Algorithmic Complexity of Correctness Testing in MC-Scheduling

Rany Kahil, Dario Socci, Peter Poplavko, Saddek Bensalem

VERIMAG, University of Grenoble Alpes

RTNS'18

Our Work

- **Correctness tests** in **mixed criticality** scheduling and their algorithmic complexity
- Focus on exact **simulation based tests**
- Model the workload of a dual criticality system as a set of **independent jobs** on a single processor
- Contrary to what was believed, checking the correctness of a policy is not linear
- Propose the **Economical Correctness Test (ECT)**
- ECT has a **lower run-time complexity** w.r.t other correctness tests in its class

Motivation

- MC-systems are **not predictable** $=>$ testing for correctness is complicated
- Exact based tests introduce **less pessimism** and are **more accurate** than response time analysis
- **Time efficiency** of simulation based tests is critical since jobs generated over hyper-periods of periodic tasks can be large in numbers
- Existing tests have at least a quadratic complexity, whereas our proposed test has a **quasilinear complexity**

Outline

Mixed Criticality Systems

Advantages and Complexities

Mixed Criticality (MC) systems integrate applications of different criticality level on the same computational platform

Advantages of integration

- Increase efficiency
- Reduce device count
- Reduced Size, weight and power consumption

Difficulties in MC-Systems

- Scheduling becomes harder
- Increase **schedulability tests** complexity (our topic)

Vestal Model (*S. Vestal, 2007*)

in Dual Criticality Systems

Scheduling in Mixed Criticality System

Very pessimistic estimation of WCET for jobs that undergo certification

- Risk of very low processor usage
- Solution: interleaving the execution of jobs of different criticality levels

Vestal Model for finite set of jobs in dual criticality systems

- Every job is is classified as hi-criticality (HI) or lo-criticality (LO)
- Every job is labeled with two Worst Case Execution Times:

  C(LO) computed with industrial standard tools (realistic estimation)

  C(HI) computed with tools compliant to certification authority standards (very pessimistic estimation)

Schedulability conditions of Mixed Criticality System

in Vestal Model

Condition 1

If all jobs respect their $C(LO)$, then both HI and LO jobs must meet their deadline

Condition 2

If at least one job's execution time exceeds its $C(LO)$, then all HI jobs must complete before their deadline (LO jobs may be dropped)

Schedulability conditions of Mixed Criticality System

in Vestal Model

Condition 1

If all jobs respect their $C(LO)$, then both HI and LO jobs must meet their deadline

Condition 2

If at least one job's execution time exceeds its $C(LO)$, then all HI jobs must complete before their deadline (LO jobs may be dropped)

- A **mode switch** occurs at the first instant a job exceeds its $C(LO)$
- A **mode-aware** policy can detect a mode switch and is able to take decisions to drop LO jobs
- A **mode-agnostic** policy is not mode aware

Example of scheduling problem

An **instance J** of the scheduling problem is a set of K jobs

A **scenario** of an instance **J** is a vector of execution times of all jobs

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Example of scheduling problem

An **instance J** of the scheduling problem is a set of K jobs

A **scenario** of an instance **J** is a vector of execution times of all jobs

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Scheduling policies:

- Criticality Monotonic (CM) policy: prioritizes HI jobs over LO jobs
- Mode-Aware EDF policy: Uses EDF priorities but in the case of a mode switch, LO jobs are dropped

Example of scheduling problem

An **instance J** of the scheduling problem is a set of K jobs

A **scenario** of an instance **J** is a vector of execution times of all jobs

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Scheduling policies:

- Criticality Monotonic (CM) policy: prioritizes HI jobs over LO jobs
- Mode-Aware EDF policy: Uses EDF priorities but in the case of a mode switch, LO jobs are dropped

Correctness criteria:

- In scenario c_1 all jobs must meet their deadlines
- In scenario c_2 only HI jobs must meet their deadlines

Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

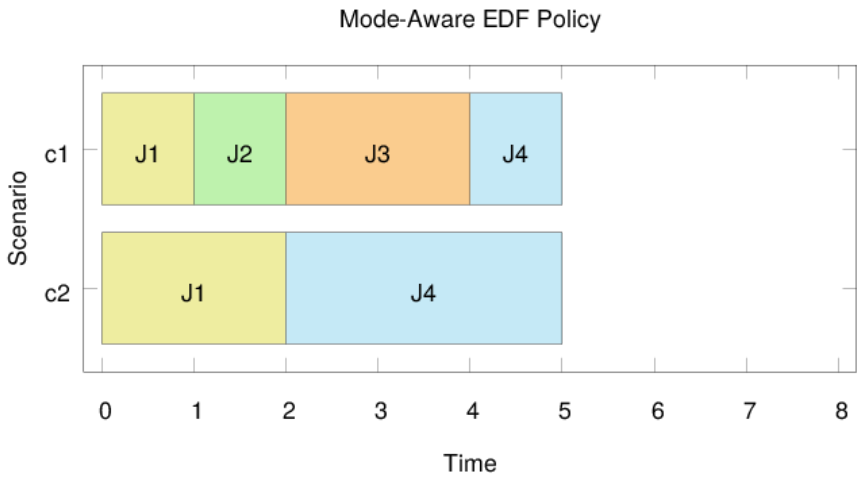
- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

