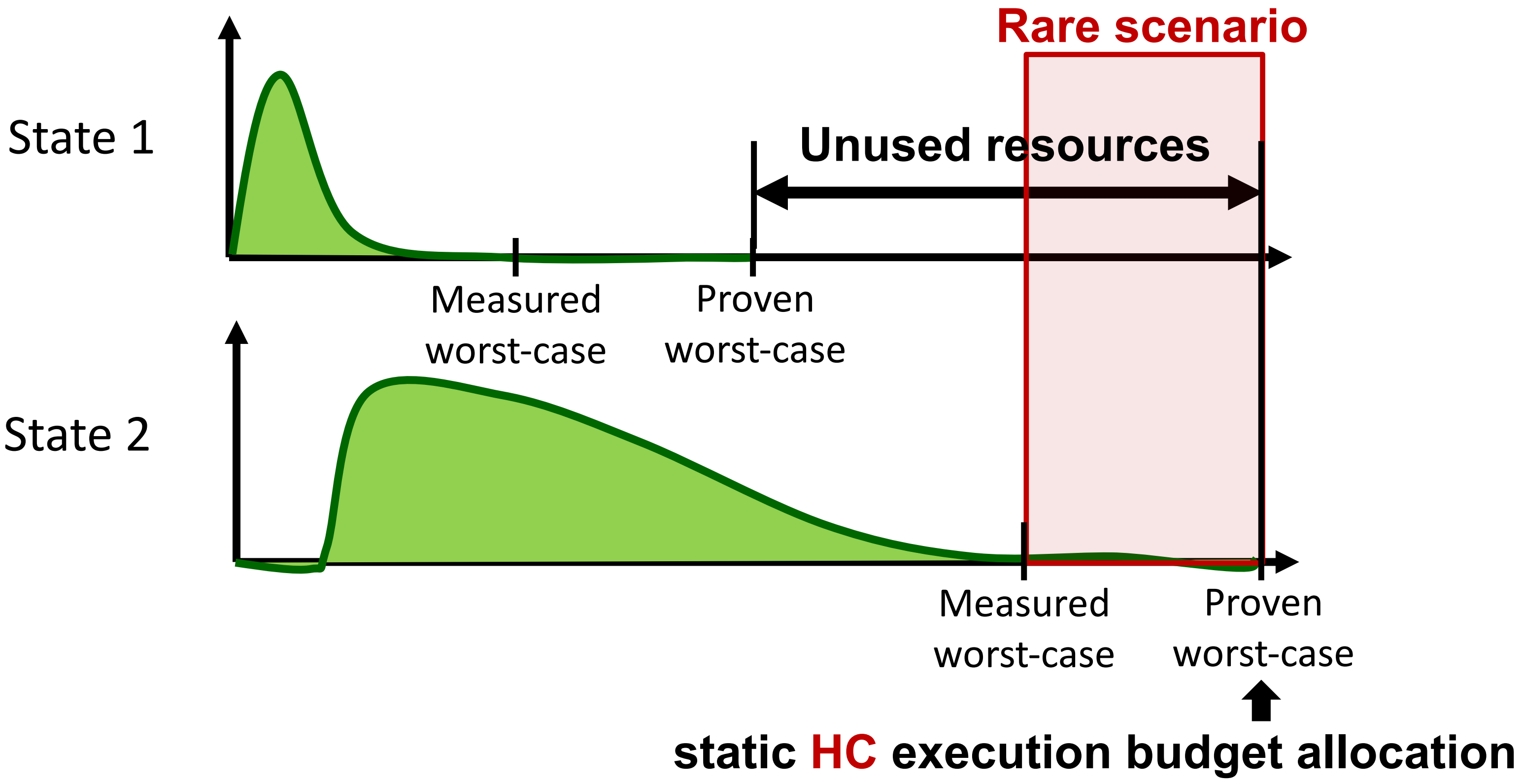
Implication

- If dynamic execution behavior is not considered,



Implication

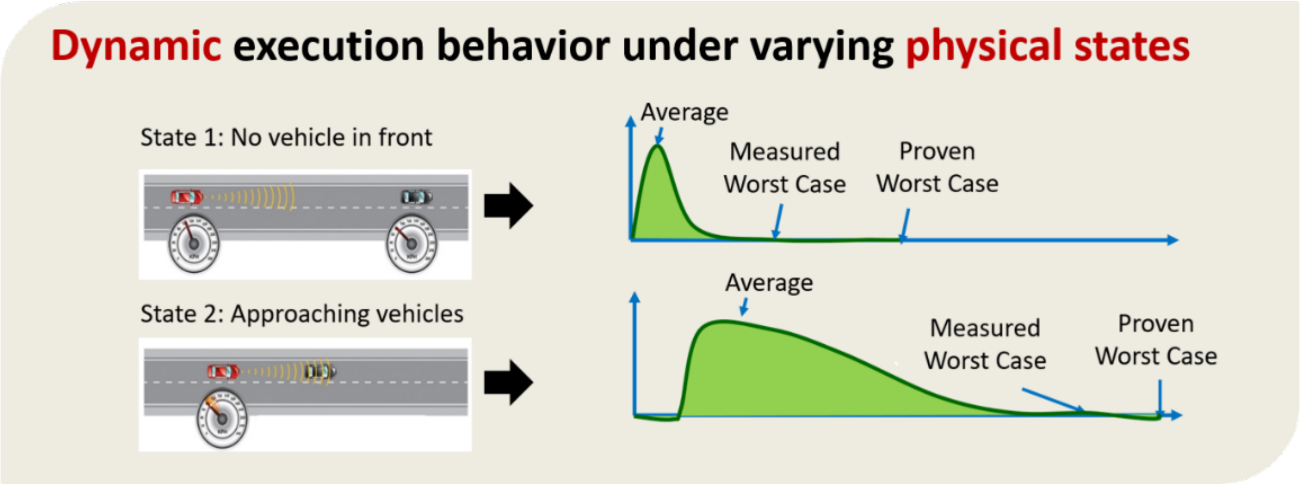
- If dynamic execution behavior is not considered,



