Schedulability Analysis of Mixed-Criticality Systems on Multiprocessors

Risat Mahmud Pathan

Chalmers University of Technology, Sweden

Mixed Criticality (MC) Systems

- Mixed-criticality system has multiple criticalities
  - **Integration** needs certification
  - Certification is about **assurance**

    higher criticality=>higher assurance in meeting deadlines

- **Different WCETs** of the same task [Vetsal,RTSS07]
  - Higher assurance => larger WCET

- Meeting **deadline depends** on the WCET
  - different assurance in meeting deadlines

Problem Statement

- Conventional scheduling policy cannot address task with multiple WCETs
  - ➢ **How to ensure that all the deadlines are met on multiprocessors?**
    - ✓ many WCETs with many assurance levels
    - ✓ the system needs certification
- The optimal fixed-priority ordering for multiprocessors is **not known**
  - ➢ **How to assign the fixed-priorities to the tasks for scheduling MC systems on multiprocessors?**

Outline

- MC task model
  - Criticality behaviors and certification
- Scheduling algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

MC Task Model

- Total *n* sporadic tasks
  - Task $\tau_i \equiv (L_i, C_i, D_i, T_i)$
    - dual-criticality $\mathbf{L_i \in \{LO, HI\}}$
    - $C_i = \langle C_i^{LO}, C_i^{HI} \rangle$ where $\mathbf{C_i^{LO} \leq C_i^{HI}}$
    - relative deadline ≤ period, i.e., $\mathbf{D_i \leq T_i}$

MC Task Model

- Tasks are given **fixed priorities**, and
  - scheduled on **m identical processors**

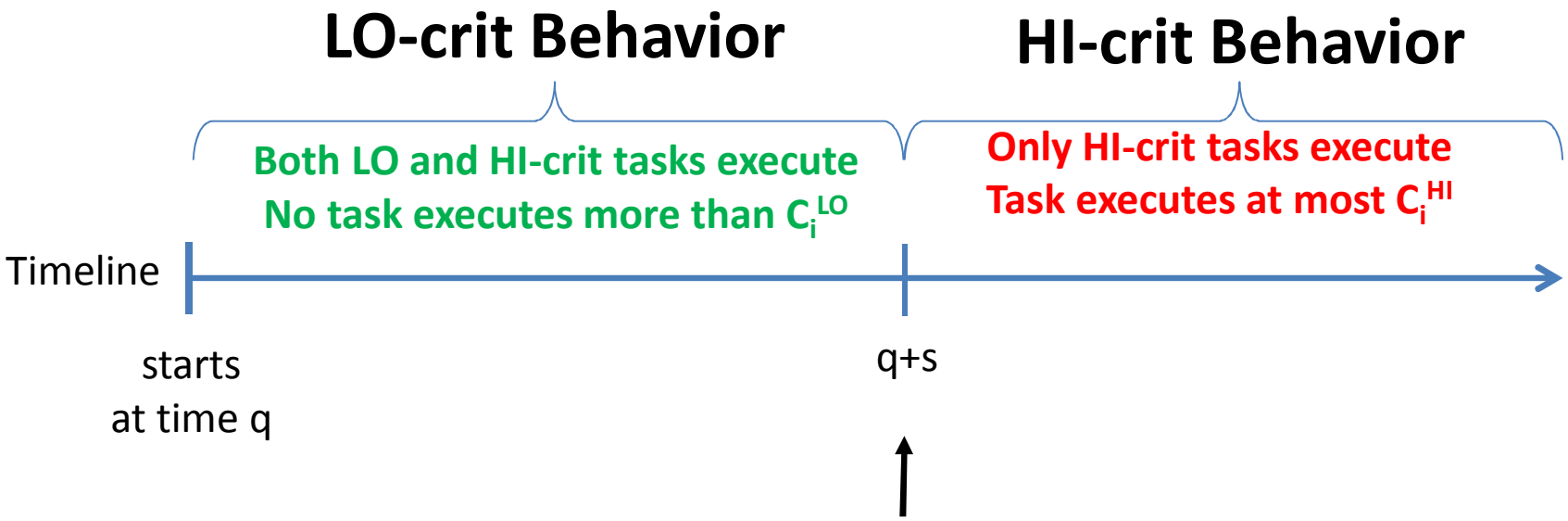
- **hp(i)**: the set of higher priority tasks of task τ_i
  - Higher-Priority and LO-Criticality (hpLc)
  - Higher-priority and HI-Criticality (hpHc)

$$\mathbf{hp(i) = hpLC(i) \cup hpHc(i)}$$

Outline

- MC task model
  - Criticality Behaviors and Certification
- Scheduling algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