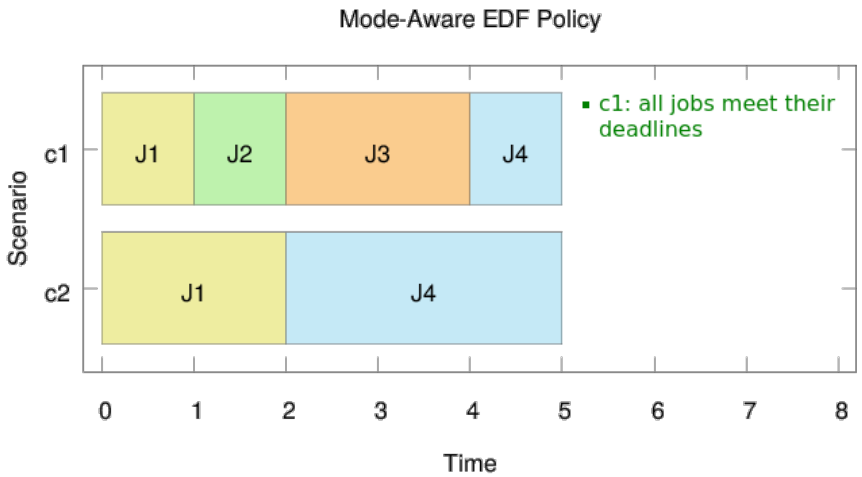


Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

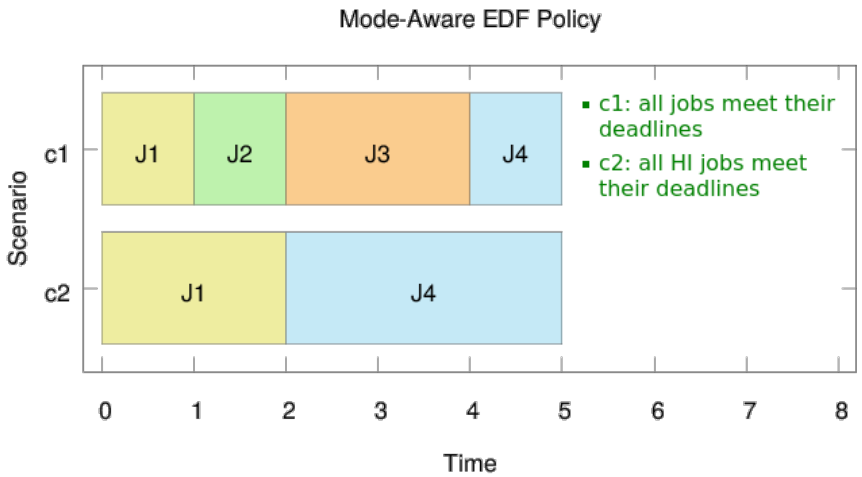


Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$



Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

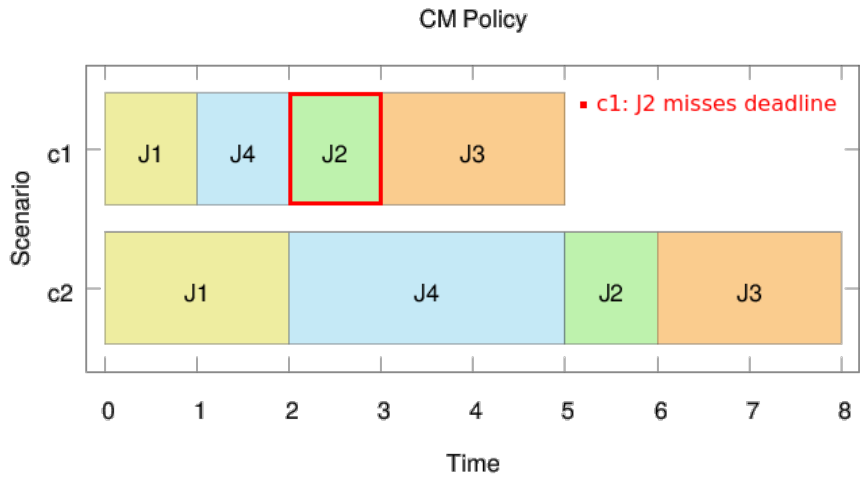
- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$



Example of a scheduling problem

Example (An instance and two scenarios)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

Testing the Schedulability of an Instance

Simulation Based Testing

- An instance is schedulable by a policy iff the policy correctly schedules all its valid scenarios
- Not enough to test only the worst case scenario
- The CM policy correctly schedules the worst case scenario but fails to schedule a simpler scenario
- There are **exponential** number of scenarios
- An **efficient correctness** test is required to guarantee that a given policy is able to correctly schedule all valid scenarios without testing all of them

The Claim of Linear Complexity

- In the original version of [Baruah et al. (2012)] it was claimed that if an instance is schedulable then there exists an **optimal scheduling policy** that can generate a schedule with a **linear** number of preemptions
- Show by a **counter example** that the above claim does not hold in general
- **Revisited lemma.** If an instance is MC-schedulable, then there exists an optimal online scheduling policy that preempts each job j only at time points t such that at time t either some other job is released, or j has executed for exactly $C_j(i)$ units of time for some $1 \leq i \leq L$

The Counter Example

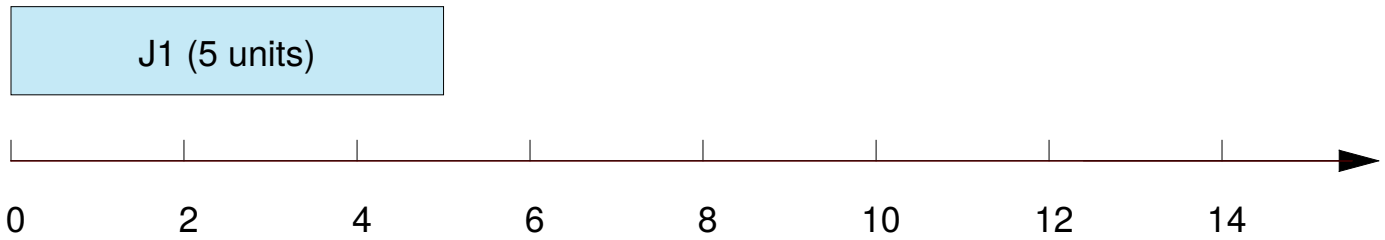
Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3

- According to the revisited lemma jobs can be preempted only at $t = 5$ *i.e.*, when J_3 arrives
- Two possibilities at $t = 0$:
  - J_1 executes
  - J_2 executes