Resource under-utilization or **service degradation in LC tasks**

Our Goal

- Motivation



- Goal

Physical-State-Aware Dynamic Slack Management for MC Systems

  - Minimize the number of LC job drops without compromising MC-schedulability

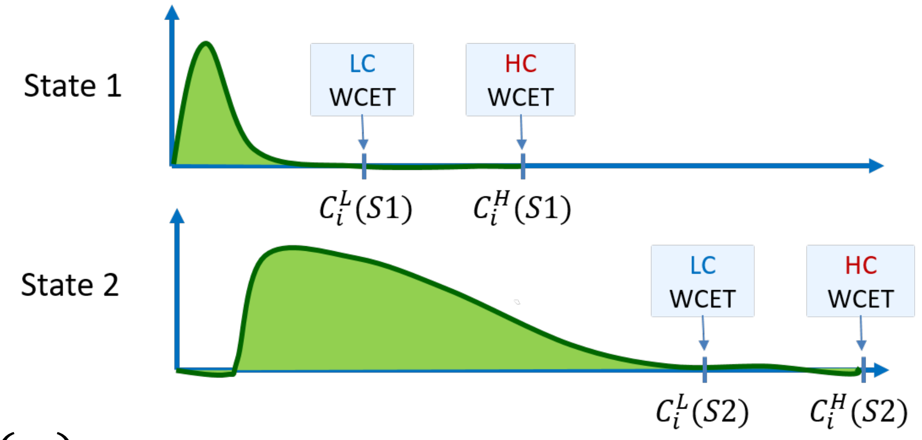
Challenge

Physical-State-Aware Dynamic Slack Management for MC Systems

- Q1. How to capture varying resource demands with physical state?
  - New MC task model
- Q2. How to calculate a dynamic slack?
  - New slack concepts for MC systems
- Q3. How to schedule the slack under varying physical state?
  - Physical-state-aware dynamic resource allocation

Physical-State-Aware MC Task Model

- Task $\tau_i = (T_i, C_i, D_i, L_i)$, where
  - $L_i \in \{LC, HC\}$;
    - LC – low-critical task, HC – high-critical task
  - $\boldsymbol{C_i = \{C_i^L(s_i), C_i^H(s_i)\}}$;
    - **Physical state s_i**
    - for LC task $C_i^L(s_i) = C_i^H(s_i)$ and for HC task $C_i^L(s_i) \le C_i^H(s_i)$
  - Generalization of the Vestal's task model



- **MC-Schedulable**
  - LC-mode guarantee: if no task executes beyond LC-WCET
    - Every job finishes its execution (≤ LC-WCET) before its deadline.
  - HC-mode guarantee: if any HC task executes beyond LC-WCET (**mode-switch**)
    - Every HC job finishes its execution (≤ HC-WCET) before its deadline.

New Slack Concept

- Resource allocation in each mode (according to MC-Schedulability)

LC-mode allocation	HC-mode allocation
Both LC and HC jobs get LC-WCET resource budget	Only HC jobs get HC-WCET resource budget

- New slack concepts for MC scheduling
  - **LC-mode slack $S_{LC}(t_1, t_2)$**
    - The amount of idle time in $[t_1, t_2)$ under LC-mode resource allocation without compromising LC-mode guarantee
  - **HC-mode slack $S_{HC}(t_1, t_2)$**
    - The amount of idle time in $[t_1, t_2)$ under HC-mode resource allocation without compromising HC-mode guarantee

Physical-State-Aware Dynamic Slack Management

- Focus on EDF-VD [Baruah *et al*. 12]

- Runtime slack scheduling
  - LC/HC-mode slack scheduling
  - Slack-based mode-switch mechanism