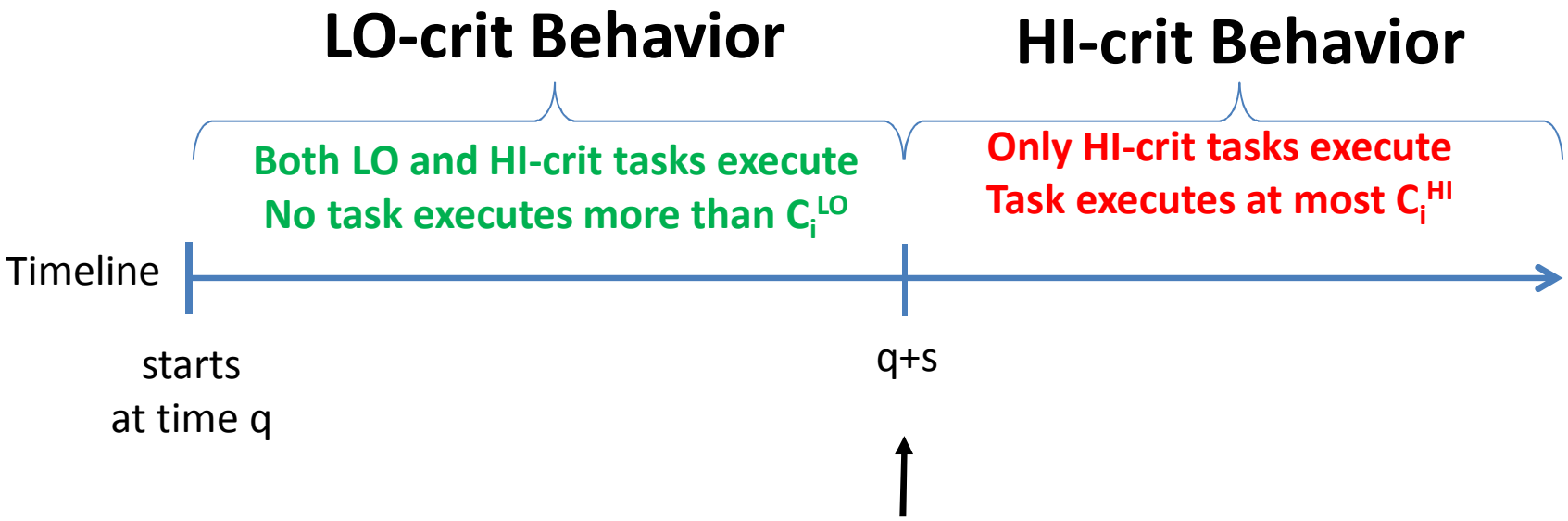
Criticality Behavior and Certification



Some task does not signal completion after executing for C^{LO} time units

Criticality behavior switches from LO to HI at (t+s)

Criticality Behavior and Certification



Some task does not signal completion after executing for C^{LO} time units

After the criticality-switch, additional (C^{HI}-C^{LO}) time units are to be executed

Outline

- MC task model
  - Criticality Behaviors and Certification
- Scheduling algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

MC Scheduling on Multiprocessor (MSM) algorithm

MSM scheduling is same as global FP scheduling

+ criticality-switch detection

+ drop all LO-crit tasks **after** switching

Outline

- MC task model
  - Criticality Behaviors and Certification
- Scheduling algorithm
  - Schedulability analysis and test
- Evaluation
- Conclusion

Schedulability Analysis

Response-time analysis for LO- and HI-crit behaviors to find

$\mathbf{R_i^{LO}}$: Response-time at LO-crit behavior

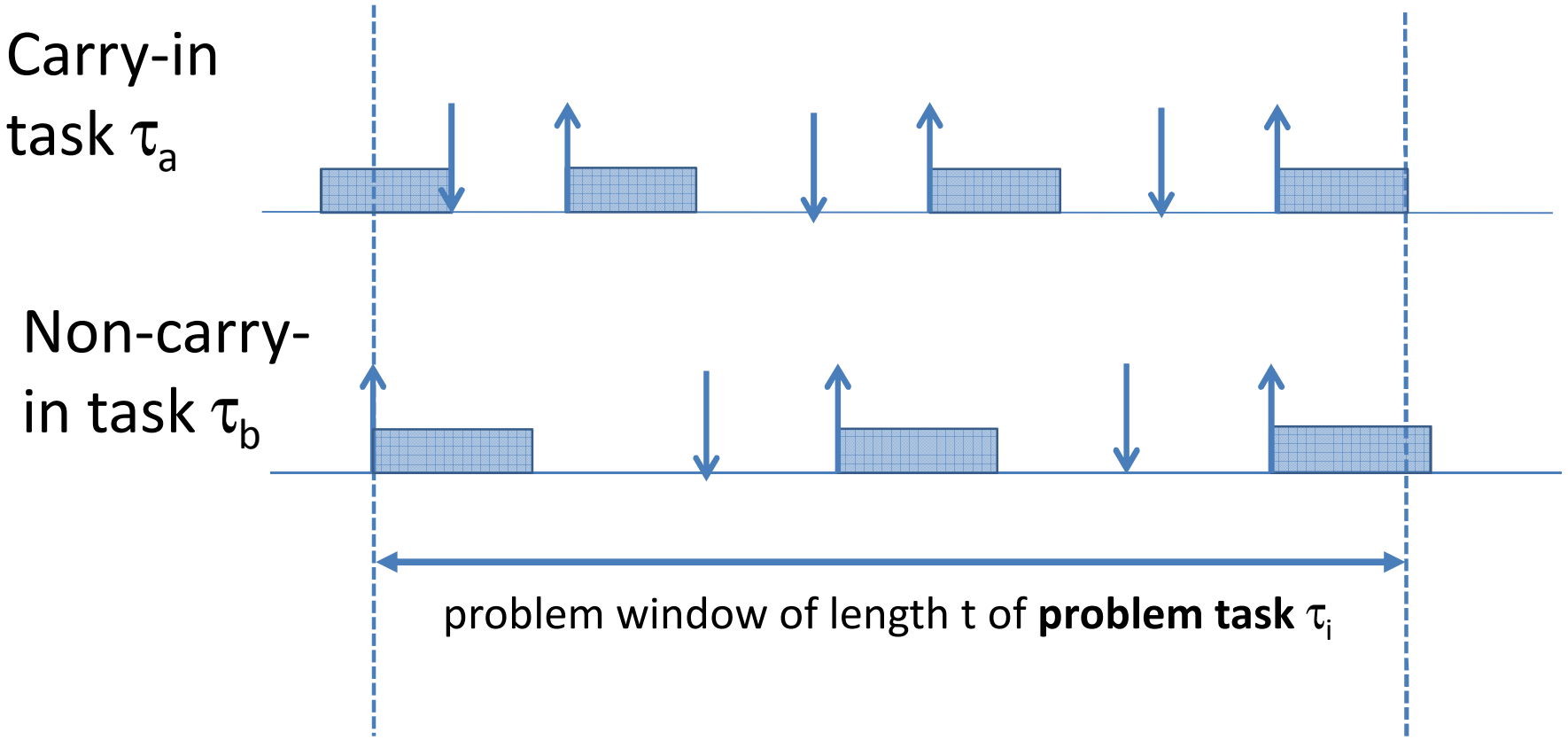
$\mathbf{R_i^{HI}}$: Response-time at HI-crit behavior

