Availability Enhancement and Analysis for Mixed-Criticality Systems on Multi-core

Roberto MEDINA, Etienne BORDE, Laurent PAUTET

Design, Automation & Test Europe

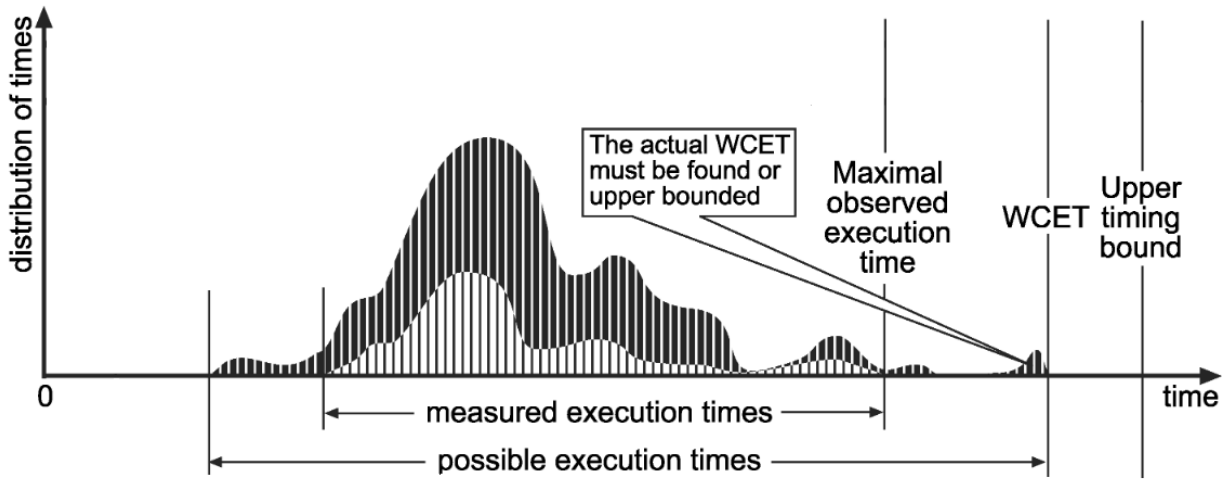
March 22nd 2018

Overview

Research and Industrial Context

- Safety-critical systems incorporate tasks with different criticalities.
  - Life-critical, mission-critical, non-critical.
- Improve resource usage offered by **multi-core architectures** thanks to **mixed-criticality**.
  - Tasks with different criticalities share a multi-core processor.
- **Safety** and **availability** need to be ensured.
  - Critical services always delivered (safety).
  - Non-critical services deliver interesting functionalities (availability).
- **Limits** on the current Mixed-Criticality model.
  - Availability estimation often neglected.
  - Pessimism on mode transitions.
  - Independent task model.

Motivation for Mixed-Criticality

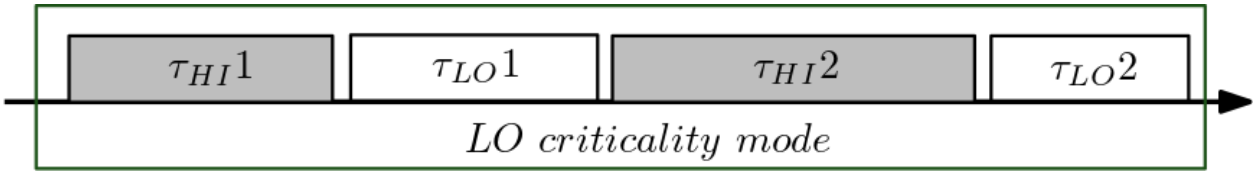


- Estimating Worst-Case Execution Time (WCET) is difficult[1].
- A task rarely executes until its WCET.
- **Problem:** make the most of processing capabilities (*eg.* multi-cores).

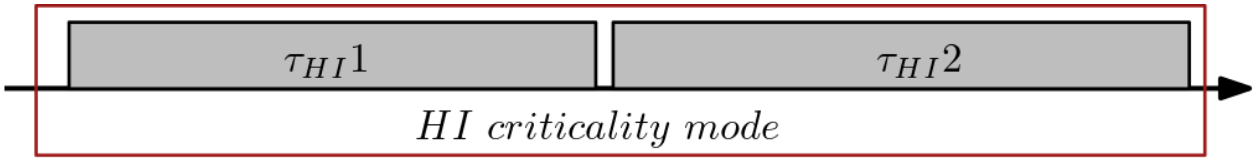
[1] Reinhard Wilhelm et al. "The worst-case execution-time problem—overview of methods and survey of tools". In: *ACM Transactions on Embedded Computing Systems (TECS)* (2008).

Mixed-Criticality Model

- When the maximal observed execution time is used:



