Directed Acyclic Graph Scheduling for Mixed-Criticality Systems

Roberto MEDINA
Laurent PAUTET
Etienne BORDE

June 15th 2017

Industrial Context

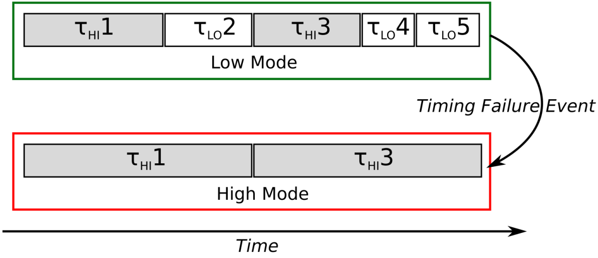
Current industrial needs in Real-Time and Safety-Critical systems

- Integrate more functionalities thanks to **multi-core architectures**.
  - Tasks with different criticalities share an architecture.
- Designers objectives differ from Certification requirements.
  - Designers: optimize performance on resource usage.

    → Estimated timing budgets.
  - Certification: strict guarantees on critical services.

    → Worst Case Execution Timing budget (WCET).
- Safety and Availability needs to be ensured.
  - Critical services always delivered (safety).

Execution Model: various timing budgets

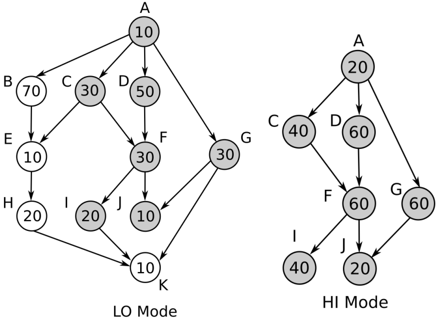


Mixed-Criticality (MC) systems

- Modes of execution (high and low modes): different timing budgets for each mode.[1]
  - Low mode: estimated timing budgets.
  - High mode: WCET.
- Low criticality mode: high (HI) and low (LO) tasks.
- High criticality mode: only high (HI) tasks.

[1] Steve Vestal. "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance". In: *RTSS* (2007).

Execution Model: Mixed-criticality dataflow graphs (MC-DFG)



Data Driven Applications

- Dataflow graphs of tasks: data dependencies and parallel execution.
- Global deadline for the graph.
- Tasks have two timing budgets and use it all (Time Triggered approach[2]).

[2] Hermann Kopetz. "The time-triggered model of computation". In: 1998.

Research question

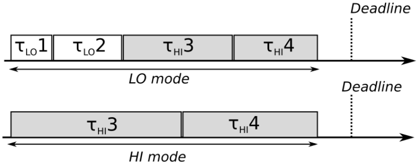
Find a safe and efficient schedule for MC-DFG on multi-core architectures.

- MC scheduling: task models rarely consider data dependencies[3].
- DFG: model is *static*, graph properties do not change[4].
- Scheduling is complex: precedence constraints, constrained platforms.

[3] Alan Burns and Robert Davis. "Mixed Criticality Systems - A Review". In: 2017.
[4] Adnan Bouakaz, Jean-Pierre Talpin, and Jan Vitek. "Affine data-flow graphs for the synthesis of hard real-time applications". In: 2012.

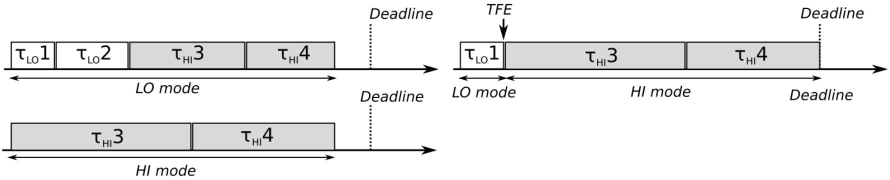
Safe schedule for MC systems

Safe mode transitions in MC systems?