Response-Time Analysis Framework

- Consider **problem task τ_i and problem window of length t**
- A job of a problem task τ_i is released at the beginning of the problem window
- To analyze the schedulability of task τ_i,we have to find
  - **Workload** of each task in $\tau_k \in$ hp(i)
    - Carry-in (CI), non-carry-in (NC) workload
  - **Interference**
    - all processors are busy with high priority tasks

Carry-in and non-carry-in workload

[Baruah 2007, Guan et. al 2009, Davis and Burns 2011]



↑ **Release Time**

↓ **Deadline**

Task τ_i is the problem task

τ_a , $\tau_b \in$ hp(i)

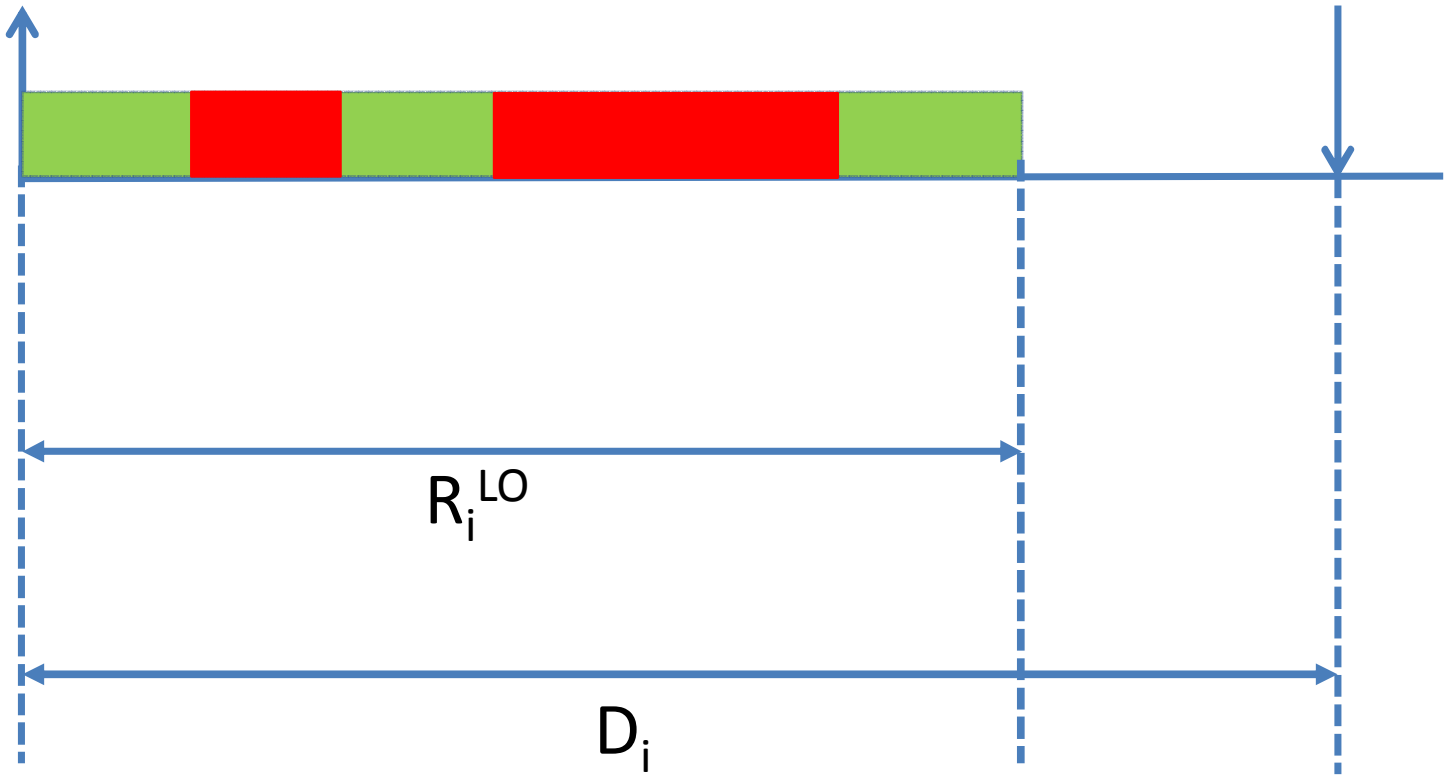
Finding R_i^{LO}

Upper bound on R_i^{LO} where L_i=HI

$\xi_i = D_i - (C_i^{HI} - C_i^{LO})$

Interference

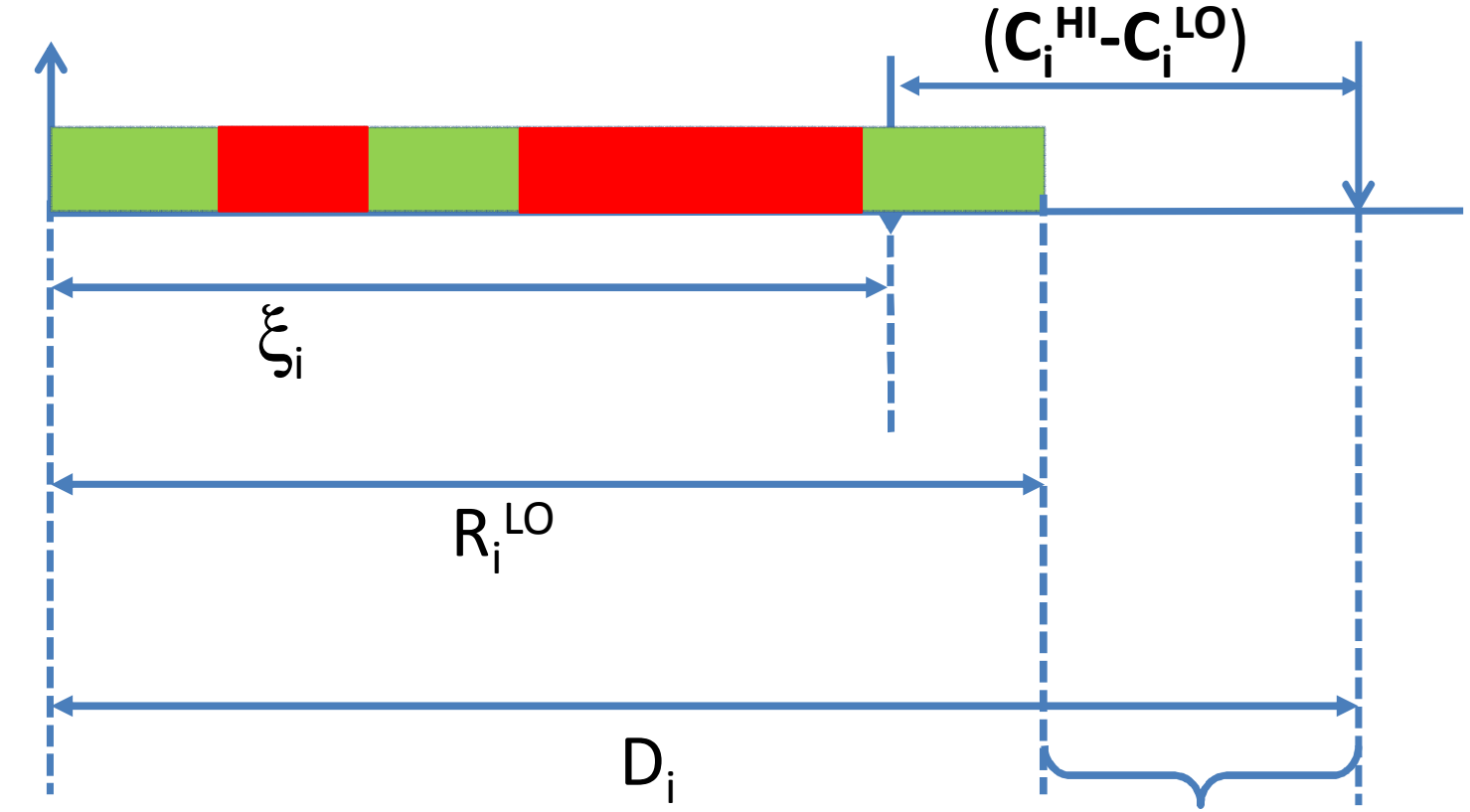
Task τ_i is executing for C_i^{LO}