  ← **How to utilize LC/HC-mode slack**

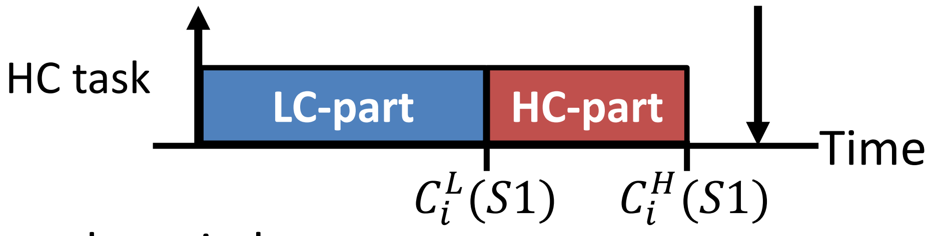
- Physical-state-aware dynamic resource allocation
  - Slack updates
  - Slack calculation

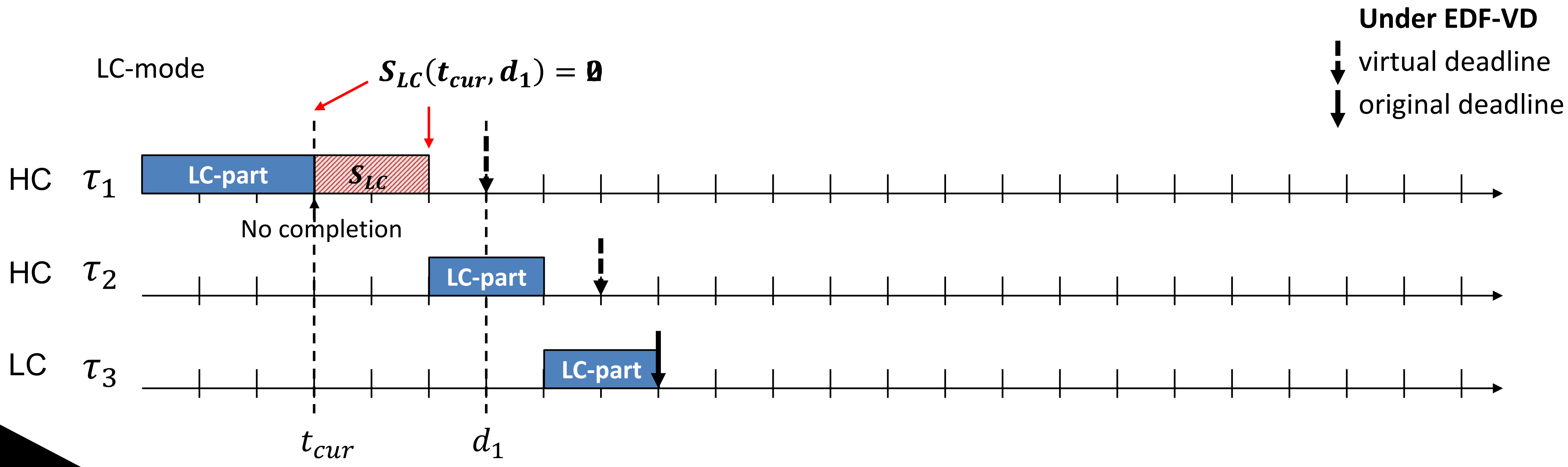
  ← **How to update/calculate slack**

Baruah *et al*. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In ECRTS, 2012.

Physical-State-Aware Dynamic Slack Management

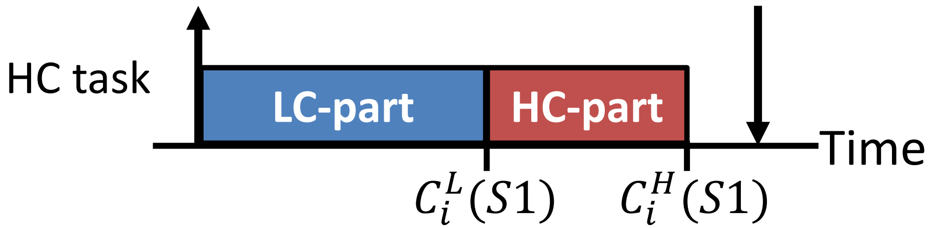
- Runtime slack scheduling
  - **LC-mode slack $S_{LC}(t_1, t_2)$** in LC-mode
    - Executing HC jobs' HC-part execution without triggering a mode-switch

