SMILEY

A Mixed Criticality Real-time Scheduler for Multicore Systems

Alen Sabu, Biju Raveendran, Rituparna Ghosh

BITS-Pilani, K.K. Birla Goa Campus, India

System in Consideration

How can we integrate tasks with different levels of *criticality* into a common system?

Mixed-Criticality Systems

Tasks in Avionics

DO-178B Software Level	Failure Condition
A	Catastrophic
B	Hazardous
C	Major
D	Minor
E	No effect

Life-critical

Mission-critical

Non-critical

Severity

Criticality

Tolerance to deadline miss

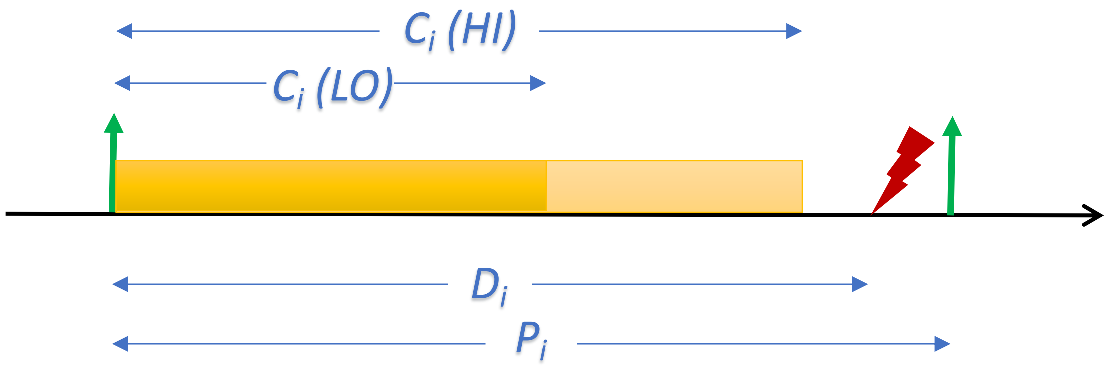
Criticality - Level of required assurance against failure

Requirements: *Timing, Security, Safety*

How can we integrate tasks with different levels of *criticality* into a common system?

- Assign each task a criticality level
- Estimate task parameters according to the requirements of each level
- System starts at the lowest criticality level
- Make sure a higher criticality task can meet the guarantees of the next highest level if it fails on a low level

Mixed-Criticality Task Model



- Program is a set $\boldsymbol{\tau}$ of tasks $\boldsymbol{\tau_i}$ (i = 0, 1, 2,…)
  - Task's minimum inter-release time $\mathbf{P_i}$
  - Task's relative deadline $\mathbf{D_i}$
  - Task's level of criticality $\boldsymbol{\chi_i} \in$ {life, mission, non-critical}
  - Task's computation time $\mathbf{C_{ij}}$ (j = 1, 2,…, χ) predicted for each level

How to schedule mixed-criticality tasks?

- Prioritize the deadline of high criticality tasks
- Possibly at the expense of lower criticality tasks
- Task suspension may occur during the scheduling of the system

Observations

High unproductive time

Discarding low criticality jobs in high criticality mode

Cascading failures

Existing mixed-criticality schedulers

Key Idea

- Best utilize the slack generated in high criticality mode
- Low criticality jobs are scheduled only if they can finish their execution without deadline miss

We consider tasks with only two criticality levels:

1. High criticality (HI): tasks that tolerate no deadline miss
2. Low criticality (LO): tasks that tolerate occasional deadline miss

Goal:
To maximize the multicore processor utilization by executing maximum number of low criticality jobs to completion without missing any HI criticality job's deadline.