The Counter Example

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3

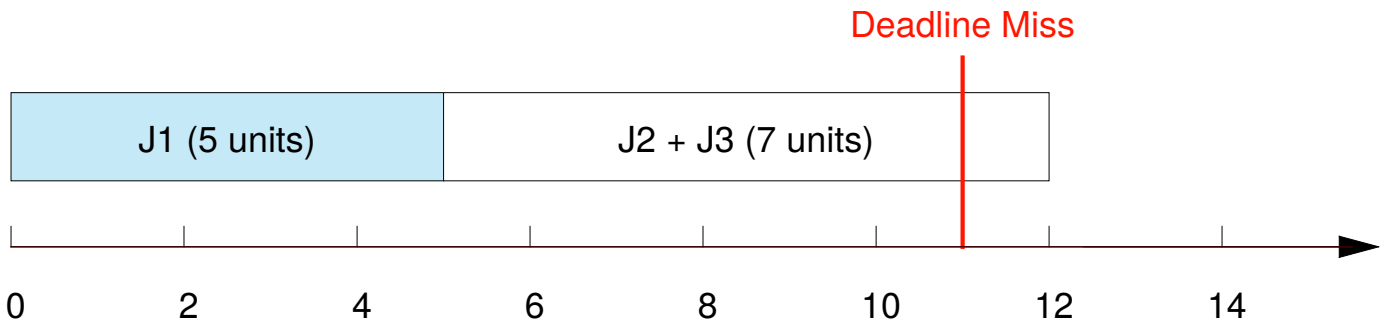
- Assume job J_1 starts the execution



The Counter Example

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3

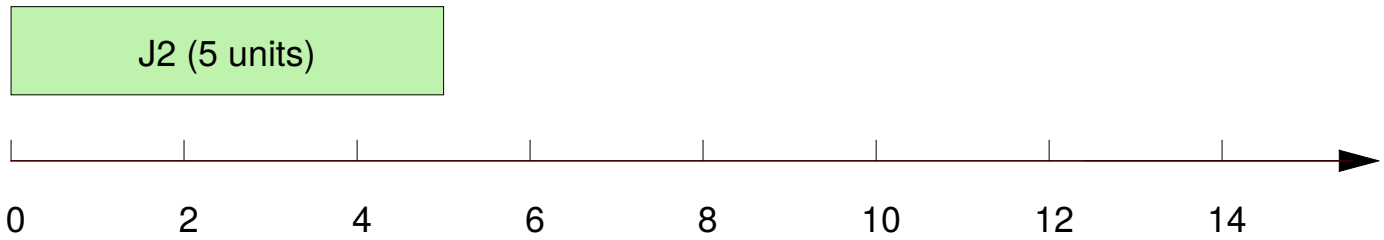
- Assume job J_1 starts the execution
- Assume after $t = 5$ we have the case where:
  - J_2 does not signal termination before it executes for 5 time units
  - J_3 does not signal termination before it executes for 2 time units



The Counter Example

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3

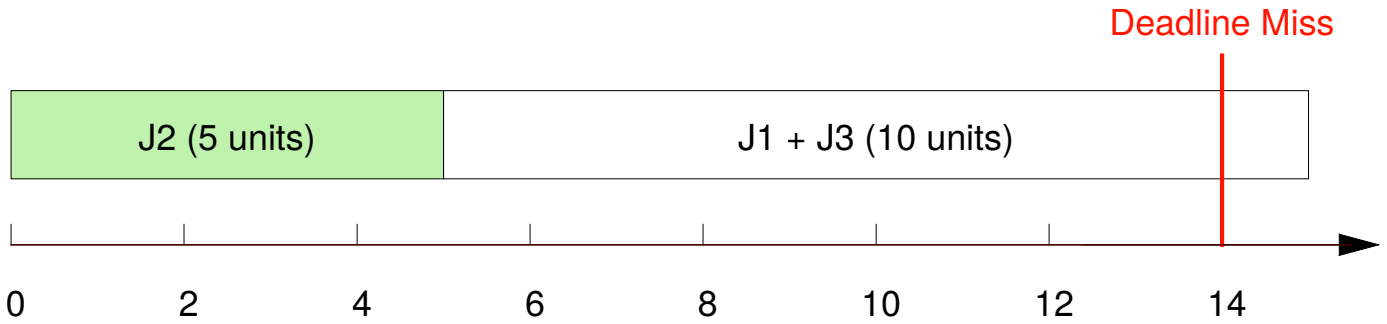
- Assume job J_2 starts the execution



The Counter Example

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3

- Assume job J_2 starts the execution
- Assume after $t = 5$ we have the case where:
  - J_1 does not signal termination before it executes for 7 time units
  - J_3 does not signal termination before it executes for 3 time units

