A Formal Approach towards Safe and Stable Schedule Synthesis in Weakly Hard Control Systems

Authors: ***Debarpita Banerjee*** [1], *Parasara Sridhar Duggirala* [2], *Bineet Ghosh* [3], *Sumana Ghosh* [1]

EMSOFT 2025

[1] **Debarpita Banerjee, Sumana Ghosh**
Indian Statistical Institute,
Kolkata, India
{debarpita2023_r, sumana} @isical.ac.in

[2] **Parasara Sridhar Duggirala**
The University of North Carolina,
Chapel Hill, USA
psd@cs.unc.edu

[3] **Bineet Ghosh**
The University of Alabama,
Tuscaloosa, USA
bineet@ua.edu

Introduction: Plant-Control Closed-Loop Systems

An autonomous car

A dynamical system like a physical process

(Plant)

Introduction: Plant-Control Closed-Loop Systems

Speed sensor measures
the car's actual speed

An autonomous car

Introduction: Plant-Control Closed-Loop Systems

Speed sensor measures the car's actual speed

Cruise controller of the car

An autonomous car

Stabilizing Feedback Controller

Introduction: Plant-Control Closed-Loop Systems

Cruise controller of the car

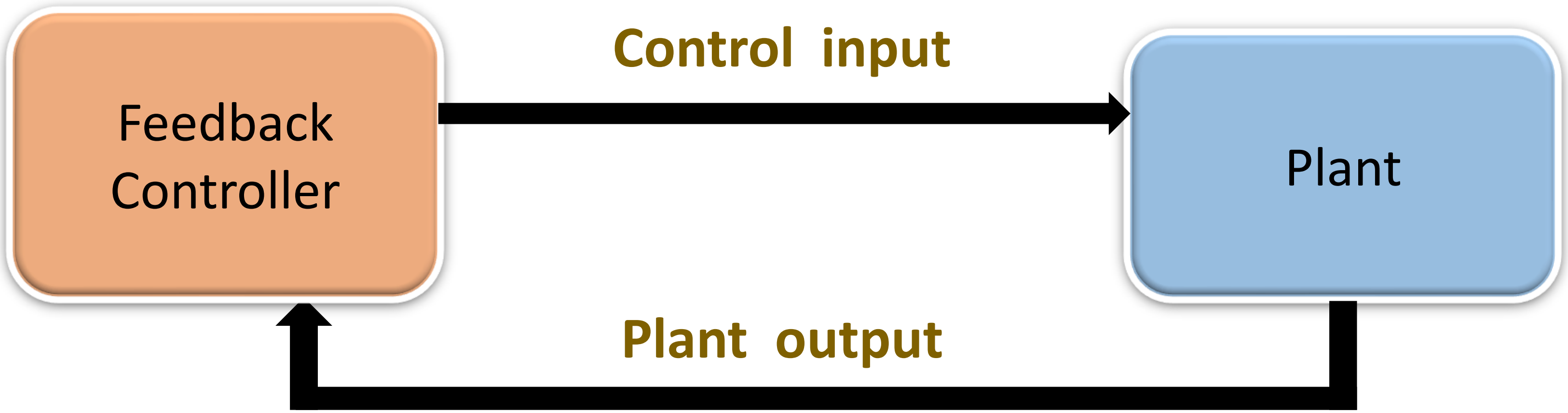
Compares actual speed with the reference speed

Speed sensor measures the car's actual speed

An autonomous car

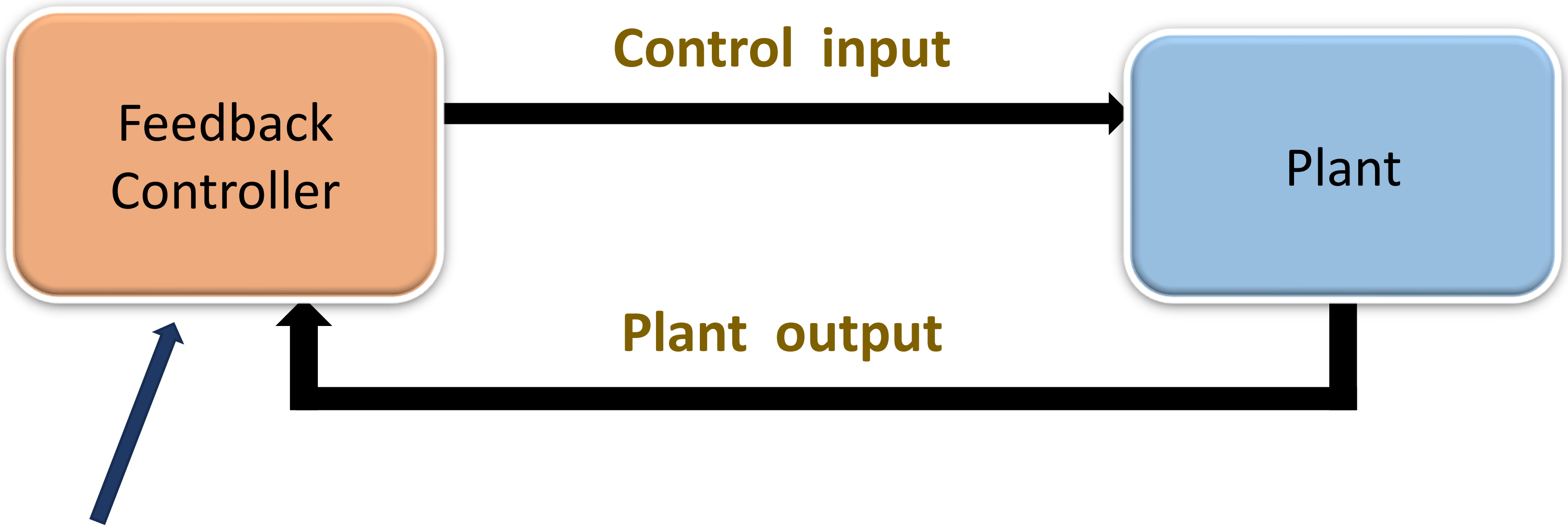
Control Input:
Adjusts the throttle force

Introduction: Plant-Control Closed-Loop Systems



Plant-Control Closed-Loop System

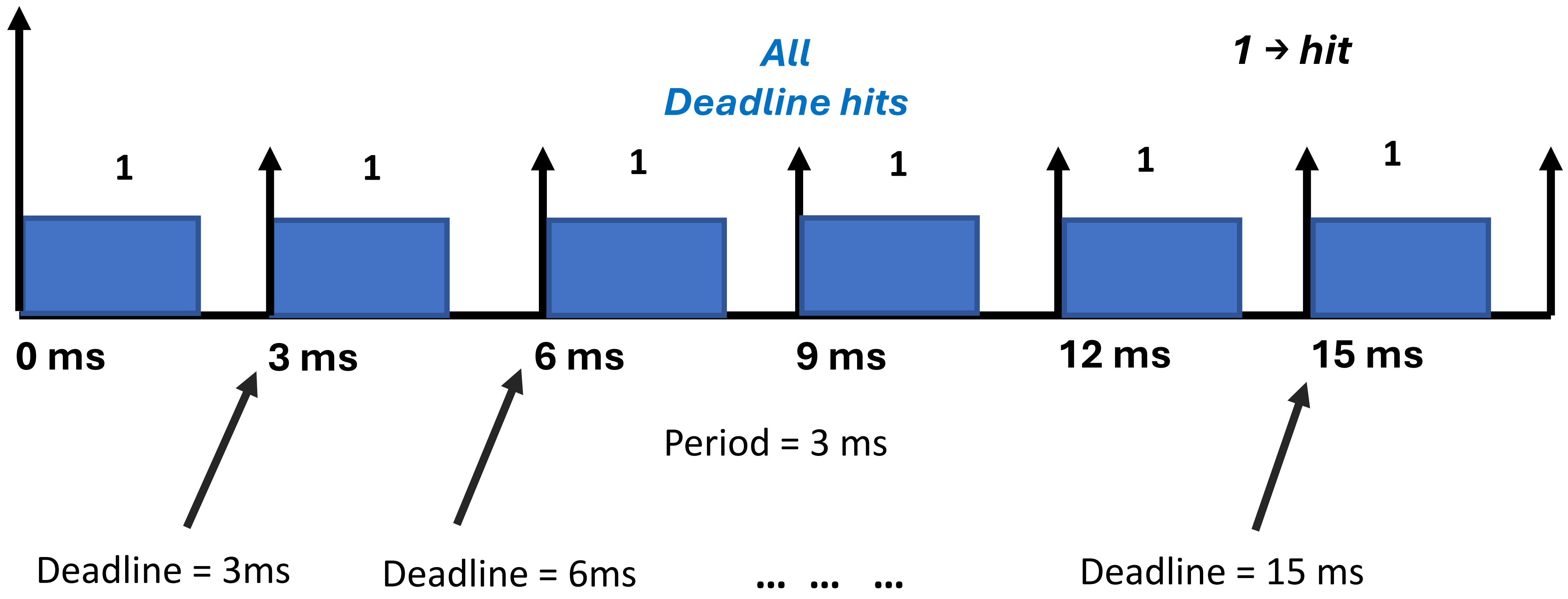
Introduction: Plant-Control Closed-Loop Systems