An Example

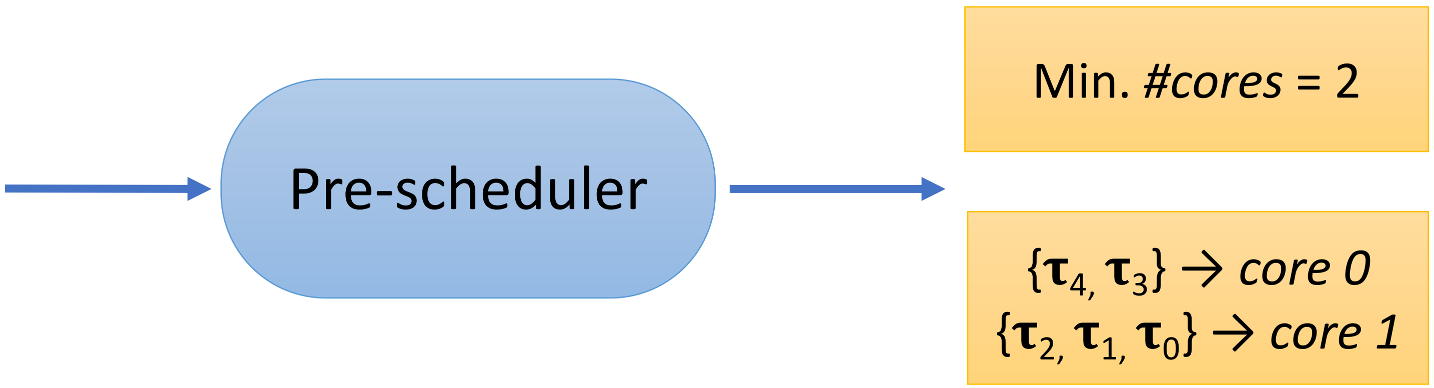
Consider a task set $\boldsymbol{\tau}$

Task	Criticality	Period	WCET(LO)	WCET(HI)	Deadline
τ_0	HI	10	1	3	10
τ_1	HI	10	2	5	10
τ_2	HI	15	2	3	15
τ_3	HI	15	4	6	15
τ_4	HI	30	5	10	30
τ_5	LO	10	3	3	10
τ_6	LO	10	2	2	10
τ_7	LO	15	4	4	15

SMILEY Stage 1 – Pre-scheduler (offline)

- Assigns HI tasks to cores based on first-fit decreasing (period) bin-packing

Task	Period	WCET (LO)	WCET (HI)	Deadline
τ_0	10	1	3	10
τ_1	10	2	5	10
τ_2	15	2	3	15
τ_3	15	4	6	15
τ_4	30	5	10	30



SMILEY Stage 2 – SlackFinder

For a LO criticality job J_{LO} with deadline D_{LO} at time currTime ($\leq D_{LO}$),

- **S_1** — Set of jobs that are already present in local ready queue
- **S_2** — Set of jobs that will arrive between *currTime* and D_{LO}