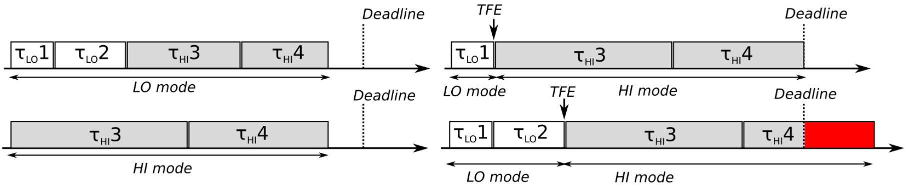
Safe schedule for MC systems

Safe mode transitions in MC systems?



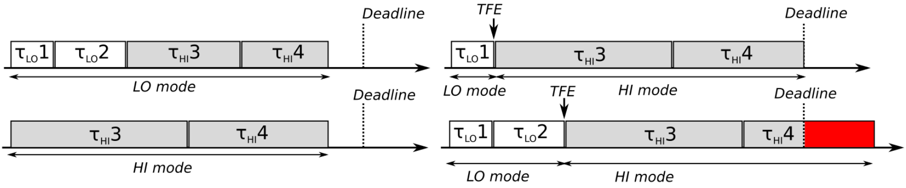
Safe schedule for MC systems

Safe mode transitions in MC systems?



Safe schedule for MC systems

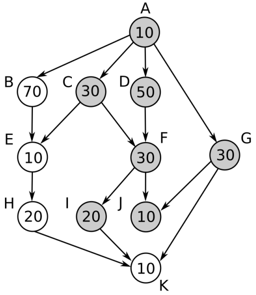
Safe mode transitions in MC systems?



- HI tasks WCET extended in HI mode $\rightarrow$ *deadline miss* may occur.
- Existing solution: run the HI tasks ASAP (even in LO mode).[5]

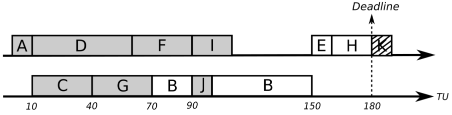
[5] Sanjoy Baruah. "Implementing mixed-criticality synchronous reactive systems upon multiprocessor platforms". In: 2013.

Illustrative example:

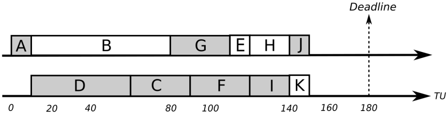


LO Mode

- HI tasks ASAP: unschedulable.



- Ignoring mode transitions:



Proposed Scheduling Approach

Overview of our Scheduling algorithm

- **Step 1**: HI scheduling table (similar to Least Laxity).
- **Step 2**: Deduction of latest safe activation instants for HI tasks.
- **Step 3**: LO scheduling table (considering activation instants of HI tasks).

We use *List Scheduling* (LS) to schedule DAGs.[6]

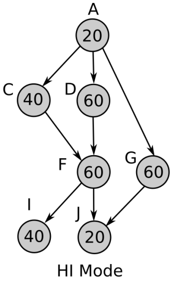
- LS creates a priority ordering of tasks to allocate them.
- *Migrations* and *preemptions* of tasks in LO mode.

[6] Yu-Kwong Kwok and Ishfaq Ahmad. "Benchmarking and Comparison of the Task Graph Scheduling Algorithms". In: (1999).

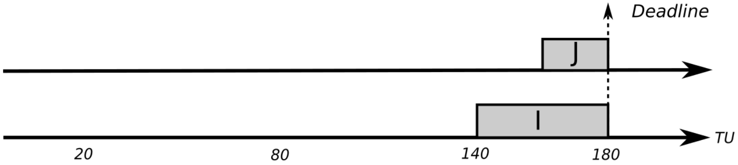
HI scheduling table computation

- Obtain the priority ordering using LS (considering HI mode budgets).
- *Reverse schedule* the DAG in HI mode (from deadline to instant 0).
- Latest instants at which HI tasks are able to be executed in HI mode.
- These instants are called Latest Safe Activation Instant (LSAI).