Task τ_i has executed for C_i^{LO} time units

Upper bound on R_i^{LO} where L_i=HI

$\xi_i = D_i - (C_i^{HI} - C_i^{LO})$



Assume

$R_i^{LO} \geq \xi_i$

Less than $(C_i^{HI} - C_i^{LO})$ time units, failure at HI-crit

New Task Model for LO-Criticality Behavior

- The **true relative deadline** of task τ_i is

$$\xi_i = D_i - (C_i^{HI} - C_i^{LO}) \quad \text{if } L_i = HI$$

$$\xi_i = D_i \quad \text{if } L_i = LO$$

During LO-criticality behavior, MSM is global FP scheduling with (C^{LO}, D=ξ, T)

The Response Time Test

- Analyze traditional **global FP scheduling** of taskset given with parameters (C^{LO}, ξ, T)

  [Bertogna et. al. 2007, Guan et. al 2009, Davis et. al. 2011]

- **Interference: $I_i(t)$**

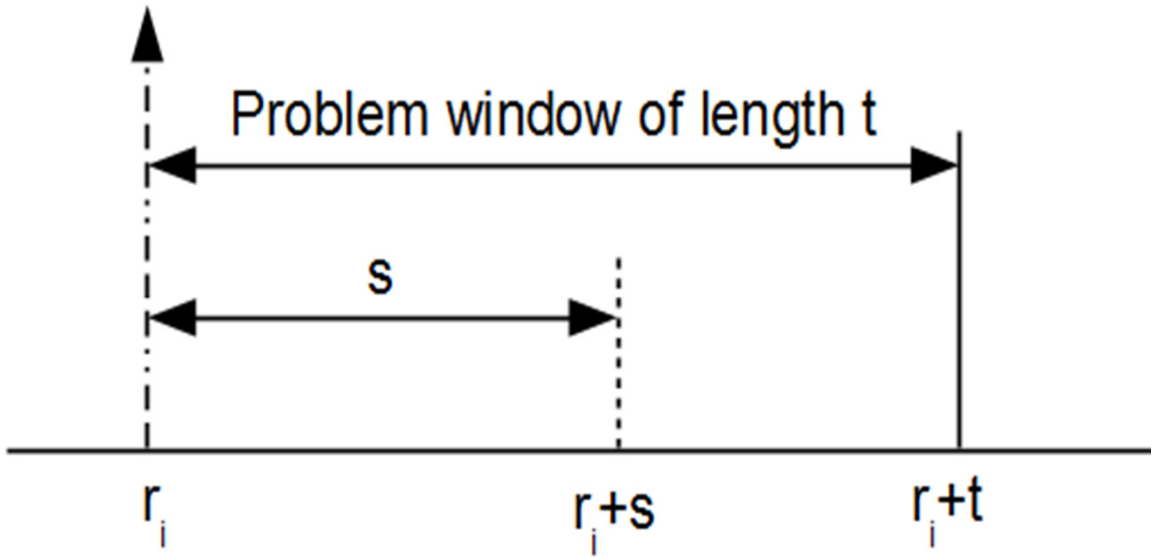
- R_i^{LO} is the solution of $t \leftarrow C_i^{LO} + I_i(t)$