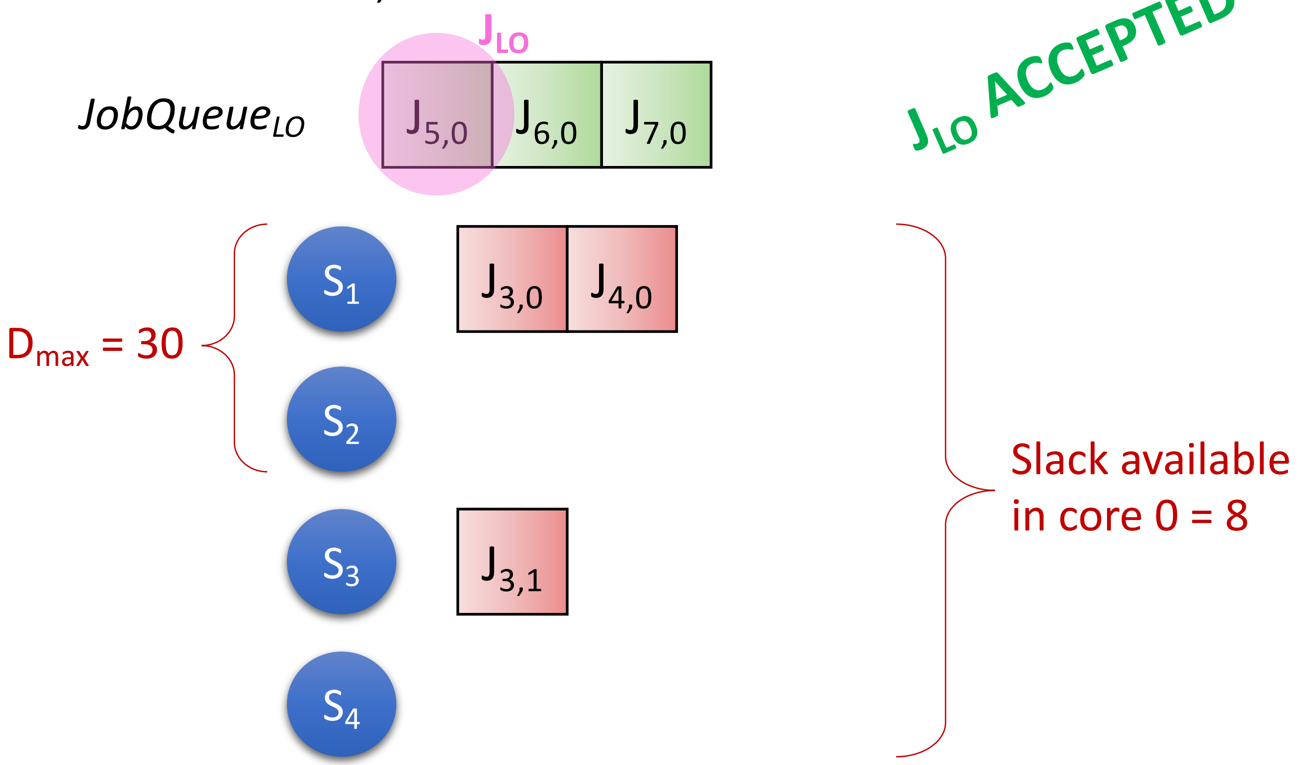
D_{max} *: Maximum of the deadlines of all jobs present in S_1 and S_2*

- **S_3** — Set of jobs that will arrive after D_{LO} with deadlines $\leq D_{max}$
- **S_4** — Set of jobs that will arrive between D_{LO} and D_{max} with deadlines $> D_{max}$