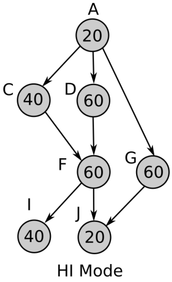
HI scheduling example (step 1)



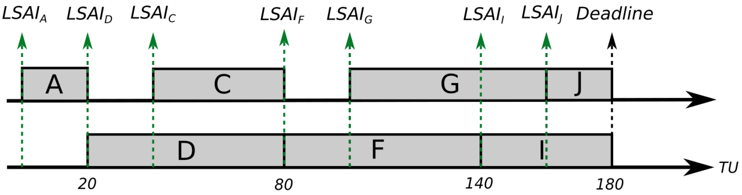
Priority ordering (longest path to an exit node):
$\langle (A, 180), (D, 160), (C, 140), (F, 100), (G, 80), (I, 40), (J, 20) \rangle$



HI scheduling with LSAI (step 1 & 2)



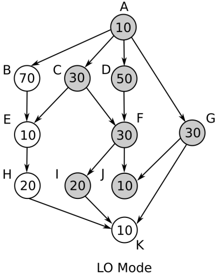
Priority ordering (longest path to an exit node):
$\langle (A, 180), (D, 160), (C, 140), (F, 100), (G, 80), (I, 40), (J, 20) \rangle$



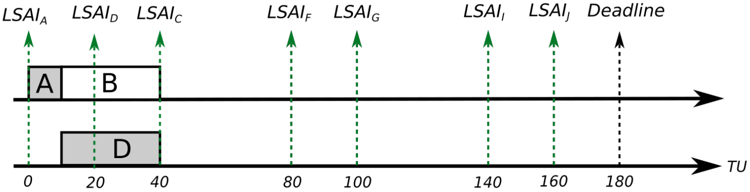
LO scheduling table computation

- Obtain the priority ordering using LS **(considering LO mode budgets)**.
- Construct the table slot by slot (from 0 to deadline).
  - If a slot corresponds to a LSAI, corresponding HI task is promoted.
  - **Preemption of LO tasks can occur**.
  - Promoted HI tasks are executed until they finish.
- If the deadline is reached and there are still tasks to be executed, the DAG is non schedulable.

LO scheduling table example (step 3)

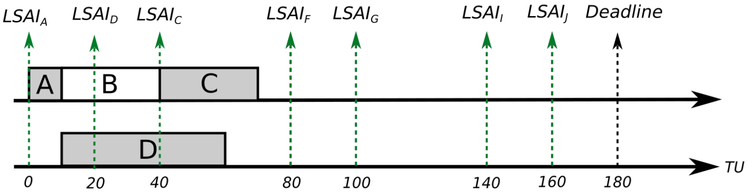


Priority ordering (longest path to an exit node):
$\langle$(*A*, 120), (*B*, 110), (*D*, 110), (*C*, 90), (*F*, 60), (*G*, 40), (*E*, 40), (*H*, 30), (*I*, 30), (*J*, 10), (*K*, 10)$\rangle$



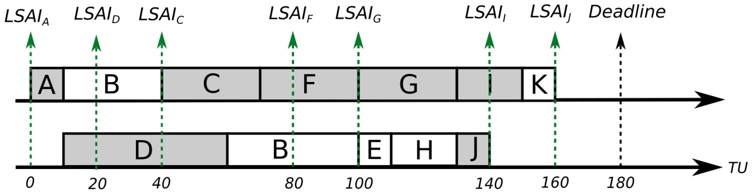
LO scheduling table example (step 3)