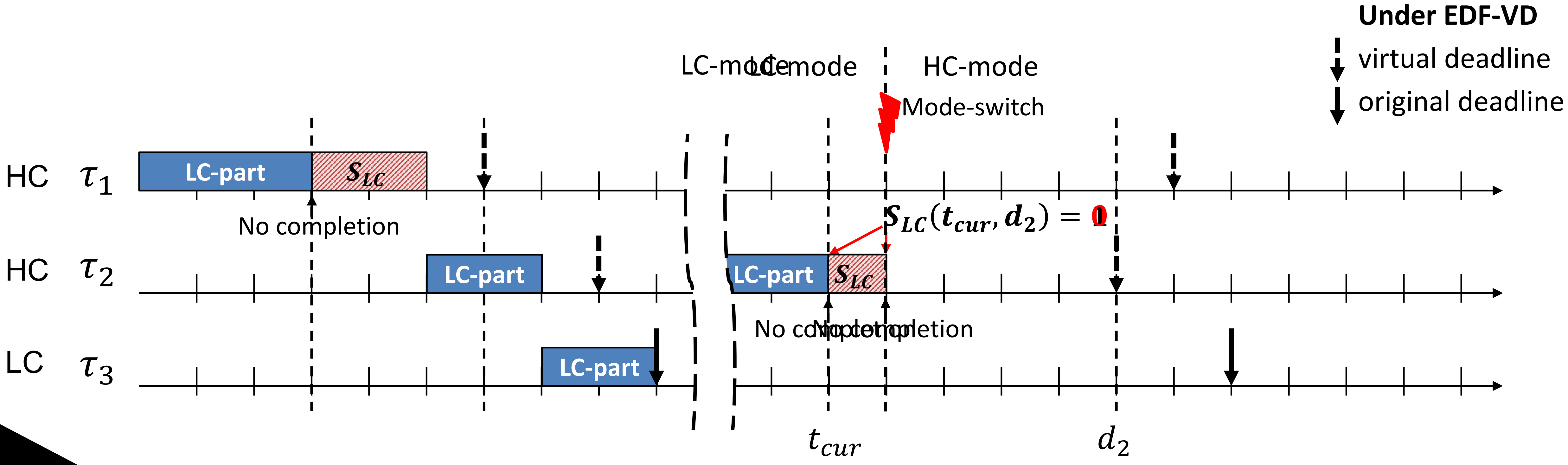




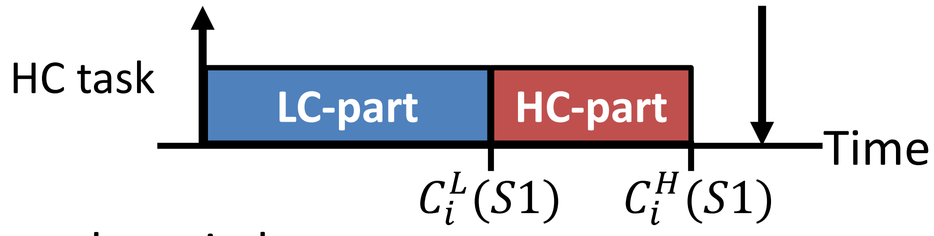
Physical-State-Aware Dynamic Slack Management

- Runtime slack scheduling
  - **LC-mode slack $S_{LC}(t_1, t_2)$** in LC-mode
    - Executing HC jobs' HC-part execution without triggering a mode-switch
  - Slack-based mode-switch mechanism
    - Triggering a mode-switch when $S_{LC}(t_1, t_2) = 0$ with no completion