SMILEY Stage 2 – SlackFinder

core 0 - Accepted HI Tasks {τ_4, τ_3}

At *currTime* = 0,

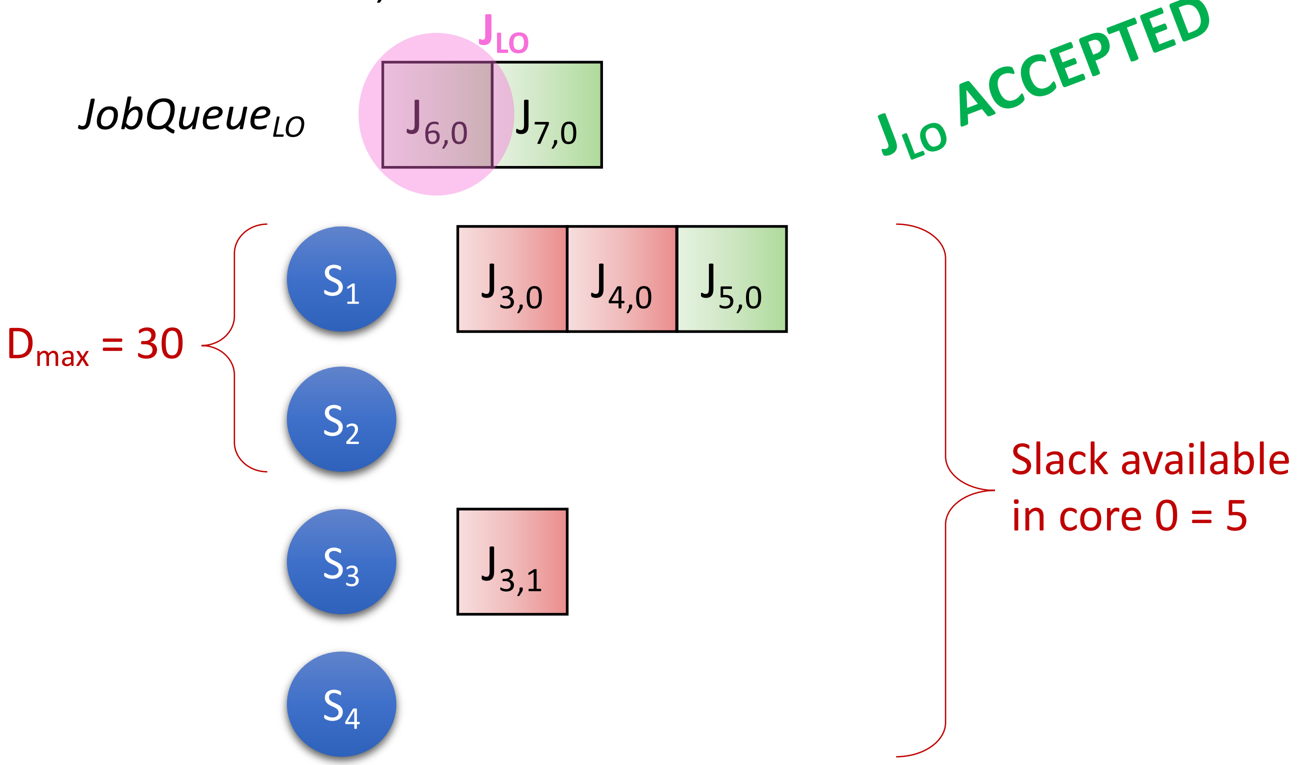


Task	Period	WCET(LO)	WCET(HI)	Deadline
τ_0	10	1	3	10
τ_1	10	2	5	10
τ_2	15	2	3	15
τ_3	15	4	6	15
τ_4	30	5	10	30
τ_5	10	3	3	10
τ_6	10	2	2	10
τ_7	15	4	4	15