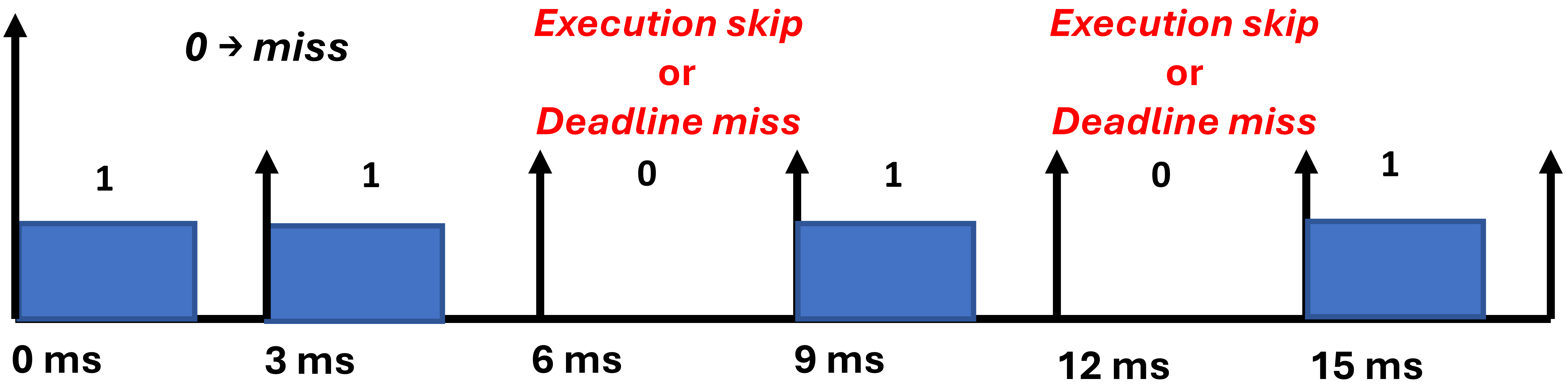
Plant-Control Closed-Loop System

- Discrete-time controller executed as a ***software control task*** in the embedded processor.
- Control Task: periodic real-time task, should complete execution before its ***deadline.***
- This is the task's ***hard real-time requirement*** to satisfy.

Introduction: Scheduling Control Task in Hard Real-Time Setting



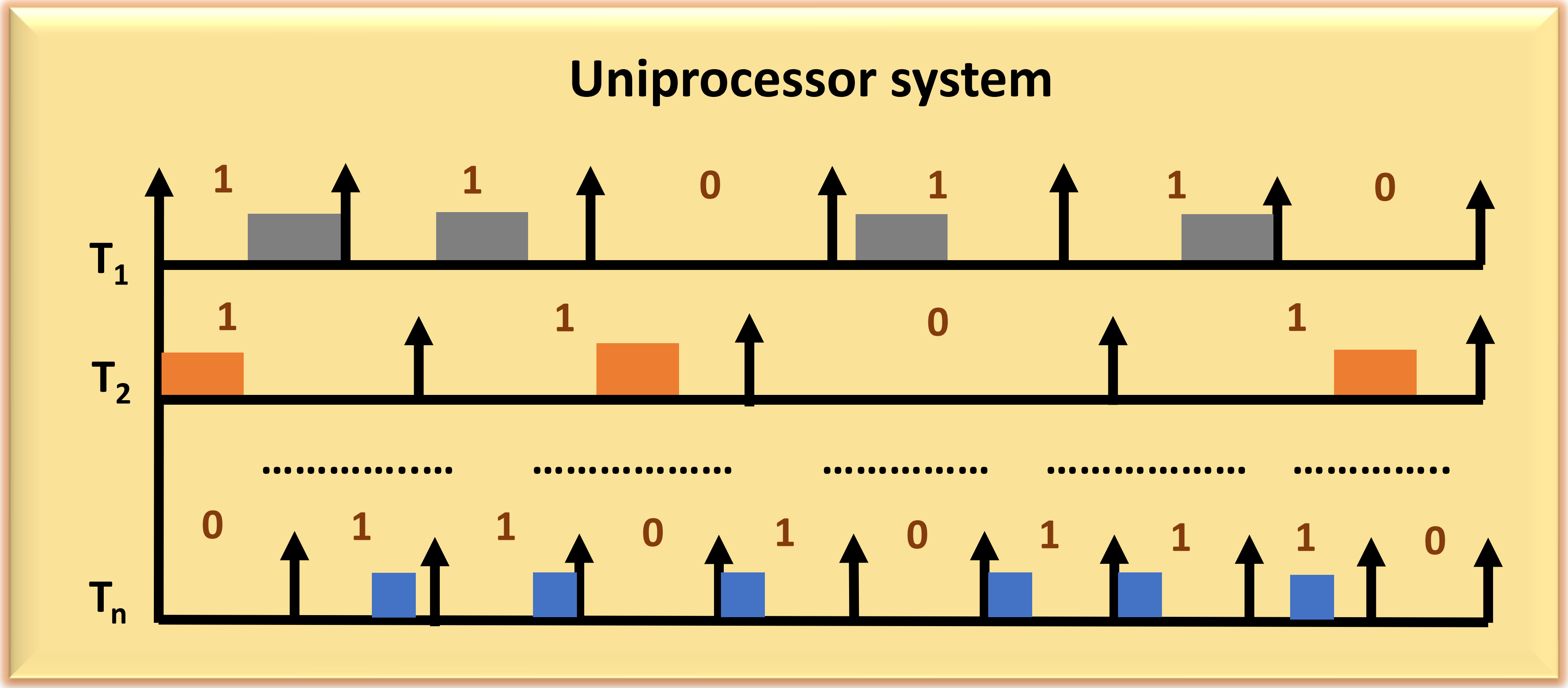
Introduction: Scheduling Control Task in Weakly Hard Setting