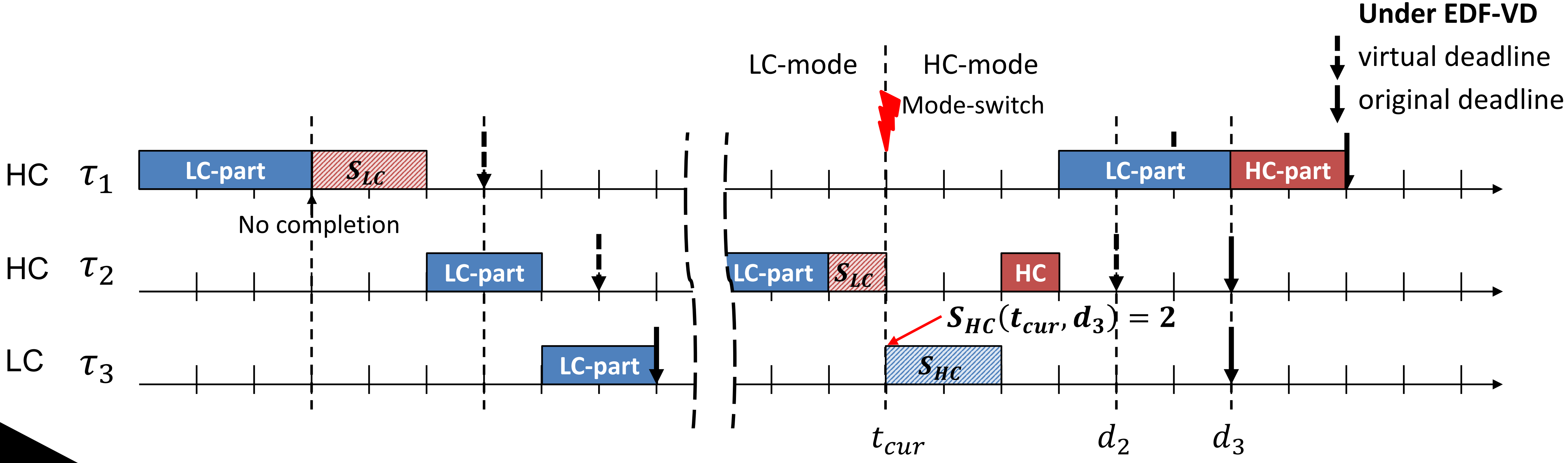




Physical-State-Aware Dynamic Slack Management

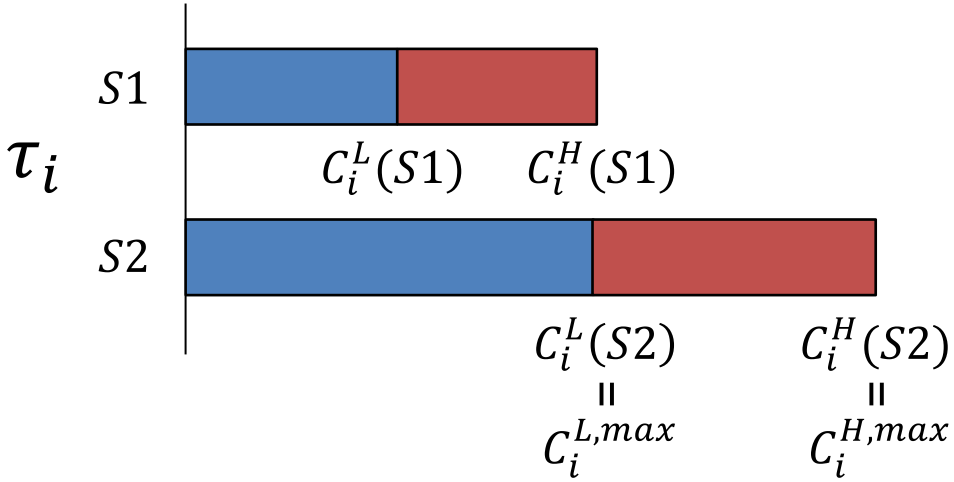


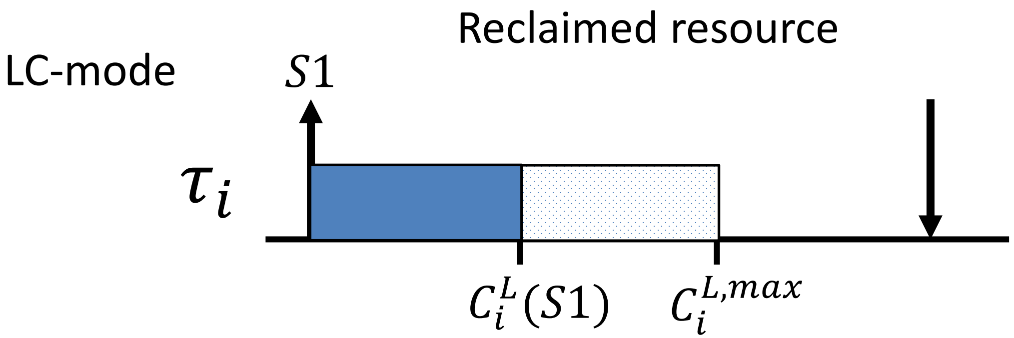
- Runtime slack scheduling
  - **LC-mode slack $S_{LC}(t_1, t_2)$** in LC-mode
    - Executing HC jobs' HC-part execution without triggering a mode-switch
  - **HC-mode slack $S_{HC}(t_1, t_2)$** in HC-mode
    - Executing LC jobs' LC-part execution without compromising other HC jobs' execution



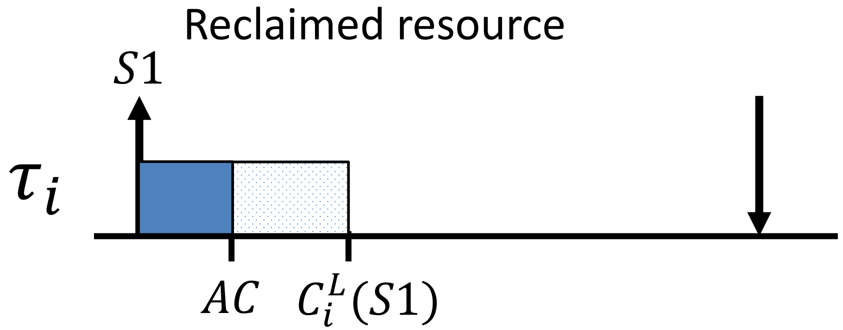
Physical-State-Aware Dynamic Slack Management

- Physical-state-aware dynamic resource allocation
  - Runtime slack update
    - Before JOB-RELEASE
      - Allocate $C_i^{M,max}$ execution budget, $M \in \{LC, HC\}$
    - Upon JOB-RELEASE **with physical state S**
      - Reclaim $C_i^{M,max} - \mathrm{C_i^M(S)}$





    - Upon JOB-COMPLETION with actual execution time AC
      - Reclaim $\mathrm{C_i^M(S) - AC}$