SMILEY Stage 2 – SlackFinder

core 0 - Accepted HI Tasks {τ_4, τ_3}

At *currTime* = 0,

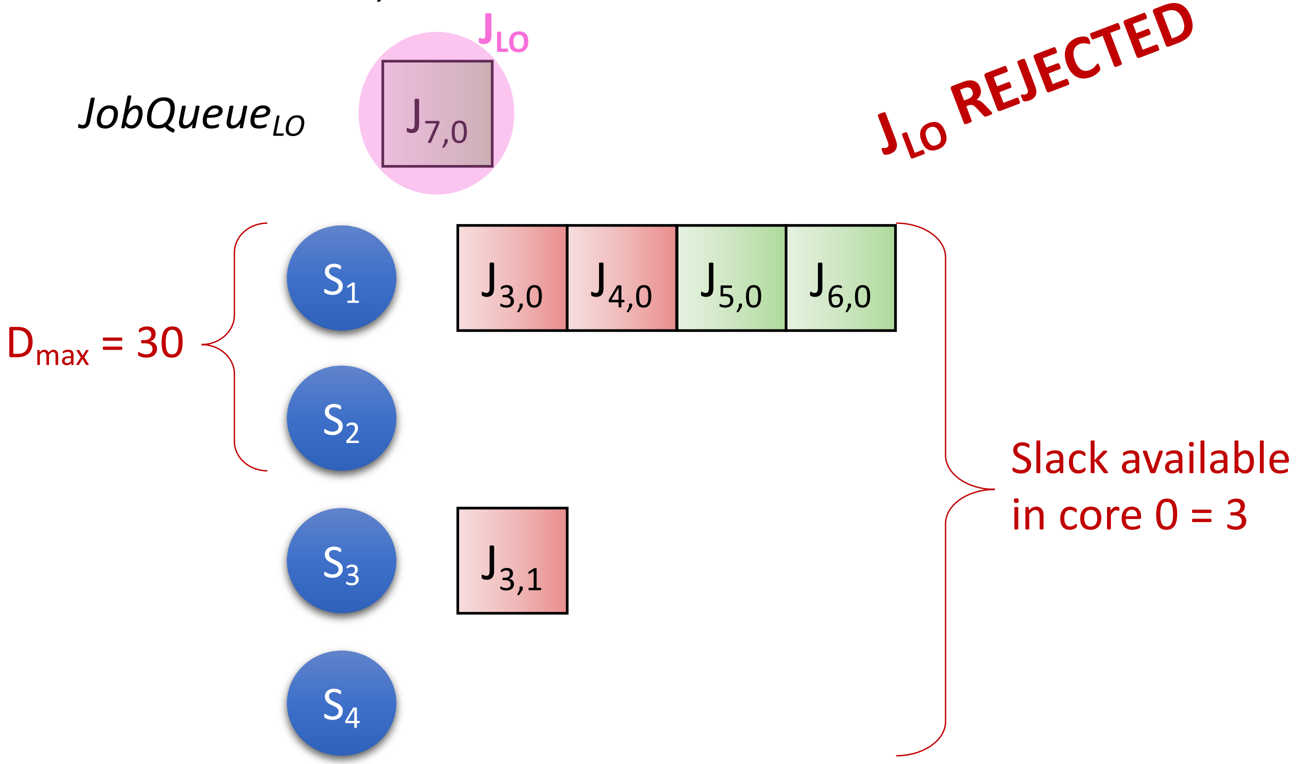


Task	Period	WCET(LO)	WCET(HI)	Deadline
τ_0	10	1	3	10
τ_1	10	2	5	10
τ_2	15	2	3	15
τ_3	15	4	6	15
τ_4	30	5	10	30
τ_5	10	3	3	10
τ_6	10	2	2	10
τ_7	15	4	4	15

SMILEY Stage 2 – SlackFinder

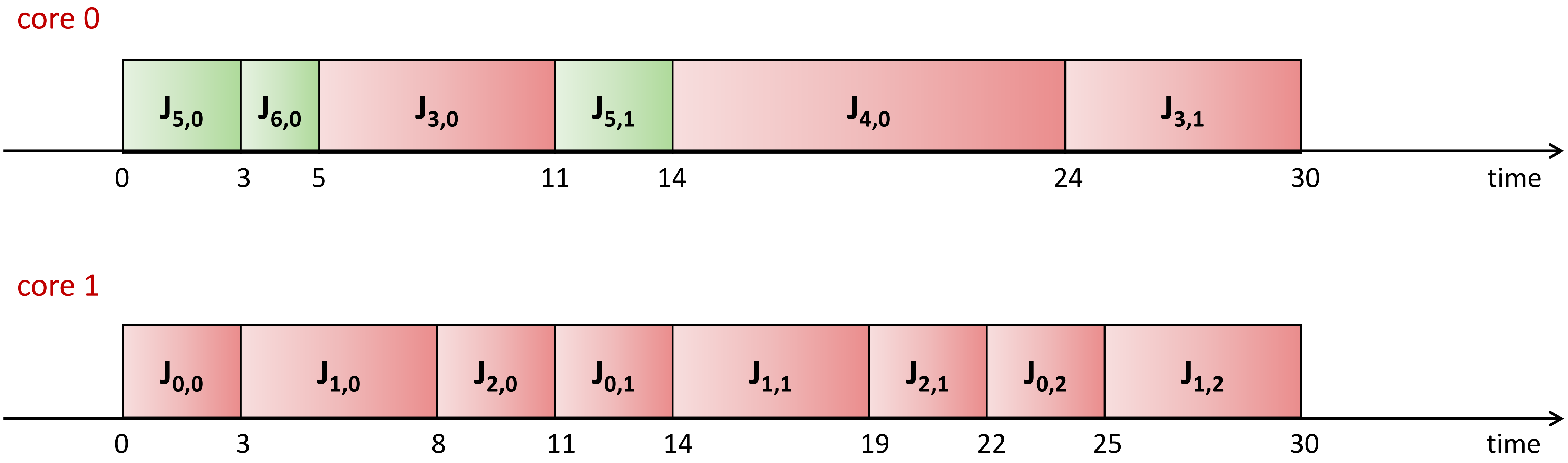
core 0 - Accepted HI Tasks {τ_4, τ_3}

At *currTime* = 0,



Task	Period	WCET(LO)	WCET(HI)	Deadline
τ_0	10	1	3	10
τ_1	10	2	5	10
τ_2	15	2	3	15
τ_3	15	4	6	15
τ_4	30	5	10	30
τ_5	10	3	3	10
τ_6	10	2	2	10
τ_7	15	4	4	15