(4, 6)-firm weakly hard constraint

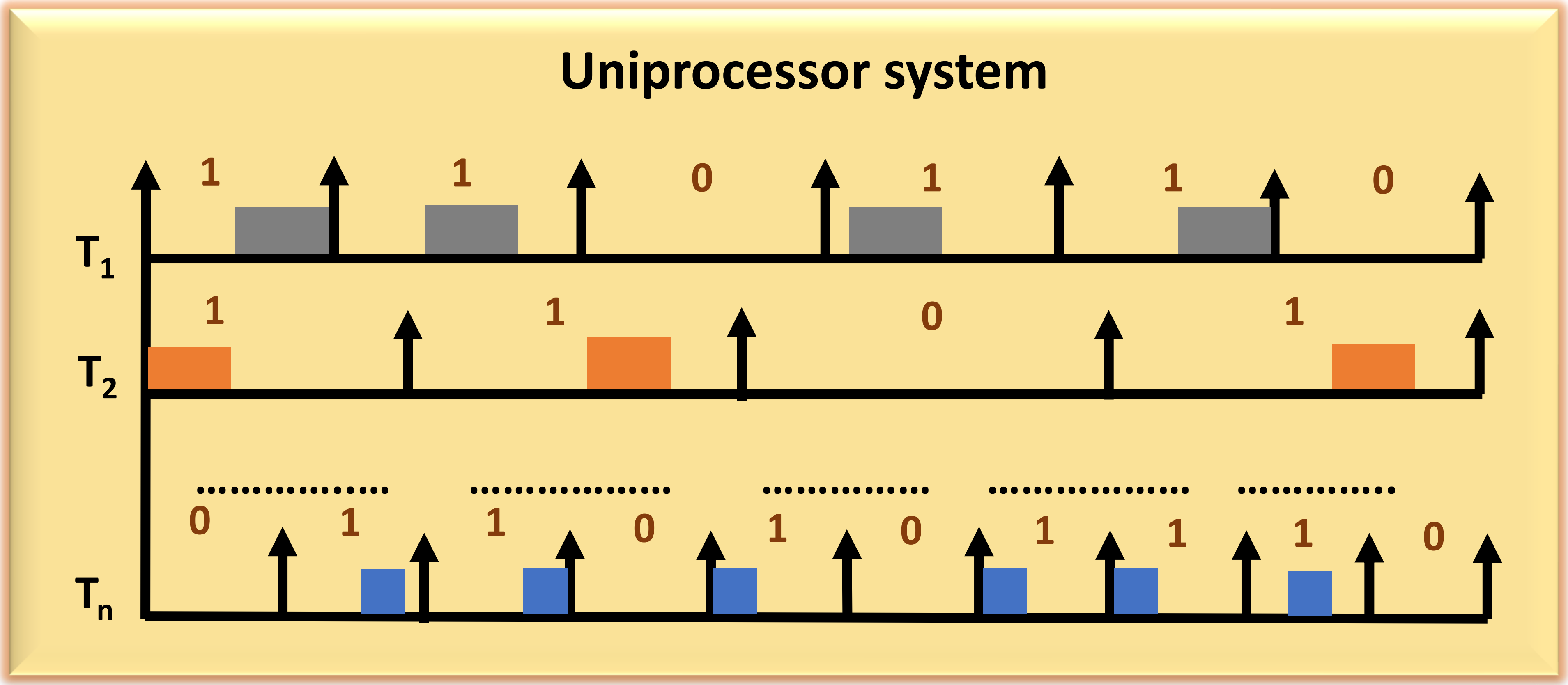
Deadline hit-miss pattern: **Control Execution Sequence (CES)** e.g., **110101**

Scheduling Multiple Weakly Hard Control Tasks in a Processor



n weakly hard control tasks in a processor corresponding to the *n* controllers

Motivation of the Work



Does such a schedule ensure the underlying ***stability*** of the control systems ?

Does such a schedule ensure the underlying ***safety*** of the control systems ?

Limitations of Existing Methods and Contributions of this Work

Limitations:

- In the context of *schedulability*: Existing methods focus **either** on *stability* [1,2] **or** safety [3,4].
- Blend: (stability, safety, schedulability) is **missing**.
- Schedule ensures **safety only over a bounded time horizon** [3,4].

---

1. A structured methodology for pattern based adaptive scheduling in embedded control, S. Ghosh et al., ACM-TECS 2017.
2. Closing the gap between stability and schedulability: A new task model for cyber-physical systems, Hoon Sung Chwa et al., RTAS 2018.
3. Statistical approach to efficient and deterministic schedule synthesis for cyber-physical systems, Shengjie Xu et al., ATVA 2023.
4. Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints, Shengjie Xu et al., ASP-DAC 2023.

Limitations of Existing Methods and Contributions of this Work

Limitations:

- In the context of *schedulability*: Existing methods focus **either** on *stability* [1,2] **or** safety [3,4].
- Blend: (stability, safety, schedulability) is **missing**.
- Schedule ensures **safety only over a bounded time horizon** [3,4].

1. A structured methodology for pattern based adaptive scheduling in embedded control, S. Ghosh et al., ACM-TECS 2017.
2. Closing the gap between stability and schedulability: A new task model for cyber-physical systems, Hoon Sung Chwa et al., RTAS 2018.
3. Statistical approach to efficient and deterministic schedule synthesis for cyber-physical systems, Shengjie Xu et al., ATVA 2023.
4. Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints, Shengjie Xu et al., ASP-DAC 2023.

Contributions:

Addressing *(stability, safety, schedulability)* for the **first time** → **schedule** ensures **stability and safety** over an **unbounded time horizon.**

Stepwise exploring the 3 aspects:

1. Ensuring exponential **stability**
2. Ensuring **safety** for infinite time length
3. Synthesizing safe and stable **schedule**

Step 1: Ensuring Stability

Ensuring Stability

(l, ∈)-exponential stability criterion

Every $\boldsymbol{l}$-length ratio of norm $||\boldsymbol{x}||$ decreases by damping ratio of $\boldsymbol{\epsilon}$.

i.e., $\frac{||\boldsymbol{x}[\boldsymbol{k}+\boldsymbol{l}]||}{||\boldsymbol{x}[\boldsymbol{k}]||} < \boldsymbol{\epsilon}$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

(l, ∈)-exponential stability criterion

Every $\boldsymbol{l}$-length ratio of norm $||\boldsymbol{x}||$ decreases by damping ratio of $\boldsymbol{\epsilon}$.

i.e., $\frac{||\boldsymbol{x}[\boldsymbol{k}+\boldsymbol{l}]||}{||\boldsymbol{x}[\boldsymbol{k}]||} < \boldsymbol{\epsilon}$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

Compute *(l, ∈)*-exponential stability criterion

$$N = \left\lceil \frac{S}{h} \right\rceil,$$

$$l = \left\lceil \frac{N}{f} \right\rceil,$$

$$\epsilon = \left(\frac{\xi}{\xi + \delta} \right)^{\frac{1}{f}}$$

h: sampling period
f : tuning parameter

(l, ∈)-exponential stability criterion

Every l-length ratio of norm $||x||$ decreases by damping ratio of ϵ.

i.e., $\frac{||x[k+l]||}{||x[k]||} < \epsilon$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

Compute *(l, ∈)*-exponential stability criterion

Compute minimum control execution rate ***r***

$$\beta = \frac{ln\left(\frac{1}{\epsilon}\right)}{l \times h},$$

$$r = \frac{2\, ln(\beta) + ln(\chi_0)}{ln(\chi_0) - ln(\chi_1)}$$

χ_0 , χ_1 : spectral radii of the open loop and closed loop dynamics matrices

(*l*, ∈)-exponential stability criterion

Every l-length ratio of norm $||x||$ decreases by damping ratio of ϵ.

i.e., $\frac{||x[k+l]||}{||x[k]||} < \epsilon$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

Compute *(l, ∈)*-exponential stability criterion

Compute minimum control execution rate ***r***

Compute ***stable (M, K)-firm*** constraint

$M = \lceil r \times l \rceil$,

$K = l$

(l, ∈)-exponential stability criterion

Every l-length ratio of norm $||x||$ decreases by damping ratio of ϵ.

i.e., $\frac{||x[k+l]||}{||x[k]||} < \epsilon$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

→ Compute *(l, ∈)*-exponential stability criterion

↓

Compute minimum control execution rate ***r***

↓

Compute ***stable (M, K)-firm*** constraint

↓

Ensuring system's ***exponential stability***

$$\frac{\|x[k+l]\|}{\|x[k]\|} < \epsilon$$

(*l*, ∈)-exponential stability criterion

Every l-length ratio of norm $\|x\|$ decreases by damping ratio of ϵ.

i.e., $\frac{\|x[k+l]\|}{\|x[k]\|} < \epsilon$

Ensuring Stability

Inputs

Settling time (S)
+
Reference value (ξ)
+
Maximum disturbance allowed at input (δ)