Physical-State-Aware Dynamic Slack Management

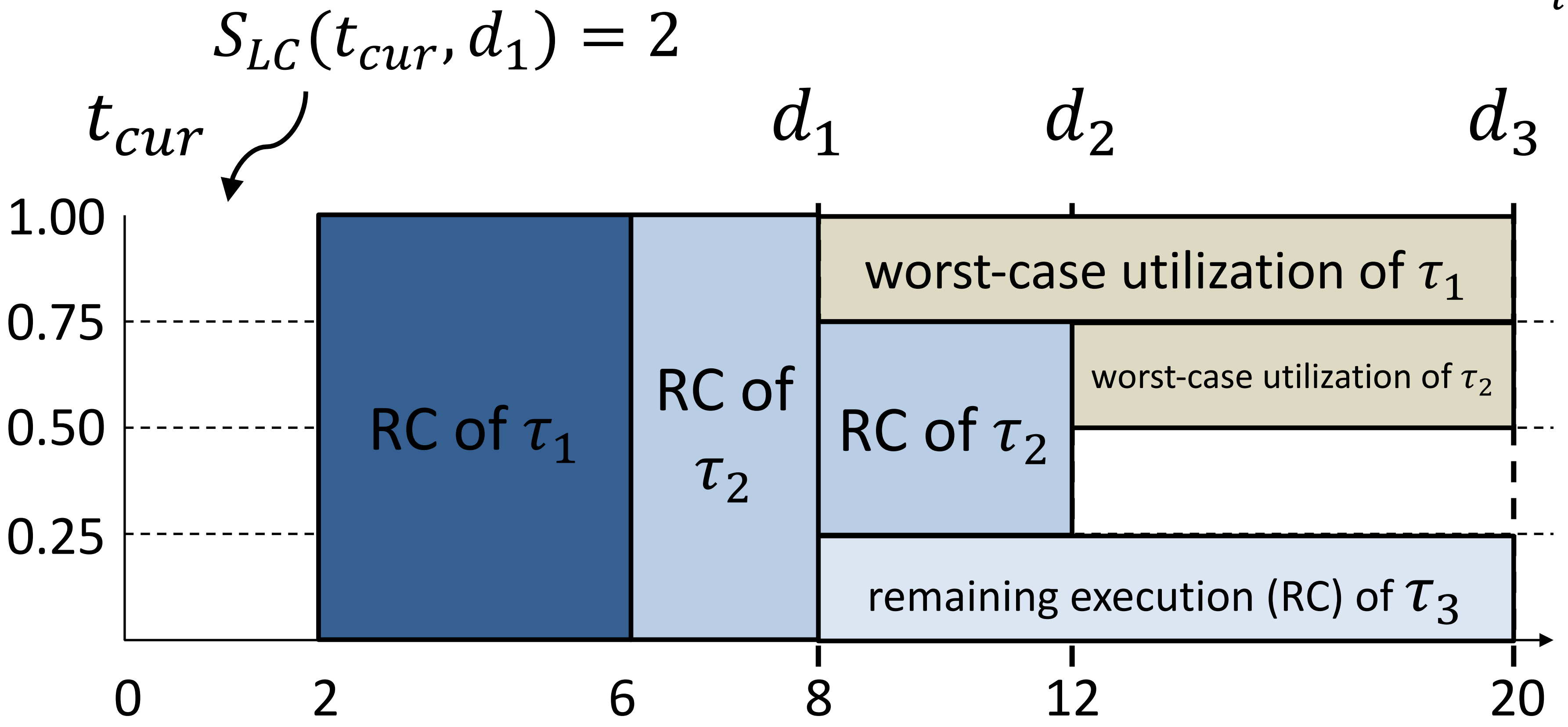
- Physical-state-aware dynamic resource allocation
  - Slack calculation
    - Find max. slack time available in $[t_{cur}, d_1(t_{cur}))$

➡ { LC-mode slack: delay **mode-switch** / HC-mode slack: delay **LC job drops** } as late as possible

Physical-State-Aware Dynamic Slack Management

- Physical-state-aware dynamic resource allocation
  - Slack calculation: LC-mode slack
    - In reverse EDF order

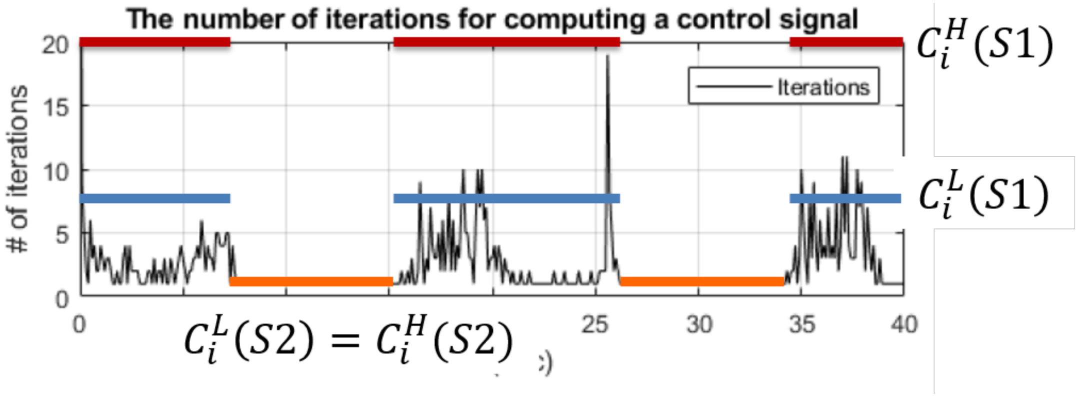
$$U = U^L_{\tau^L} + \frac{1}{x} \cdot U^L_{\tau^H} \leq 1$$