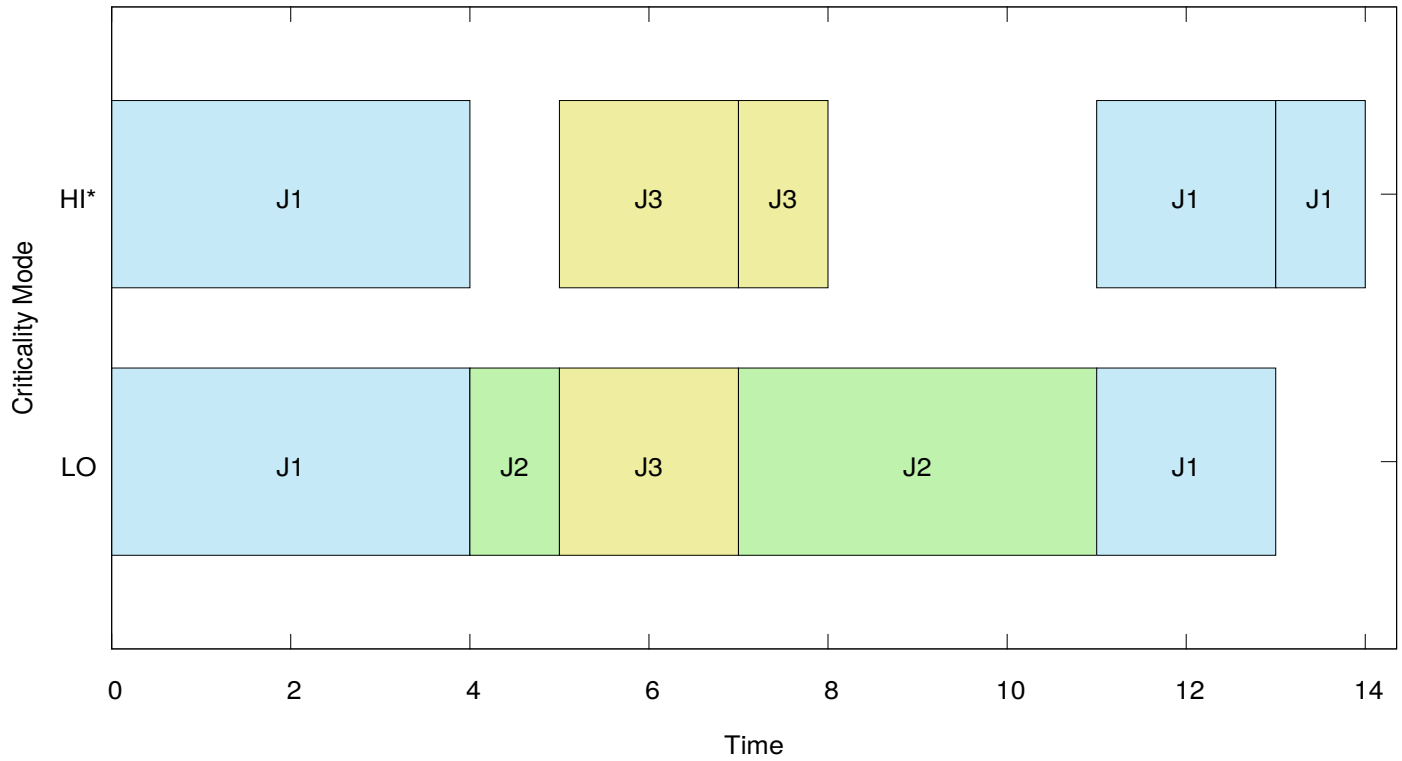


The Counter Example

- According to the 'revisited lemma' this instance is not schedulable
- A correct scheduling policy exists but it must preempt jobs at different time than allowed by the lemma
- In the next slide we give an STTM policy that correctly schedules the instance
- **Single Time Table per Mode (STTM)** policy assigns one time triggered table for each criticality mode
  - In a dual-critical system it has two tables the *LO* and *HI**
  - The schedule starts by using the *LO* table and in case of a mode switch it switches to using the *HI** table

The Counter Example

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	14	HI	6	7
2	0	11	LO	5	5
3	5	10	HI	2	3



Complexity of Existing Correctness Tests

- Using the revisited lemma it was proved that the scheduling problem belongs to class NP
- After our counter example the authors published an erratum to their proof
- The problem is in NP but a quadratic number of preemptions is needed instead of linear
- FPM is a particular case where linear number of preemptions is enough