Once workload is known, finding the response time is same as in Guan et al. RTSS2009

Finding R_i^{HI}

Problem window and higher priority tasks

r_i is the release time of a job of problem task
s is the relative distance for the criticality switch



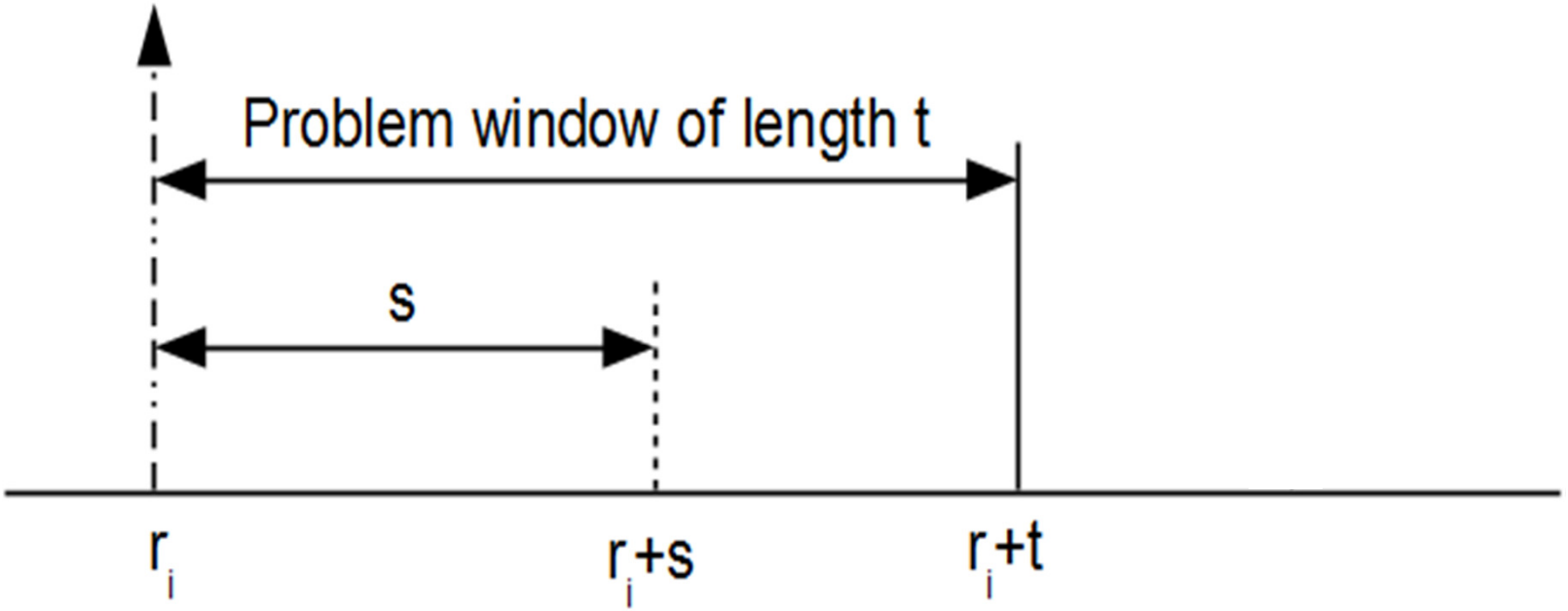
hp(i) = hpLc(i) ∪ **hpHc(i)**

To find interference, we have to find

Non-Carry-in and Carry-in workload of each task in **hpLc(i)** and **hpHc(i)**

Non-Carry-in and Carry-in workload of each task in **hpLc(i)**

Workload for each task in hpLc(i)