Compute *(l, ∈)*-exponential stability criterion

Compute minimum control execution rate ***r***

Compute ***stable (M, K)-firm*** constraint

Ensuring system's ***exponential stability***

System's state converges to equilibrium point with ***exponential decay rate***

$\|\boldsymbol{x}[\boldsymbol{k}]\| \rightarrow \mathbf{0}$

(*l*, ∈)-exponential stability criterion

Every $\boldsymbol{l}$-length ratio of norm $\|\boldsymbol{x}\|$ decreases by damping ratio of $\boldsymbol{\epsilon}$.

i.e., $\frac{\|\boldsymbol{x}[\boldsymbol{k}+\boldsymbol{l}]\|}{\|\boldsymbol{x}[\boldsymbol{k}]\|} < \boldsymbol{\epsilon}$

SASO Constraint

Stable
(M, K)-firm constraint

Compute *(m, k)-firm* SASO constraint

SASO: Scheduling-Affable Stability-Oriented

m and k → small divisors of M and K respectively

- *Scheduling affable:* SASO constraints speed up the scheduling process.
- SASO constraints enhance the control performance:
  CESs following SASO constraints avoid scenarios of missing deadlines consecutively often.

Step 2: Ensuring Safety

Criterion for Safety

Bounded deviation from ideal behavior or nominal trajectory

Criterion for Safety

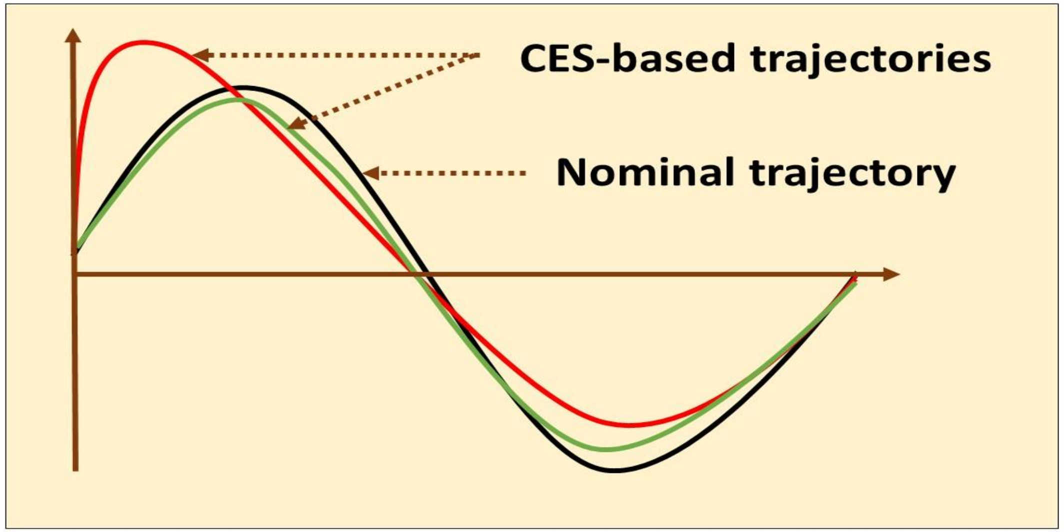
Bounded deviation from ideal behavior or nominal trajectory :

- ***Nominal trajectory (N):*** State evolution trajectory for all deadlines met, i.e., pattern '111…' .

  ideal behavior of the system

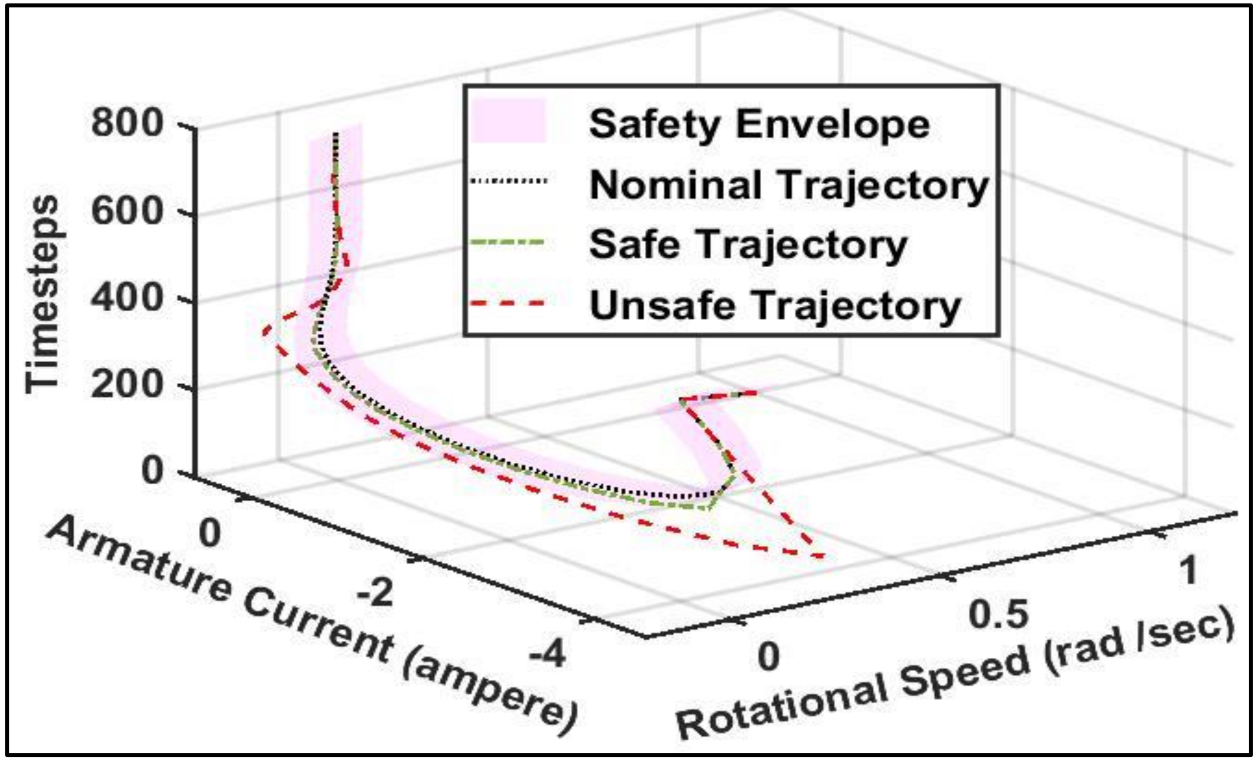
- ***CES-based trajectory (C_P):*** State evolution trajectory for deadline hit-miss pattern (CES), e.g., 11010.

  system's behavior with deadline misses



Criterion for Safety

Bounded Deviation: The deviation must be bounded, i.e., $\boldsymbol{dis(N, C_p) \leq d^{safe}}$, where d^{safe} is the safety bound.