Complexity of Existing Correctness Tests

- So far we considered the number of events needed to simulate one scenario
- Now consider the number of scenarios that have to be simulated
  - Total cost = num scenarios **x** num events per scenario
- Existing Canonical Correctness Test (CCT) tests linear number of scenarios
  - The cost of CCT is at least cubic in general
  - For FPM the cost is at least quadratic
- Propose Economical Correctness Test (ECT) tests two scenarios
  - The cost of ECT is at least quadratic in general
  - For FPM the cost is $n \log n$

The Economical Correctness Test

Overview

- ECT is an improvement over CCT
- CCT needs to test a linear number of scenarios which cover all the rest, ECT tests only two
- ECT transforms the policy into an **equivalent STTM** policy
  - Two policies are **equivalent** if they successfully schedule the same set of instances
- The tables of the transformed STTM policy are tested for correctness

The Economical Correctness Test

Correctness of STTM

The STTM policy correctly schedules an instance if the following conditions are true:

- All jobs are scheduled after their arrival and before their deadline
- Allocate for each job $C_j(\text{LO})$ time units in LO table
- Allocate for each HI job $C_j(\text{HI})$ time units in HI^* table
- If at any time a switch occurs from LO to HI^*, then all non-yet-terminated HI jobs will have enough time to continue their execution so as to reach $C_j(\text{HI})$ time units

Generating the LO Table

- The LO table is generated by simulating the scheduling policy for the LO basic scenario (where all jobs execute for their $C(LO)$)
- In an FPM policy this requires scheduling decisions to be taken only at *arrival* and *termination* times
- An event based simulator is used having two events per job **Event.Arrive** and **Event.Terminate**

Theorem

In the case of FPM, the complexity of generating the LO table is $O(n \log n)$ *where* n *is the number of jobs*

Generating the HI* Table

The Idea

- A **requirement of the HI* table**: a non-switched job should not execute in the HI* table for more time than it does in the LO table
- This requirement guarantees that after a mode switch all non-finished *HI* jobs will have enough time to terminate if they are given $C(HI)$ execution time in HI* before their deadline
- A set of **enabling/disabling rules** are used to prevent jobs from executing in HI* in a way that violates the requirement
- A disabled job is simply removed from the ready queue until it is enabled again

Generating the HI* Table

Enabling/Disabling Rules

A job J_j is *enabled*, *i.e.*, it is placed in the ready queue, at time t if it is ready and at least one of the following *rules* is true:

Enabling/Disabling Rules

1. $T_j^{LO}(t) = C_j(\text{LO})$
2. $T_j^{HI*}(t) < T_j^{LO}(t)$
3. $T_j^{HI*}(t) = T_j^{LO}(t) \wedge \ S^{LO}(t) = J_j$

$T_j^{LO}(t)$ (resp. $T_j^{HI*}(t)$) denotes the cumulative execution progress of job J_j by time t in table **LO** (resp. **HI***)

Generating the HI* Table

The Simulation

- Information from the LO table is needed in this simulation
- Additional events are needed to enforce the disabling rules:
  - **Event.LO-START** and **Event.LO-STOP** mark the beginning and the end of execution times of HI jobs in S_{LO} table
  - **Event.Disable** used to remove a job from the ready queue
- All jobs are considered disabled at the beginning of the simulation
- A disabled job is enabled when an **Event.LO-START** occurs

Theorem

The transformed simulation has the same algorithmic complexity as the non-transformed one which is $O(n(\log n))$

ECT Example

Example (Our Previous Instance)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