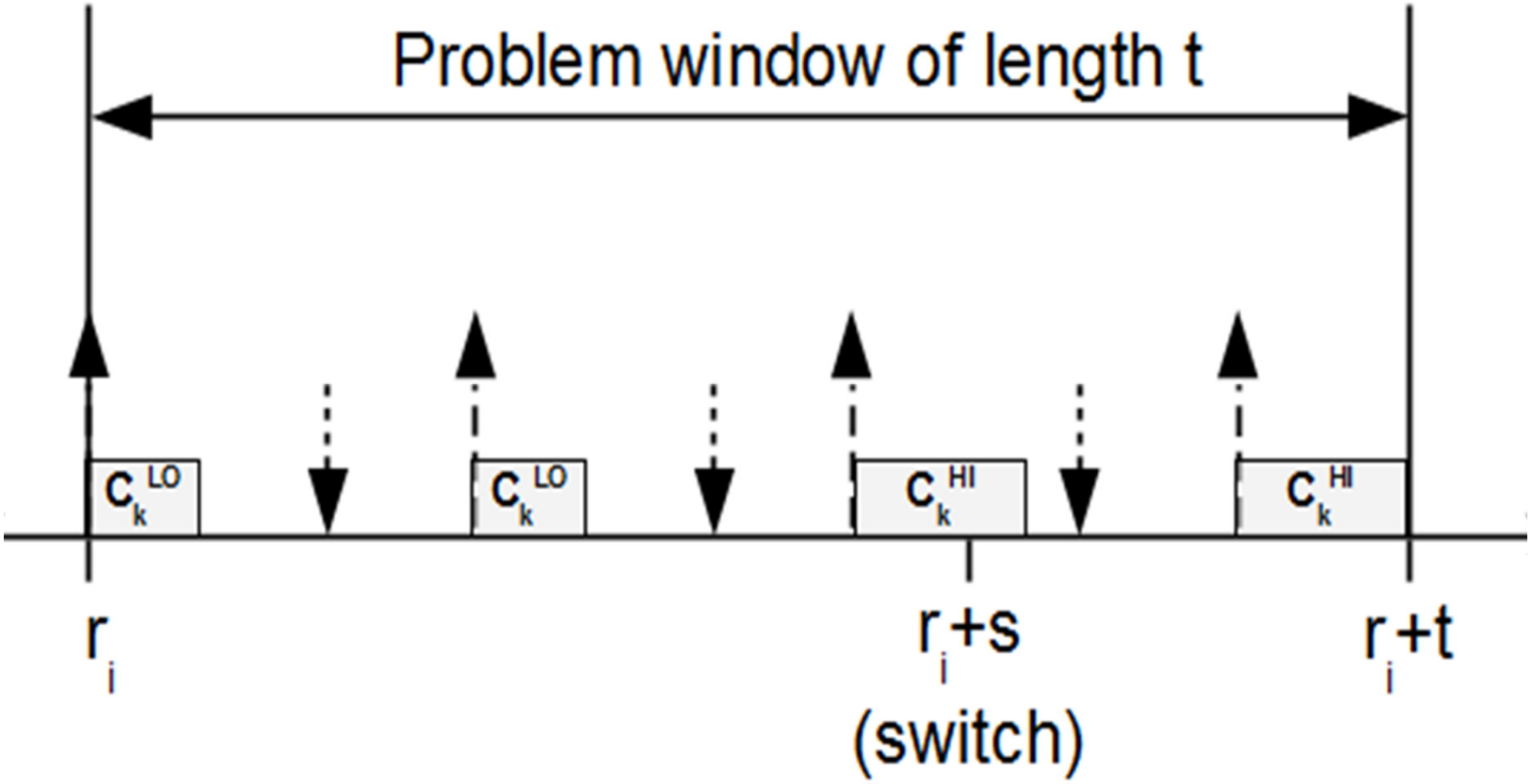


Same as the carry-in and non-carry-in workload calculation used in LO-crit behavior

i.e., we find workload of tasks in hpLc(i) with parameters (C^{LO},D=ξ, T) within an interval of length s

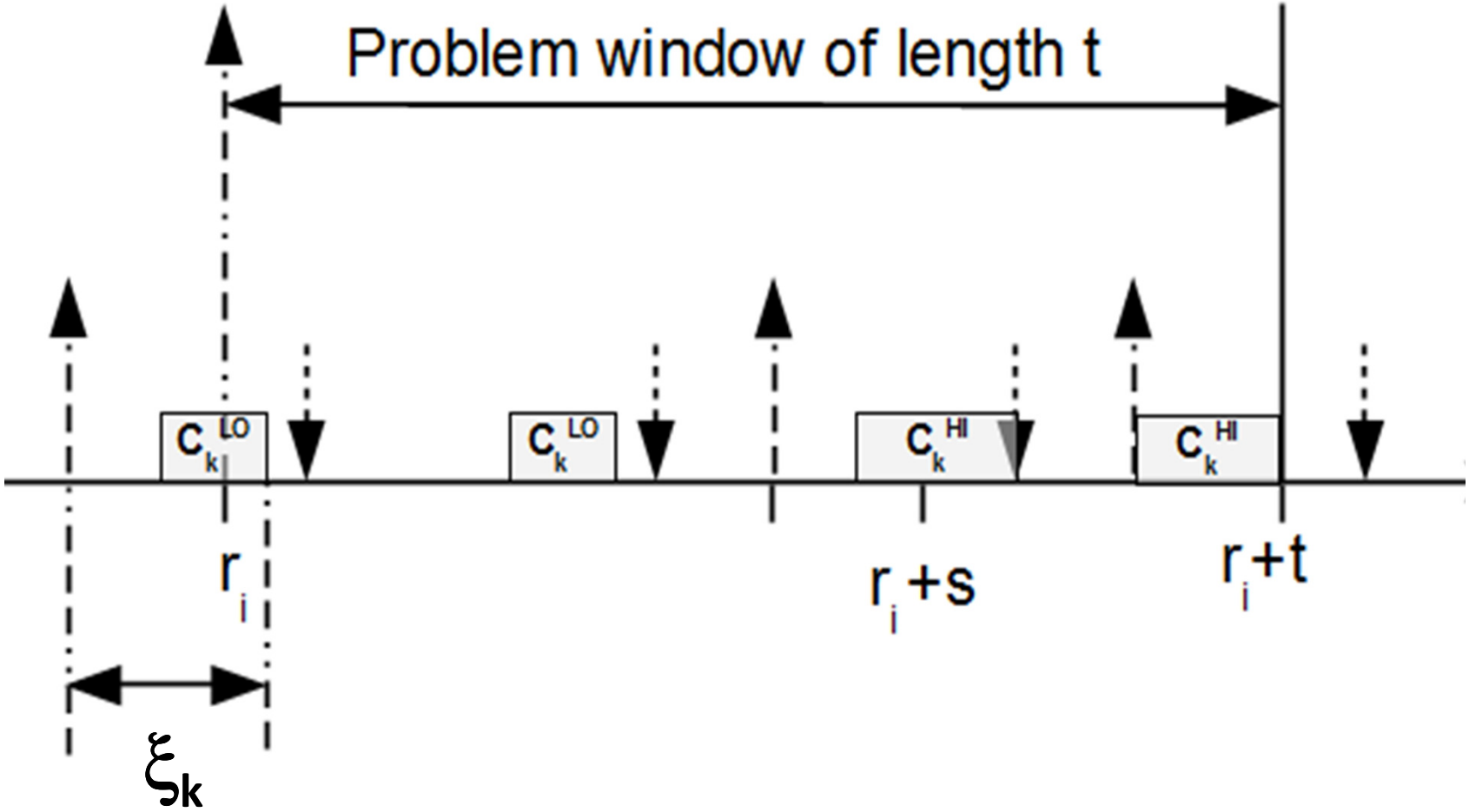
Non-Carry-in and Carry-in workload of each task in **hpHc(i)**