DC motor speed control system

Deviation of C_P from N:
$\boldsymbol{dis(N, C_p)}$

(measured in terms of Euclidean distance)

Ensuring Safety

Input

(m, k)-firm SASO constraint

Ensuring Safety

Input

(m, k)-firm SASO constraint

→

SASSO constraint = (m’, k)

SASSO: Scheduling-Affable Stability-and-Safety-Oriented

Ensuring Safety

Input

(m, k)-firm SASO constraint

SASSO constraint = (m', k)

SASSO: Scheduling-Affable Stability-and-Safety-Oriented

Deducing a **safe CES** *p*

satisfies safety criterion

$$dis(N, C_p) \leq d^{safe}$$

Theoretical Results Established

1. A safe CES ***p*** corresponding to a SASSO constraint ensures control safety over an ***unbounded time horizon***.

$$dis(N, C_P) \rightarrow 0 \text{ as } t \rightarrow \infty$$

Theoretical Results Established

1. A safe CES ***p*** corresponding to a SASSO constraint ensures control safety over an ***unbounded time horizon***.

2. There exists an upper bound T_{ub} on the time horizon length for safety verification (i.e., checking $dis(N, C_p) \leq d^{safe}$).

$$T_{ub} = \left\lceil \frac{\ln\left(2\,||\mathcal{A}_{ns}||^{l-1}\,||x^n[0]||\right) - \ln\left(d^{safe}\right)}{|\ln \epsilon|} \right\rceil \times l.$$

Theoretical Results Established

1. A safe CES ***p*** corresponding to a SASSO constraint ensures control safety over an ***unbounded time horizon***.

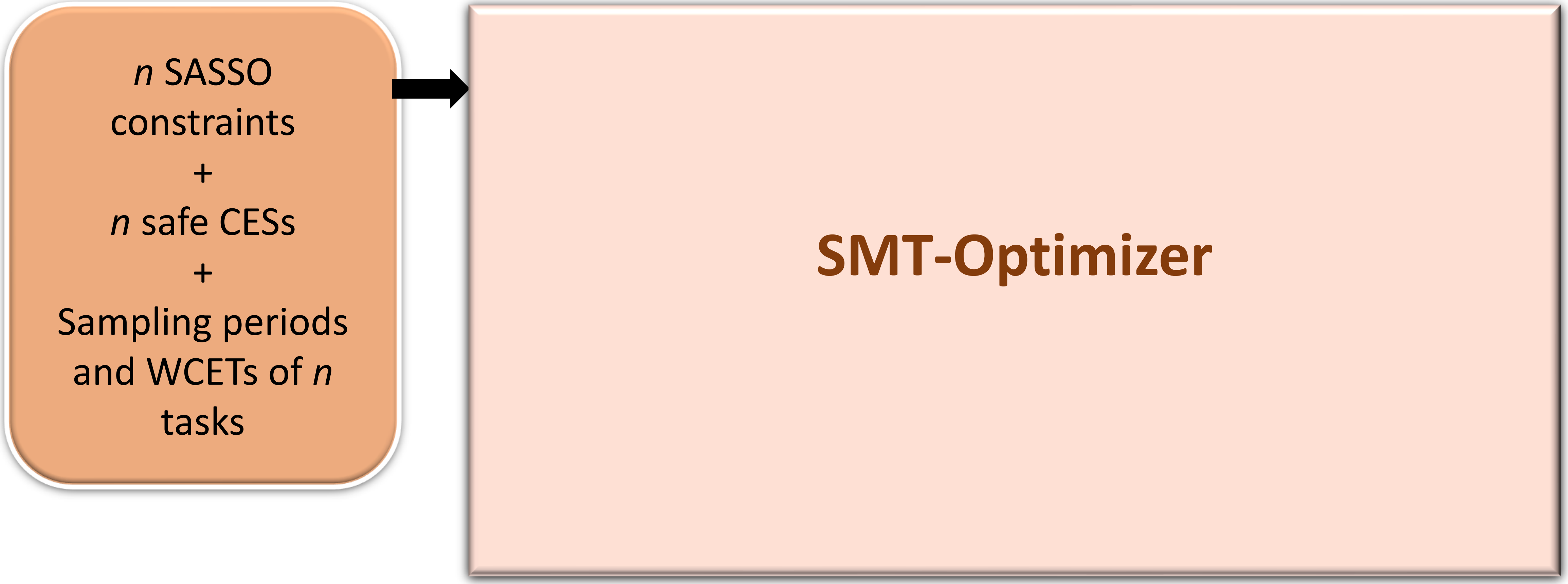
2. There exists an upper bound $\boldsymbol{T_{ub}}$ on the time horizon length for safety verification (i.e., checking $dis(N, C_p) \leq d^{safe}$).

3. Exact time horizon length for safety verification is the ***settling time***, from the time of application of an external disturbance.

Step 3: Schedulability Test

and Synthesizing Schedule

Schedulability Test and Synthesizing Schedule



Schedulability Test and Synthesizing Schedule

n SASSO constraints
+
n safe CESs
+
Sampling periods and WCETs of *n* tasks

→

SMT-Optimizer

- Feasibility related constraints
- Response-time related constraints
- Conflict-removing constraints
- Constraint for minimizing worst-case response time

Schedulability Test and Synthesizing Schedule

n SASSO constraints
+
n safe CESs
+
Sampling periods and WCETs of *n* tasks

SMT-Optimizer

- Feasibility related constraints
- Response-time related constraints
- Conflict-removing constraints
- Constraint for minimizing worst-case response time

SAT

A feasible schedule

Schedulability Test and Synthesizing Schedule

n SASSO constraints
\+
n safe CESs
\+
Sampling periods and WCETs of *n* tasks

SMT-Optimizer

- Feasibility related constraints
- Response-time related constraints
- Conflict-removing constraints
- Constraint for minimizing worst-case response time

UNSAT → **Tasks not schedulable**

SAT → **A feasible schedule**

Summary of the Proposed Method

Control Design Parameter:
Settling Time

Step 1: Ensuring exponential ***stability*** of the system

Set of n SASO constraints

Control Safety Metric:

Deviation between ideal behavior and behavior with deadline misses

Step 2: Ensuring ***safety*** over infinite time horizon

Set of n safe CESs and SASSO constraints

Minimizing the Worst-Case Response Time **(WCRT)**

Step 3: Synthesizing an ***SMT-based***, safe and stable ***schedule*** with ***minimized WCRT***

Evaluation

Sl. No.	Benchmark control systems considered from the automotive domain	Order of the system
1	Resistor-capacitor network (RC)	2
2	DC-motor speed control (DC)	2
3	Vehicle dynamic control (VDC)	2
4	Lane following Controller of an F1 tenth model car (F1)	2
5	Trajectory tracking control (TTC)	2
6	DC-servo control (DCS)	2
7	Cruise control (CC)	3
8	Adaptive cruise control (ACC)	3
9	Suspension control (SC)	4
10	Lane keeping system (LK)	4
11	Vision-based lateral control (LC)	5

Evaluation

2 parameters

Number of plant-control systems
and corresponding control tasks

Processor utilization

Evaluation

2 parameters

Number of plant-control systems and corresponding control tasks

Processor utilization

Tasks $\{T_i\}_{i=1 \ to \ n}$: $T_i = \{(m'_i, k_i), p_i, c_i, h_i\}$

(m'_i, k_i) : SASSO constraint

p_i : Safe CES

c_i : WCET of the control task

h_i : Sampling period of the controller

Util. = $\frac{m'_i \times c_i}{k_i \times h_i}$

Scalability Analysis

2 parameters

Number of plant-control systems and corresponding control tasks

Processor utilization

- *Increase* the number of ***tasks*** and ***job instances***.
- *Increase* the ***processor utilization*** by *increasing* ***WCET***.
- ***Aim:*** ***Report a feasible schedule*** with
  1. A ***fairly reasonable runtime overhead.***
  2. ***Improved*** scope of ***schedulability.***

Scalability Analysis

No of control tasks	No of jobs scheduled	Range of processor utilization for which a schedule is obtained	Time range to synthesize a safe and stable schedule
5	300	0.7 – 0.82	0.08 – 0.126 s **(Less than 2s)**
7	630	0.7 – 1.0	0.55 – 6.058 s **(Less than 7s)**
9	14,490	0.7 – 0.97	0.07 – 3.556 s **(Less than 4s)**
11	18,270	0.7 – 0.98	1.547 – 161.836 s **(Less than 3min)**
13	20,790	0.7 – 0.97	16.167 – 400.053 s **(Less than 7min)**
15	46,620	0.7 – 0.92	485.305 – 8517.031 s **(Less than 2.5hr)**

Scalability Analysis

No of control tasks	No of jobs scheduled	Range of processor utilization for which a schedule is obtained	Time range to synthesize a safe and stable schedule
5	300	0.7 – 0.82	0.08 – 0.126 s **(Less than 2s)**
7	630	0.7 – 1.0	0.55 – 6.058 s **(Less than 7s)**
9	14,490	0.7 – 0.97	0.07 – 3.556 s **(Less than 4s)**
11	18,270	0.7 – 0.98	1.547 – 161.836 s **(Less than 3min)**
13	20,790	0.7 – 0.97	16.167 – 400.053 s **(Less than 7min)**
15	46,620	0.7 – 0.92	485.305 – 8517.031 s **(Less than 2.5hr)**

- For 15 tasks, schedules are synthesized within **7 min-38 min up** to **Util. < 0.85**.
- For **Util. > 0.85**, we obtain schedules but with considering a **time-out of 3 hrs**.