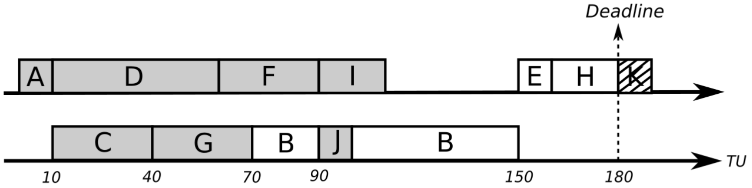
At TU = 40, LSAI for C→ C promoted with highest priority.
Priority ordering (longest path to an exit node):
$\langle (C, max), (D, max), (B, 110), (F, 60), (G, 40), (E, 40), (H, 30), (I, 30), (J, 10), (K, 10) \rangle$



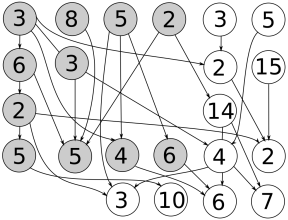
LO scheduling table example

Priority ordering (longest path to an exit node):
$\langle (A, 120), (B, 110), (D, 110), (C, 90), (F, 60), (G, 40), (E, 40), (H, 30),$
$(I, 30), (J, 10), (K, 10) \rangle$





Evaluation of the Scheduling Approach



Unbiased DAG generation for MC:

- Parallelism degree + edge probability.[7]
- Utilization of tasks in HI and LO mode[8].
- Utilization of HI tasks in LO mode.

[7] Abusayeed Saifullah et al. "Parallel real-time scheduling of DAGs". In: 2014.

[8] Paul Emberson, Roger Stafford, and Robert I Davis. "Techniques for the synthesis of multiprocessor tasksets". In: 2010.

MC DAG generation

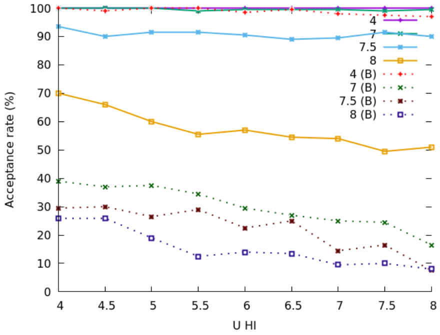
Overview:

1. Create nodes in HI mode until U_{HI} is reached.
2. Reduction of the DAG until U_{HIinLO} is reached.
3. Complete U_{LO} with LO nodes.

Benchmarking Results

Full lines: our approach.
Dotted lines: existing approach of the literature.[9]

- Tested DAGs are schedulable ignoring mode transitions.
- Progressively increment U_{LO} and U_{HI} until reaching the max utilization.
- Test the same DAGs with the two approaches.

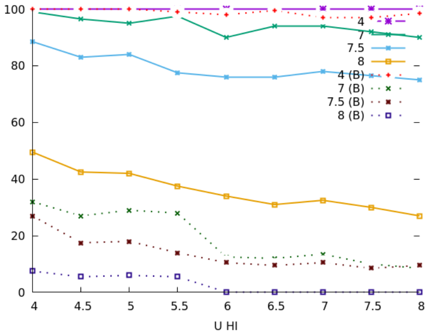


20% edge probability, 8 cores.

[9] Sanjoy Baruah. "Implementing mixed-criticality synchronous reactive systems upon multiprocessor platforms". In: 2013.

Benchmarking Results

- Far better acceptance rate (utilization above 7, 8 cores).
- Good acceptance rate even with high utilization (LO and HI mode and dense graphs).
- Efficient scheduling computation: 200 DAGs in 70s.



60% edge probability, 8 cores.

Future work perspectives

Current research perspectives include the following points:

- Availability analysis for our multi-core scheduling approach.[10]
- Can we interrupt only certain LO services to avoid a complete mode switch?
- Schedule multiple DAGs with different deadlines on a single architecture.

[10] Roberto Medina, Etienne Borde, and Laurent Pautet. "Availability analysis for synchronous data-flow graphs in mixed-criticality systems". In: *Proceedings - SIES* (2016).