Non carry-in workload $\tau_k \in$ hpHc(i)



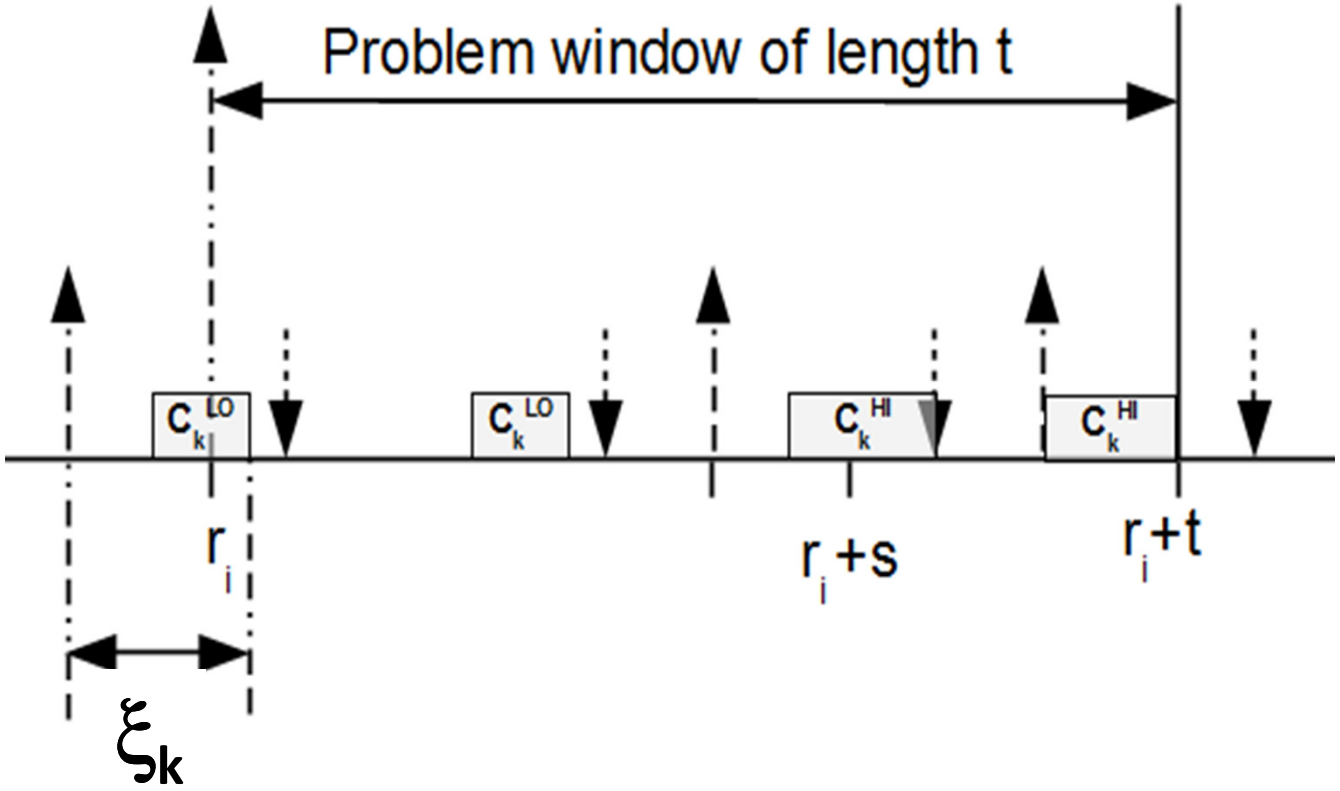
At least $\lfloor s/T_k \rfloor$ jobs of τ_k do not execute the additional part, i.e., $(C_k^{HI}-C_k^{LO})$

Carry-in workload $\tau_k \in$ hpHc(i)



Reference Pattern

Carry-in workload $\tau_k \in$ hpHc(i)



- Reference pattern **is not the worst-case**

Carry-in workload = workload in reference pattern

$+ (C_i^{HI}-C_i^{LO})$

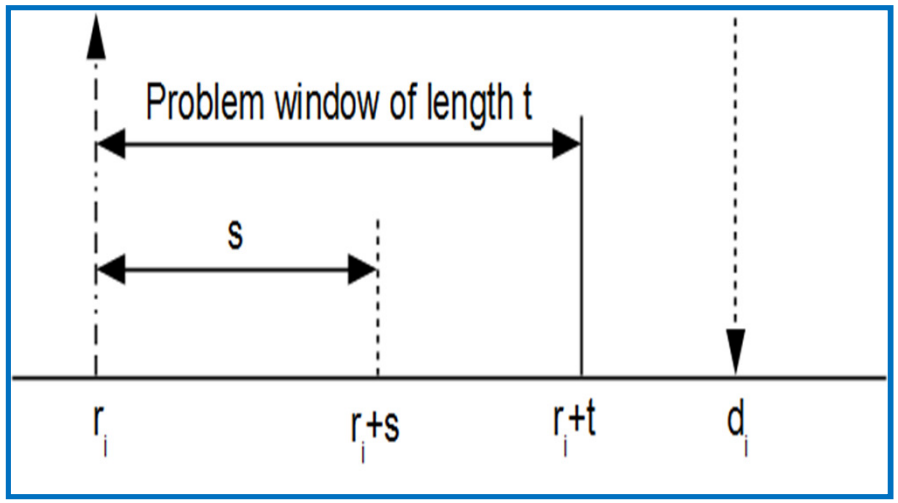
The Response-Time Test

Interference for given **s** and **t**: $I_i(s,t)$

$R_{i,s}^{HI}$ is the solution of

$$t \leftarrow C_i^{HI} + I_i(s,t)$$

$$\mathbf{R_i^{HI} = \max \{ R_{i,s}^{HI} \}}$$



Schedulability Test

- Given a fixed-priority ordering, we find
  - R_i^{LO} for each LO-crit task
  - R_i^{LO} and R_i^{HI} for each HI-crit task
- If all the response times are less than deadline, then the taskset is schedulable
- If not, do we have another priority assignment?
  - Such an assignment can be searched using Audsley's Optimal Priority Assignment (OPA) algorithm