Comparison with the State-of-the-Art

State-of-the-art methods:

1) PGS (Pattern Guided Stable schedule)

- Sumana Ghosh, Souradeep Dutta, Soumyajit Dey and Pallab Dasgupta. 2017. *A Structured Methodology for Pattern Based Adaptive Scheduling in Embedded Control.* ACM Transactions on Embedded Computing Systems (TECS) 16, 5s, 189:1–189:22.

2) DSHT (Deterministic verification of schedule constructed with Statistical Hypothesis Testing)

- Shengjie Xu, Bineet Ghosh, Clara Hobbs, Enrico Fraccaroli, Parasara Sridhar Duggirala and Samarjit Chakraborty. 2023. *Statistical approach to efficient and deterministic schedule synthesis for cyber-physical systems*. In Proc. International Symposium on Automated Technology for Verification and Analysis (ATVA). Springer, 312–333.

3) SCS (Safe Constraint Synthesis to generate safe schedule)

- Shengjie Xu, Bineet Ghosh, Clara Hobbs, P. S. Thiagarajan and Samarjit Chakraborty. 2023. *Safety-aware flexible schedule synthesis for cyber-physical systems using weakly-hard constraints.* In Proc. Asia and South Pacific Design Automation Conference (ASP-DAC). 46–51.

Comparison with the State-of-the-Art

Our method referred to as **FMSS: Formal Methods for synthesizing a Safe and Stable schedule**

1. PGS (constructs a stable schedule)

Comparison with w.r.t. metrics:

- Safety
- Scope of schedulability
- Runtime Overhead

2. DSHT, SCS (construct a safe schedule)

Comparison with w.r.t. metrics:

- Stability
- Runtime Overhead

Comparison with PGS

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
No of control tasks	Util.	Time taken by FMSS	Time taken by PGS stable constraints	Time taken by PGS with SASO constraints
5	0.76	0.100 s	80.625 s	0.456 s
	0.82	0.070 s	207.547 s	1.344 s
7	0.78	0.421 s	10.234 s	0.100 s
	0.82 – 1.0	✓	> 1 hr (timed out)	X
9	0.70	0.030 s	26.219 s	0.077 s
	0.72 – 0.97	✓	> 1 hr (timed out)	X
11	0.70 – 0.98	✓	> 1 hr (timed out)	X
13	0.70 – 0.92	✓	> 1 hr (timed out)	X
15	0.70 – 0.85	✓	> 1 hr (timed out)	X

1. Comparing runtime overhead
2. Comparing scope of schedulability

SASO constraints: part of FMSS

Considered here for fair comparison

Comparison with PGS

Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
No of control tasks	Util.	Time taken by FMSS	Time taken by PGS stable constraints	Time taken by PGS with SASO constraints
5	0.76	0.100 s	80.625 s	0.456 s
	0.82	0.070 s	207.547 s	1.344 s
7	0.78	0.421 s	10.234 s	0.100 s
	0.82 – 1.0	✓	**> 1 hr (timed out)**	**X**
9	0.70	0.030 s	26.219 s	0.077 s
	0.72 – 0.97	✓	**> 1 hr (timed out)**	**X**
11	0.70 – 0.98	✓	**> 1 hr (timed out)**	**X**
13	0.70 – 0.92	✓	**> 1 hr (timed out)**	**X**
15	0.70 – 0.85	✓	**> 1 hr (timed out)**	**X**

Improved runtime overhead in FMSS

Feasible schedule obtained within reasonable time

Suffers from time-out issues

Comparison with PGS

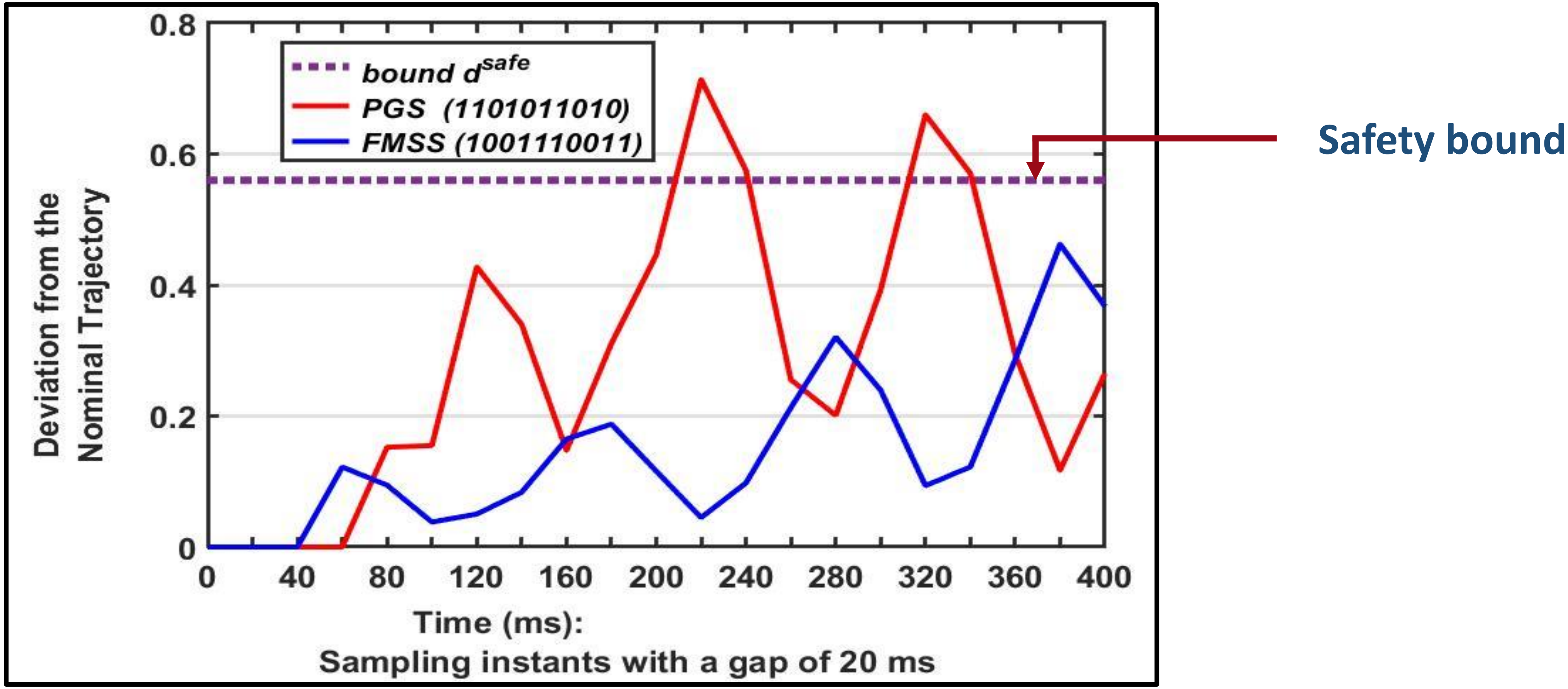
Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
No of control tasks	Util.	Time taken by FMSS	Time taken by PGS stable constraints	Time taken by PGS with SASO constraints
5	0.76	0.100 s	80.625 s	0.456 s
	0.82	0.070 s	207.547 s	1.344 s
7	0.78	0.421 s	10.234 s	0.100 s
	0.82 – 1.0	✓	> 1 hr (timed out)	X
9	0.70	0.030 s	26.219 s	0.077 s
	0.72 – 0.97	✓	> 1 hr (timed out)	X
11	0.70 – 0.98	✓	> 1 hr (timed out)	X
13	0.70 – 0.92	✓	> 1 hr (timed out)	X
15	0.70 – 0.85	✓	> 1 hr (timed out)	X