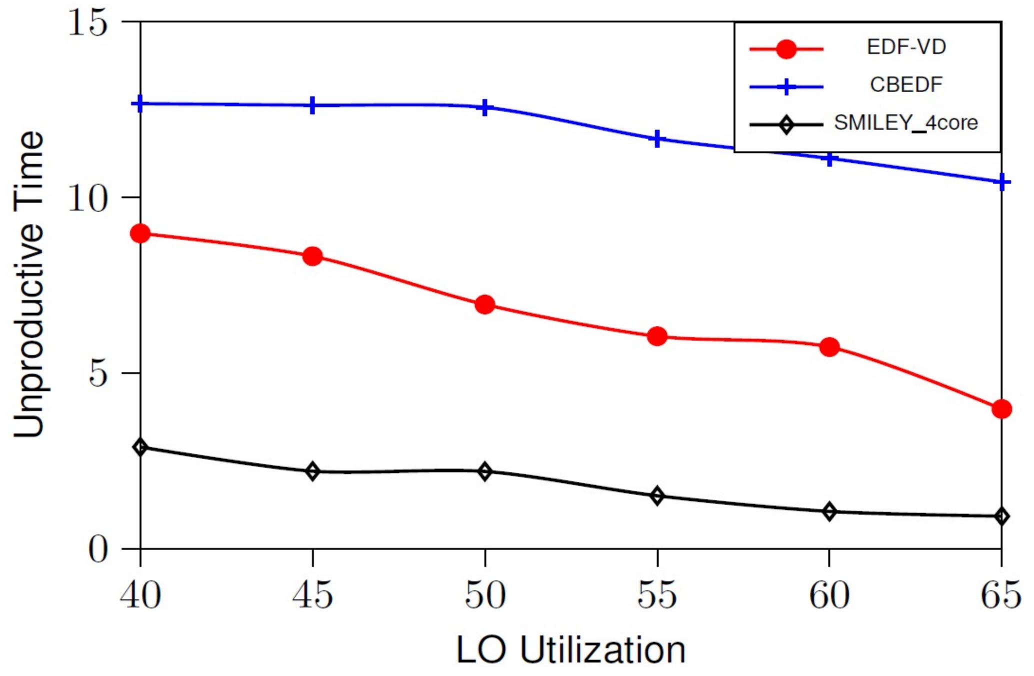
SMILEY Stage 3 – Runtime scheduler



Evaluation Unproductive Time

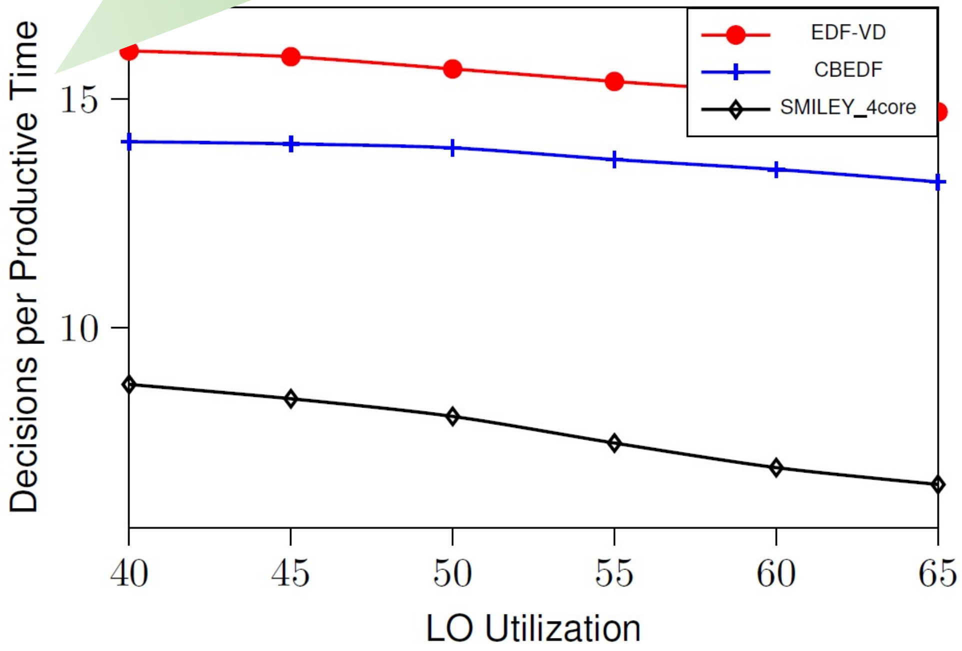
$$Unproductive\ Time = \frac{Hyperperiod - Total\ Productive\ Time}{Time\ available\ for\ LO\ execution}$$

73.9% and 85.2% of saving in Unproductive Time in comparison with EDF-VD and CBEDF respectively.

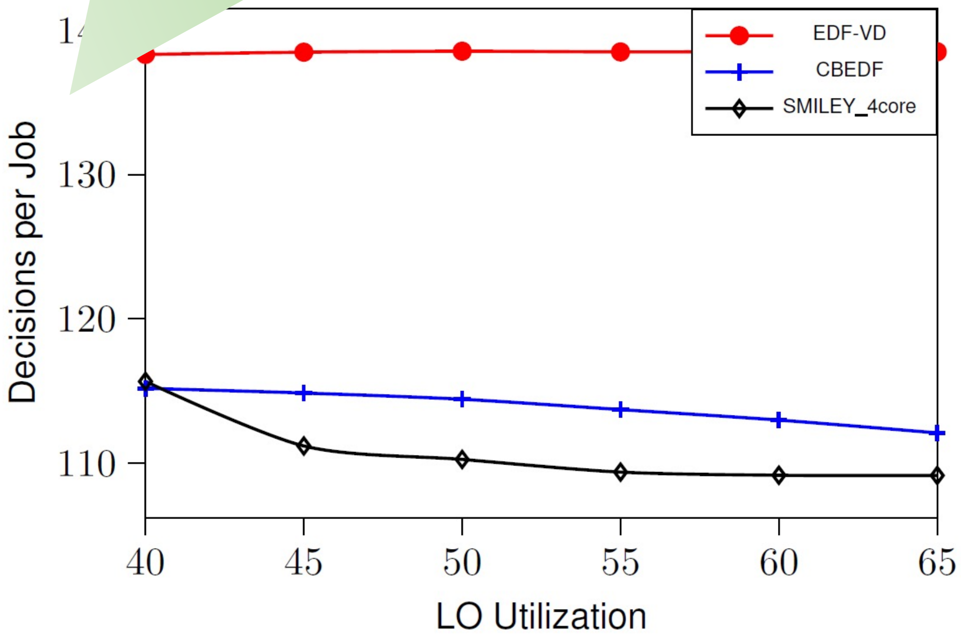


Evaluation Decision points

50.3% and 43.9% lesser values when compared to EDF-VD and CBEDF respectively



20% and 2.7% lesser values when compared with EDF-VD and CBEDF respectively



Conclusion

- This work proposed SMILEY, a mixed-criticality scheduling algorithm for multicore systems
- The results show that SMILEY outperform widely used mixed-criticality scheduling algorithms like EDF-VD and CBEDF
- SMILEY tries to include maximum number of LO criticality jobs and maximizes the productive time

SMILEY

A Mixed Criticality Real-time Scheduler for Multicore Systems

Alen Sabu, Biju Raveendran, Rituparna Ghosh

BITS-Pilani, K.K. Birla Goa Campus, India