Audsley's OPA

Audsley' OPA algorithm [Davis and Burns, RTSS09]

for each priority level k, lowest first

    for each priority unassigned task τ_i

        If $R_i^{HI} \leq D_i$ and $R_i^{LO} \leq D_i$ assuming higher priorities for the other priority unassigned task, then

            assign τ_i to priority k

            break (continue outer loop)

    return "unschedulable"

return "schedulable"

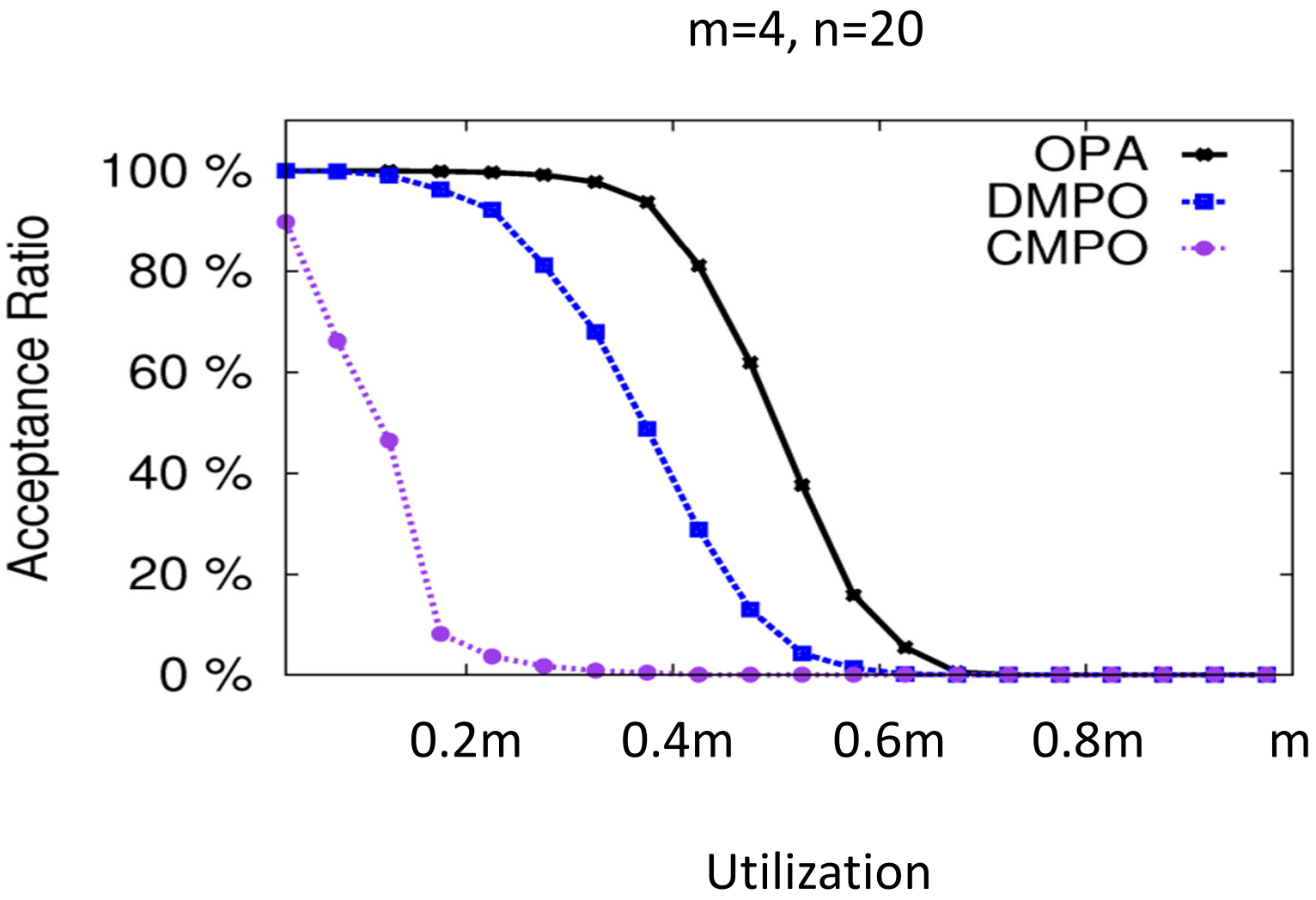
Time-complexity is $O(n^2 . T_{max}^2)$ for dual-criticality

Evaluation

Taskset Generation

- Same as [Baruah, Burns and Davis 2011]
  - But using uunifast-discard algorithm
- Three priority assignment policies: CMPO, DMPO, OPA

Acceptance ratio with increasing load



Conclusion

- Fixed-priority scheduling of MC tasks on multiprocessors
  - Efficient resource utilization and certification
- Applicable for more than two criticality levels
- Priority assignment with Audsley's OPA

Future work: different $T_i = \langle T_i^{LO}, T_i^{HI} \rangle$ where $T_i^{LO} \geq T_i^{HI}$

Thank You

risat@chalmers.se