Evaluation

- Case study: ADAS system
  - 2 HC tasks: ACC and AVS
    - Period: 100ms
    - Actual execution time traces from a real driving scenario

<ACC HC task>

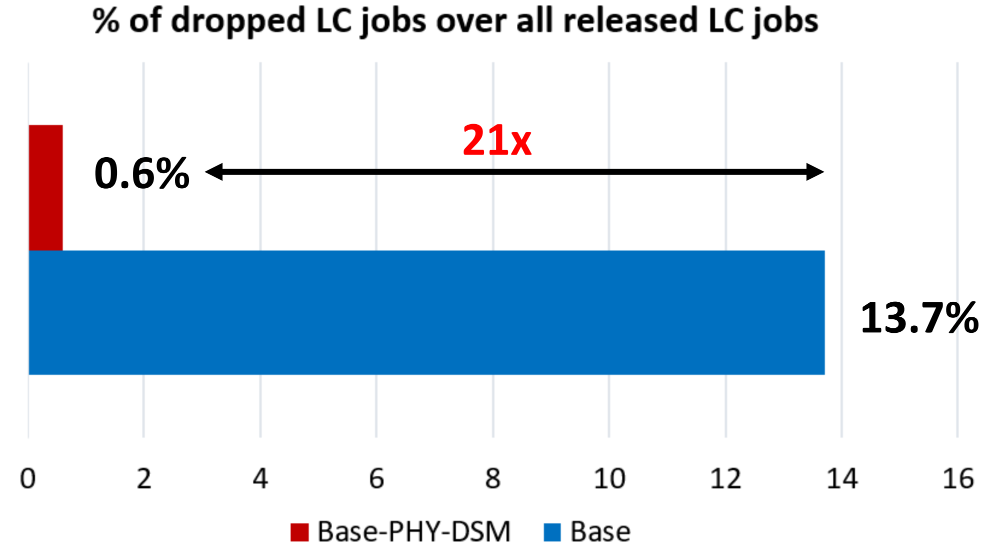


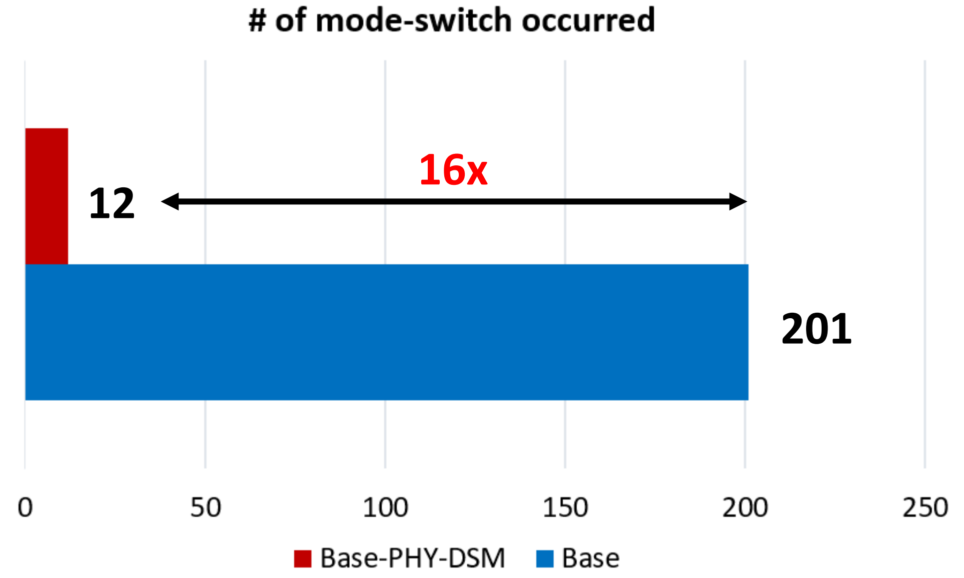
  - 4 LC tasks

(ms)	LC task 1	LC task 2	LC task 3	LC task 4
Period T_i	200	200	80	50
$C_i^L(s_i)$	$\{61, 17\}$	$\{35, 10\}$	$\{5, 2\}$	$\{7, 3\}$