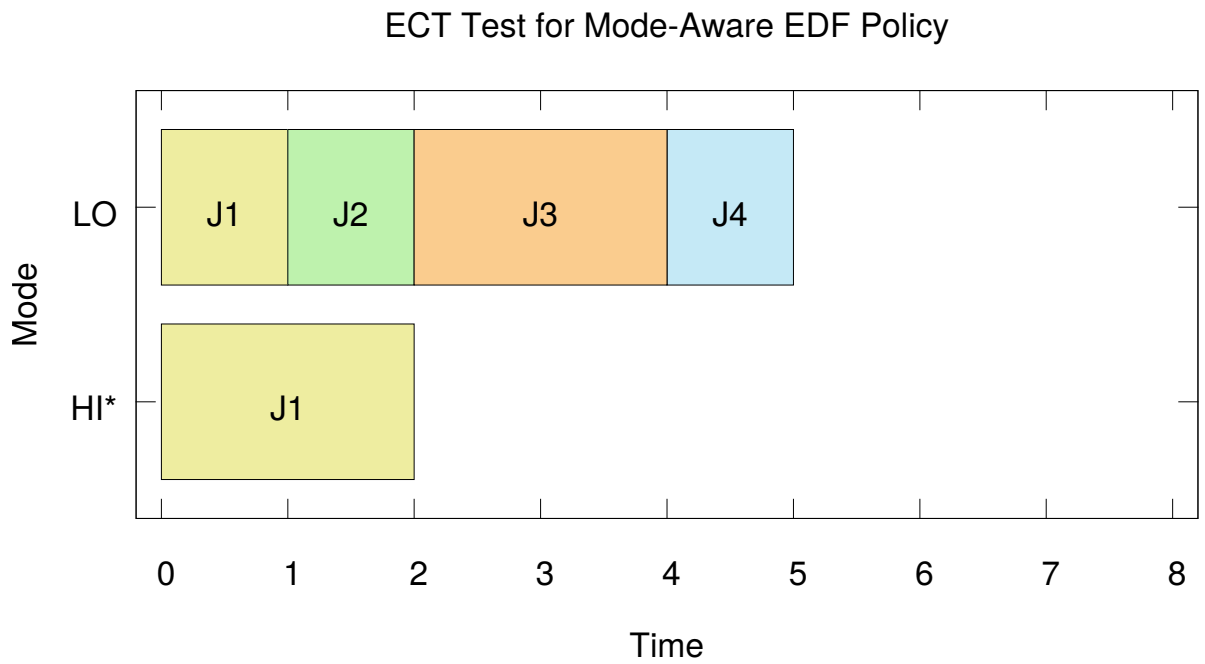
- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

ECT Example

Example (Our Previous Instance)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

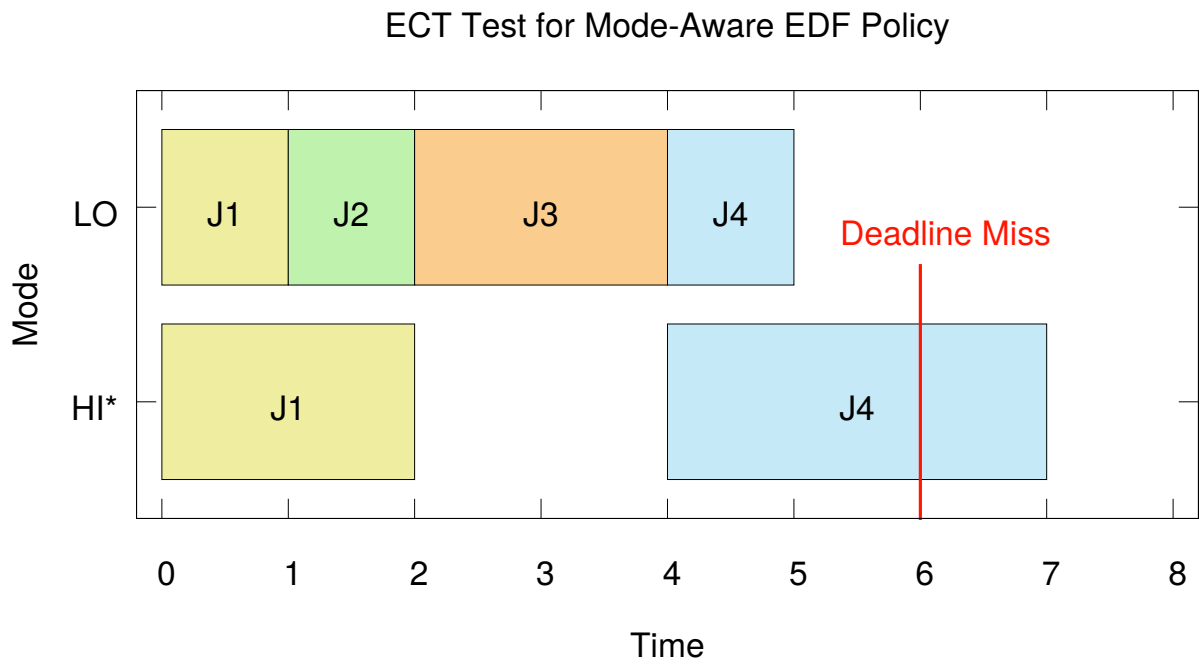


ECT Example

Example (Our Previous Instance)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$