Improved scope of schedulability in FMSS

Feasible schedule obtained

Tasks not schedulable

Comparison with PGS



Violation of safety bound in PGS in F1-tenth model car system

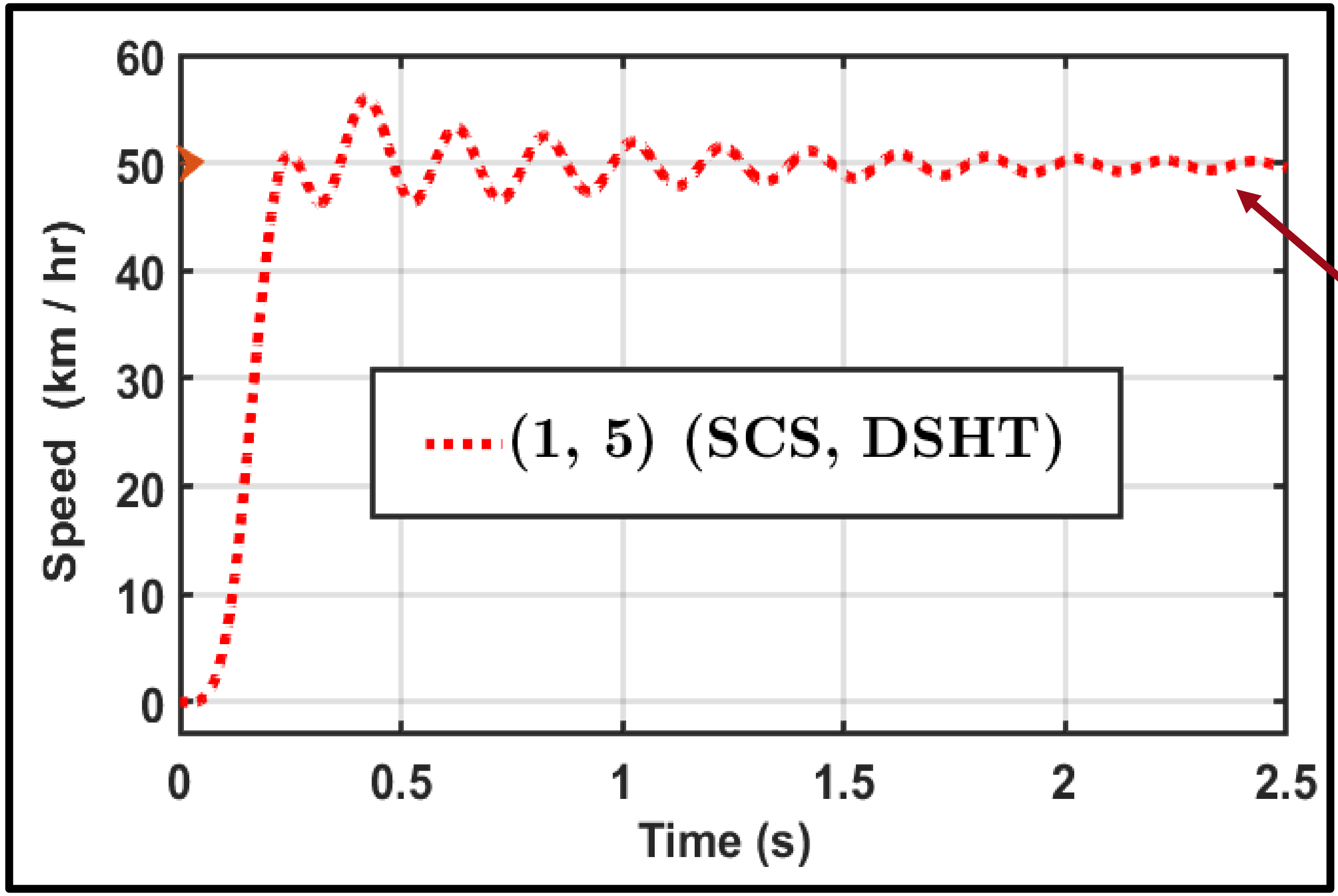
Comparison with DSHT and SCS

- We correlate the $(l, \in)$-exponential stability criterion with the ***settling time***.

Settling time → $(l, \in)$-exponential stability criterion → stable constraint → SASSO constraint → ***safe CES***.

- A ***shorter settling time*** signifies a ***stable*** and a ***more responsive*** system.

