# Probabilistic Analysis of Low-Criticality Execution

Martin Küttler[1] Michael Roitzsch[1]
Claude-Joachim Hamann[1] Marcus Völp[2]

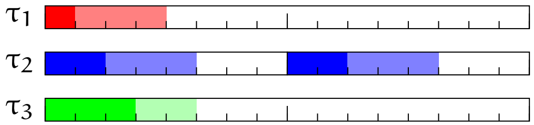
*[1] Technische Universität Dresden* *[2] University of Luxembourg*

5th December 2017

## Probabilistic Analysis of Low-Criticality Execution

- Traditionally in MC, in HI-mode LO-jobs are dropped.
- Keeping LO-jobs as best effort jobs in a low priority band increases the chance of finishing them.
- Given execution time distributions for all tasks, what are the probabilities for each job in a hyperperiod to complete?
- This work presents two approaches to compute these probabilities.

## Analytical Solution

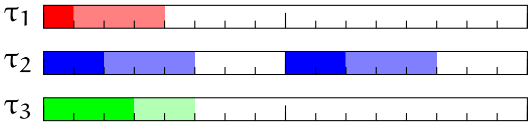


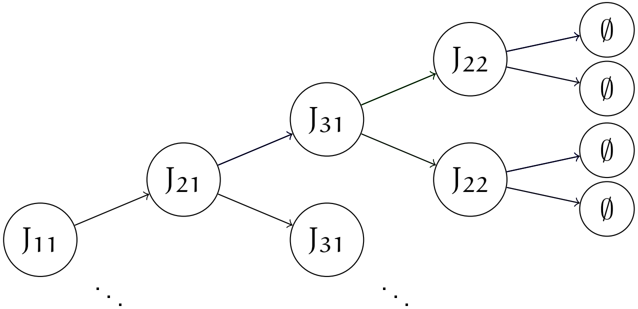
- For each LO-task, split hyperperiod in phases with width = period.
- Iterate over phases, keep track of remaining execution time of HI jobs.
- Add up execution time distributions in each phase, starting from the top.
- Recalculate phases for each new LO-task, to avoid combining non-independent distributions.

There are formulas in the paper for

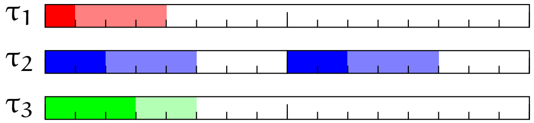
- harmonic task sets with
- critiality-monotonic (static) priorities.

# Enumeration



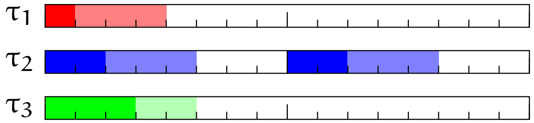


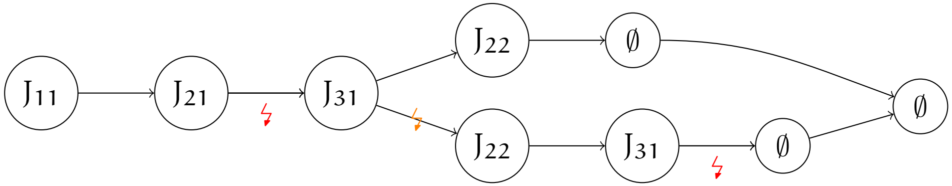
# Symbolic Scheduler



$$\begin{tikzcd}
 & & & J_{22} \arrow[r] & \emptyset & \\
 & & J_{31} \arrow[ur] \arrow[dr, "\lightning"] & & & \\
 & & & J_{22} \arrow[r] & J_{31} \arrow[r] & \emptyset \\
J_{11} \arrow[r] & J_{21} \arrow[uur] \arrow[ddr, "\lightning"] & & & & \\
 & & & & \emptyset & \\
 & & J_{22} \arrow[r] & J_{31} \arrow[ur] \arrow[dr, "\lightning"] & & \\
 & & & & \emptyset &
\end{tikzcd}$$

## Symbolic Scheduler – Reduce Branching





## Symbolic Scheduling – Criticality Switches

- Criticality switches introduce a third branching reason.
- Can't be handled quite like other scheduling events: deadlines and releases are absolute times, criticality switches happen based on execution time.
- The previous optimizations all work, with the caveat that branches with different criticality levels can not be merged.

**Questions?**