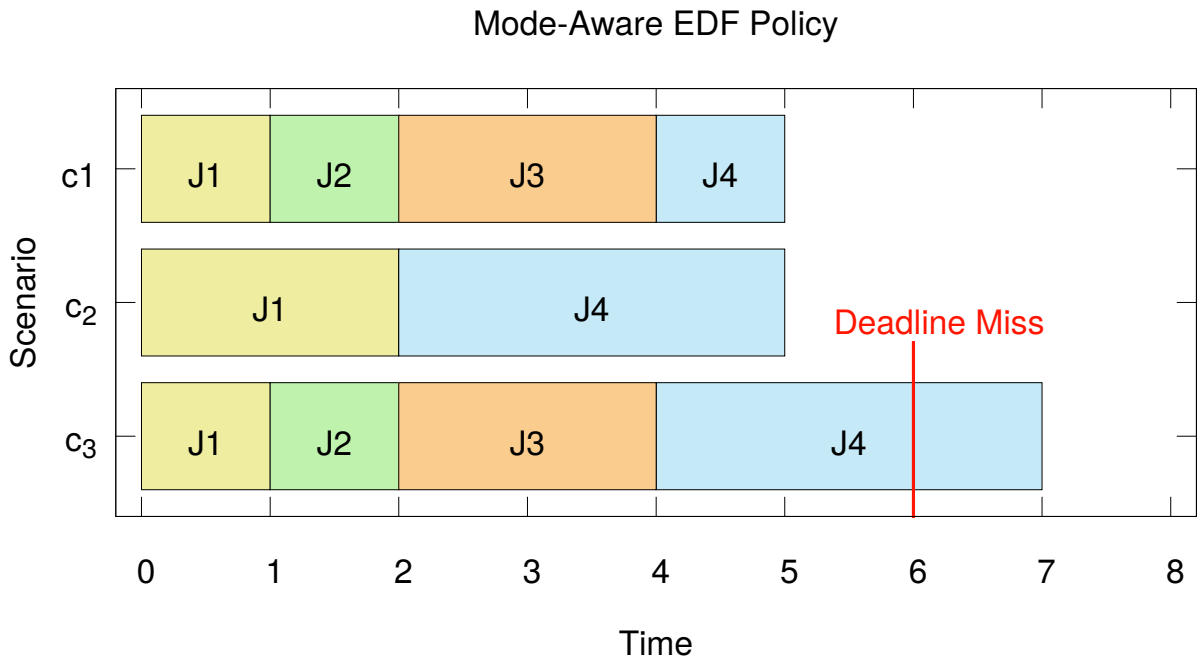


ECT Example

Example (Our Previous Instance)

Job	A	D	χ	$C(LO)$	$C(HI)$
1	0	2	HI	1	2
2	0	2	LO	1	1
3	0	5	LO	2	2
4	0	6	HI	1	3

- $c_1 = (1, 1, 2, 1)$
- $c_2 = (2, 1, 2, 3)$
- $c_3 = (1, 1, 2, 3)$



Conclusion and Future Work

- Taking scheduling decisions at a linear number of events is not always enough to successfully schedule an instance
- Proposed the Economical Correctness Test (ECT) to test dual criticality instances on single processor
- ECT Transforms a policy into an equivalent STTM policy with two time tables
- The time-complexity of ECT is $O(n \log n)$ for FPM which is mode-aware, the same cost as for mode-agnostic FP

Future Work

Identify ECT's limits and study its applicability on a real system, by evaluating its performance w.r.t. the size of a periodic task system

References I

Baruah, S., Bonifaci, V., D'Angelo, G., Li, H., Marchetti-Spaccamela, A., Megow, N., & Stougie, L. (2012). Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, *61*(8), 1140–1152.