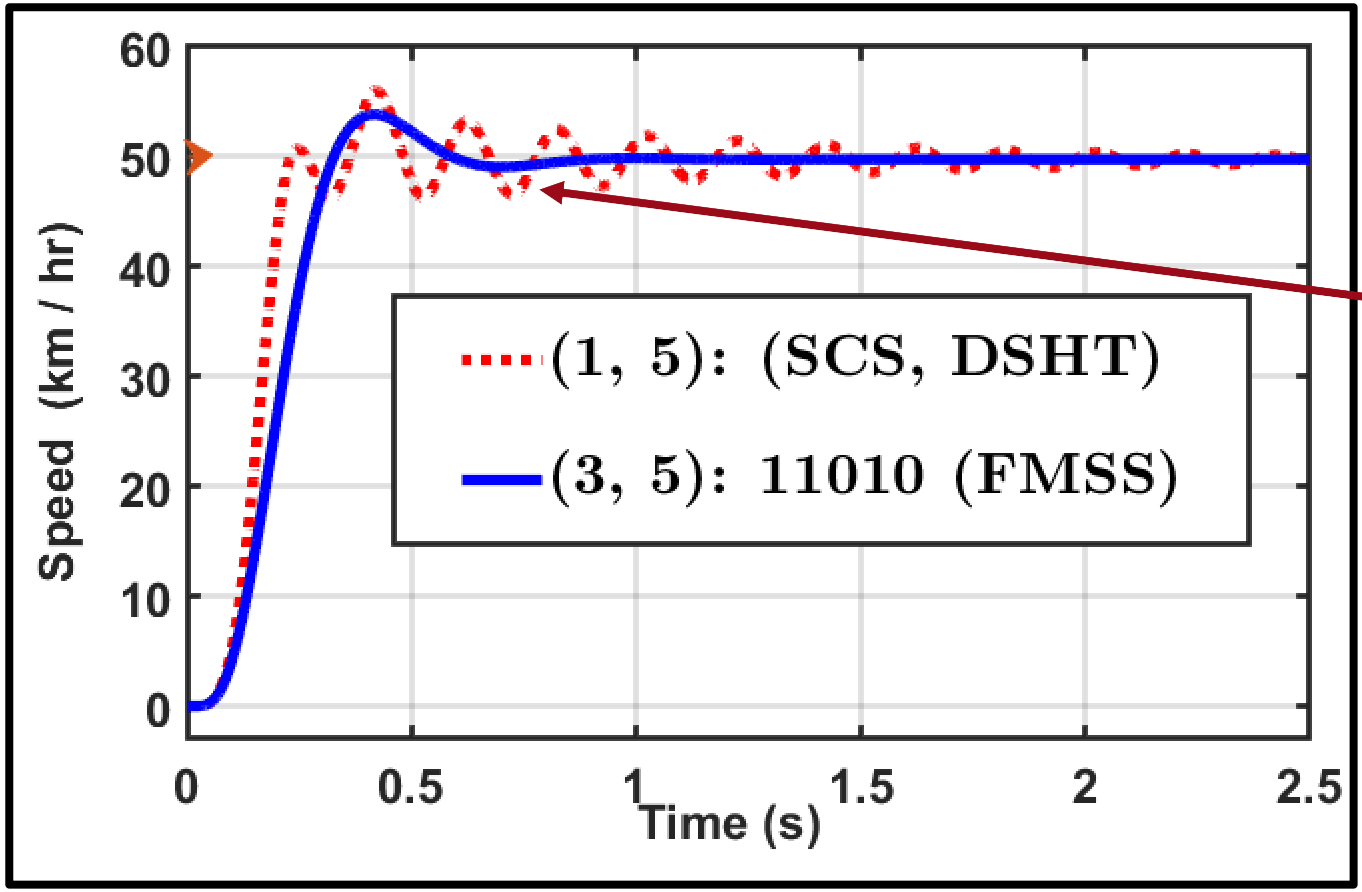
Comparison with DSHT and SCS



Settling time is
> 2.5 s (4s) in SCS, DSHT

Cruise control system: (1, 5) is a safe constraint reported by SCS and DSHT

Comparison with DSHT and SCS

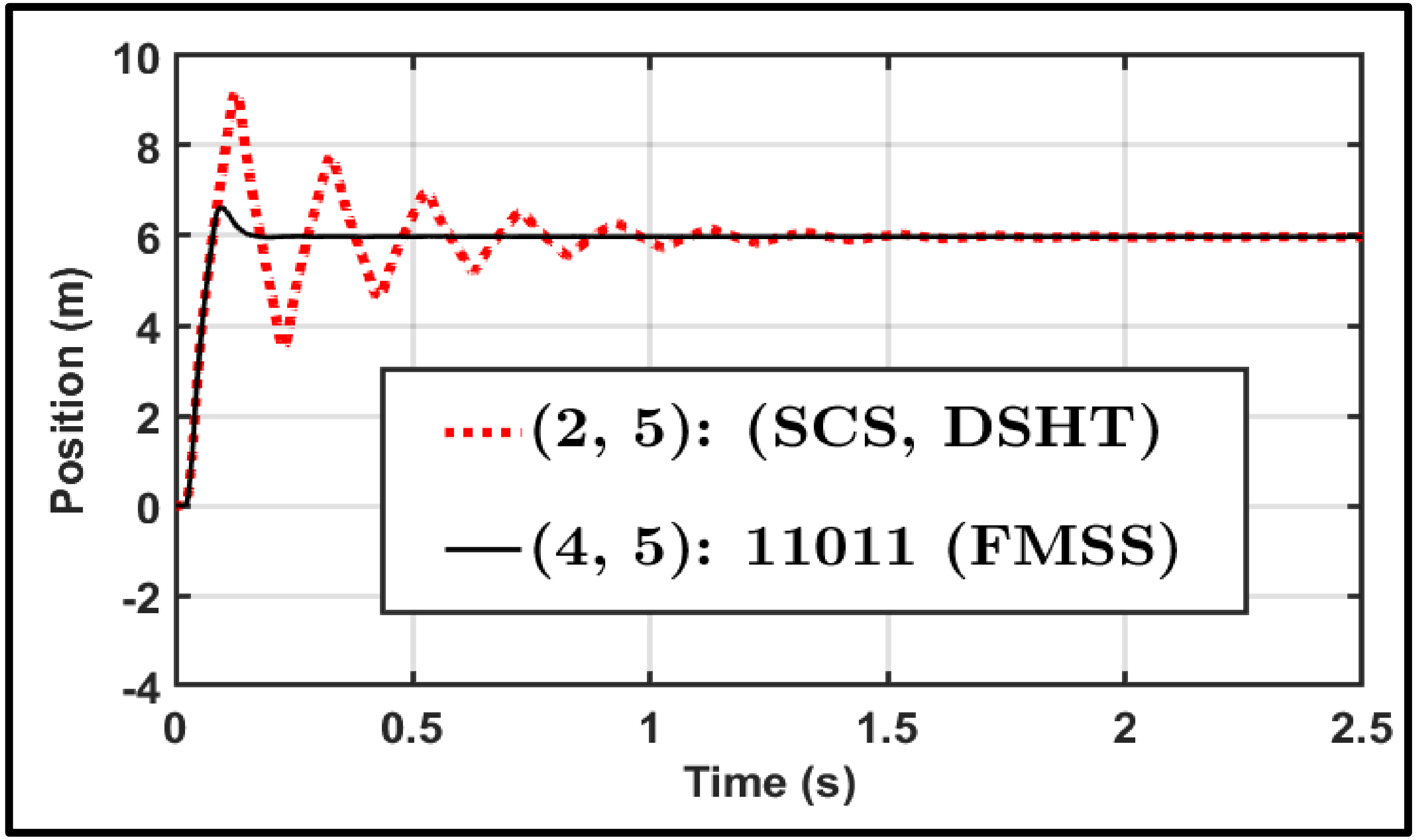


Settling time is 0.8s in FMSS

80% improvement in FMSS

Cruise control system: (3, 5) is a SASSO constraint, 11010 is a safe CES, reported by FMSS

Comparison with DSHT and SCS



Settling time:
2.5s (DSHT, SCS)

Settling time:
0.2s (FMSS)

92% improvement in FMSS

Suspension control system: (2, 5) is safe constraint reported by DSHT and SCS

(4, 5) is a SASSO constraint, 11011 is a safe CES, reported by FMSS

Comparison with DSHT and SCS

Comparing the runtime overhead

No. of control tasks	Time taken (FMSS)	Time taken (DSHT)	Time taken (SCS)
5	**0.08 s**	13.50 s	748.37 s (12.47 min)

Runtime is quite high for a small task setup

Comparison with DSHT and SCS

Comparing the runtime overhead

No. of control tasks	Time taken (FMSS)	Time taken (DSHT)	Time taken (SCS)
5	0.08 s	13.50 s	748.37 s

DSHT and SCS: employ **time-consuming** iterative approaches, probabilistic methods, reachability analysis

An exponential time rise possible for a higher number of tasks in DSHT and SCS !!

Comparative Comments on Other Works

Method in [1]:

- Constructs an SMT-based schedule.
- Ensures control safety in a weakly hard scenario.
- But the method is limited to medium-sized systems.

[1]: Anand Yeolekar, Ravindra Metta, and Samarjit Chakraborty. 2024. ***SMT-based Control Safety Property Checking in Cyber-Physical Systems under Timing Uncertainties***. In Proc. on VLSI Design and Embedded Systems (VLSID). 276–280.

Comparative Comments on Other Works

Comparing the runtime overhead

No. of control tasks	Time taken (FMSS)	Time taken [1]	Memory Consumed (FMSS)	Memory Consumed [1]
4	0.04 s	40 s	20 MB	48 MB

SMT-based scheduling in FMSS is **time-** and **compute-efficient**: pruned search space of the SMT-solver with SASO and SASSO constraints

Runtime and memory consumption is quite high for a small task setup

[1]: Anand Yeolekar, Ravindra Metta, and Samarjit Chakraborty. 2024. ***SMT-based Control Safety Property Checking in Cyber-Physical Systems under Timing Uncertainties***. In Proc. on VLSI Design and Embedded Systems (VLSID). 276–280.

Conclusion

Contributions of this paper:

- ✓ ***Triplet (stability, safety, schedulability):*** Explored for the *first* time in the literature.
- ✓ ***Stability and safety:*** Ensured over an *infinite* horizon.
- ✓ ***Proposed scheduling approach:*** Minimized WCRT offers *improved scope of schedulability*.
- ✓ ***Proposed SMT-based approach:*** Time-efficient, hence *increases scalability*.
- ✓ ***Experimental comparison:*** Proposed method *outperforms the state-of-the-art* methods.

Conclusion

Contributions of this paper:

- ✓ ***Triplet (stability, safety, schedulability):*** Explored for the *first* time in the literature.
- ✓ ***Stability and safety:*** Ensured over an *infinite* horizon.
- ✓ ***Proposed scheduling approach:*** Minimized WCRT offers *improved scope of schedulability*.
- ✓ ***Proposed SMT-based approach:*** Time-efficient, hence *increases scalability*.
- ✓ ***Experimental comparison:*** Proposed method *outperforms the state-of-the-art* methods.

Future works:

- Stability and safety for scheduling in *non-linear control systems.*
- *Dependencies in the task model* while designing a safe and stable schedule.

THANK YOU !