Evaluation

- Case study: ADAS system
  - Simulation results for 80 seconds



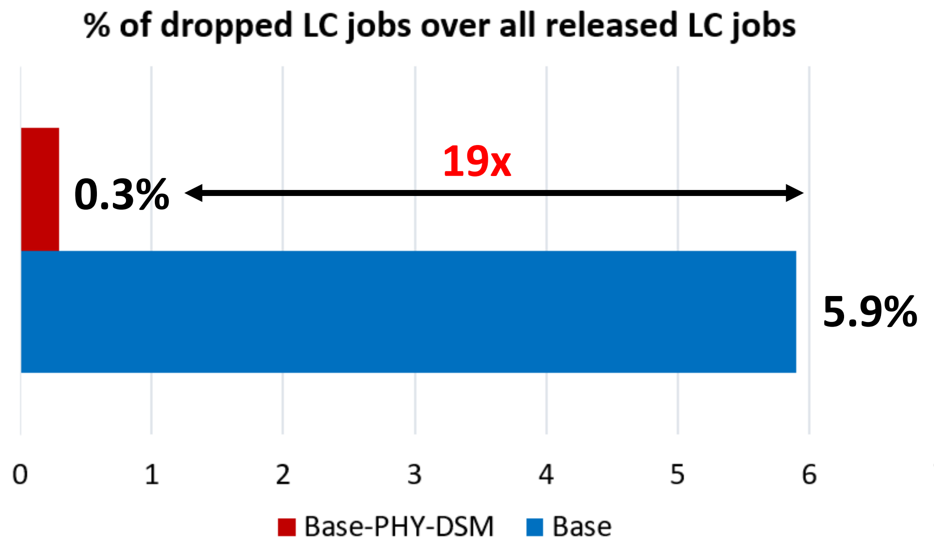


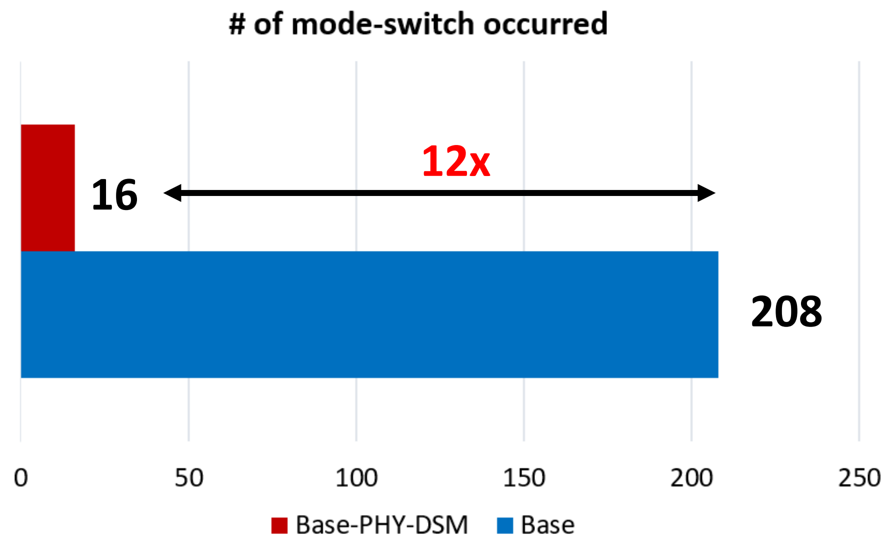
By utilizing 2% slack time of total simulation time (1,753/80,000 ms)

Base-PHY-DSM: EDF-VD with the physical-state-aware dynamic slack management framework
Base: EDF-VD with the classic MC task model

Evaluation

- Extensive simulations
  - Synthetic task sets
    - # of tasks: 4, 6, 8
    - 300 task sets



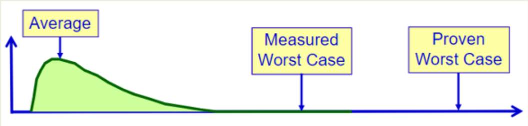


Summary

Different criticality levels

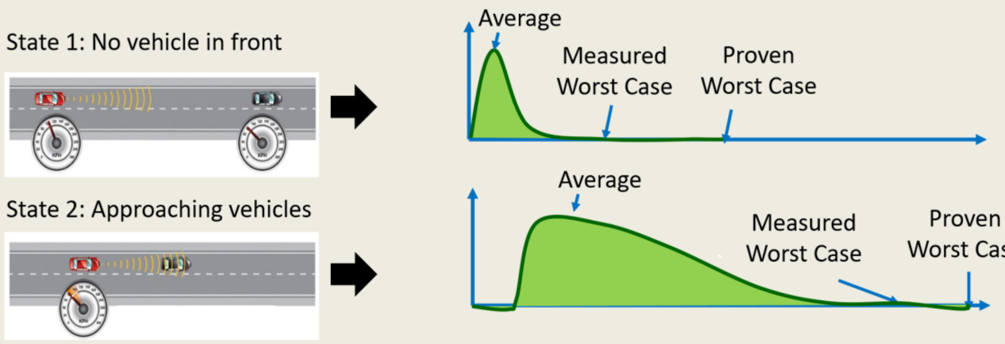
A	Low
B	Medium
C	High
D	Critical

Multiple WCET estimates



✚

Dynamic execution behavior under varying physical states



Proposed **new MC task model & slack concept** that capture varying resource demands

Developed **dynamic slack management** that enables adaptive resource allocation

Enhanced **the performance of low-criticality tasks** significantly

PHYSICAL-STATE-AWARE DYNAMIC SLACK MANAGEMENT FOR MIXED-CRITICALITY SYSTEMS

Q&A?