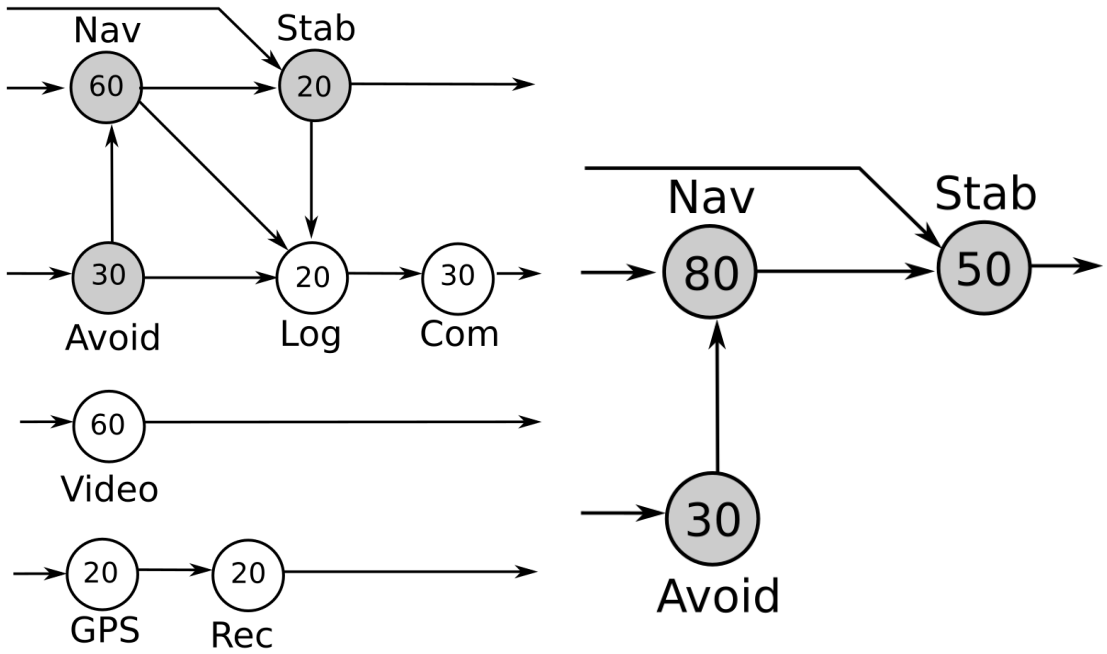
- When upper-bounded WCET is used:



- Tasks have different timing budgets: $C_i(LO)$ and $C_i(HI)$[2].
- Modes of execution ensure the safety of the system.
  - Low criticality mode: high (HI) and low (LO) tasks.
  - High criticality mode: only high (HI) tasks.
- **Timing Failure Events occurs:** switch to the high criticality mode.

[2] Steve Vestal. "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance". In: *Real-Time Systems Symposium*. 2007.

Mixed-Criticality dataflow graphs (MC-DFG)



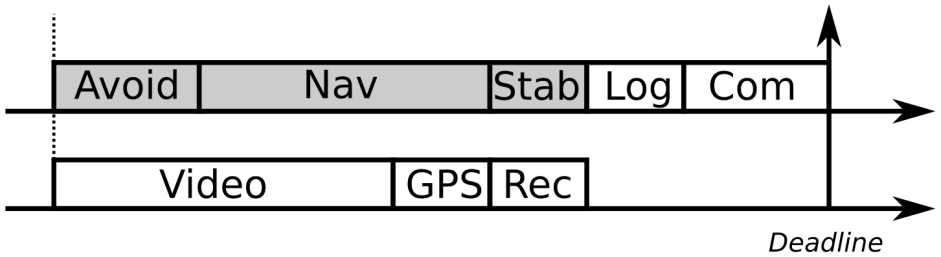
(a) LO Mode (b) HI Mode

- Dataflow graphs of tasks: data dependencies, parallel execution and deterministic scheduling tables.
- Tasks use all their timing budgets: Time Triggered approach[3].
- Often used in flight control and monitor systems.

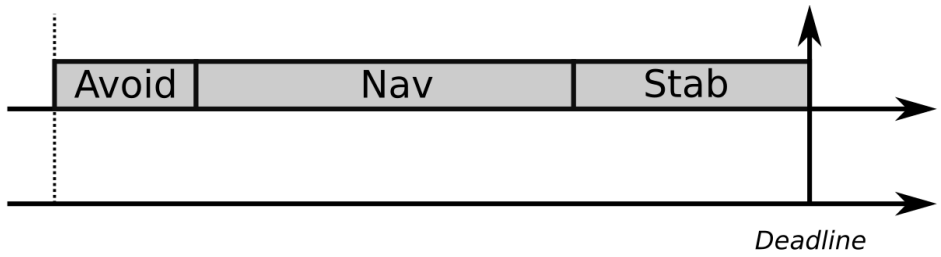
[3] Hermann Kopetz. "The time-triggered model of computation". In: *Real-Time Systems Symposium*. 1998.

Motivating example

Scheduling tables:

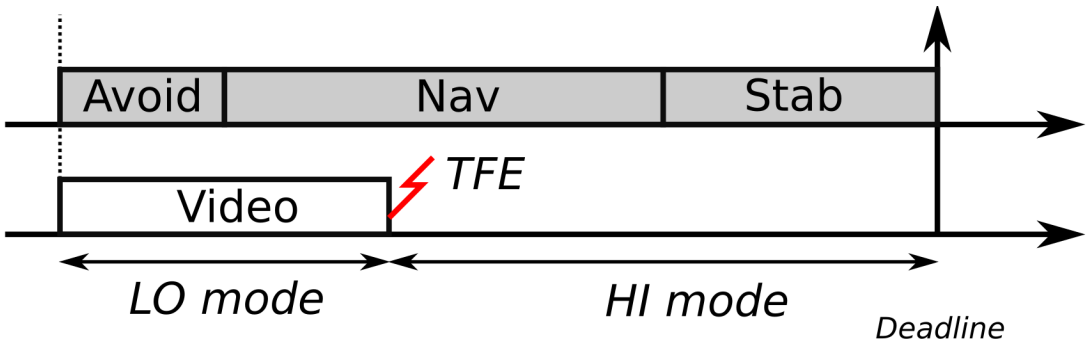


(c) LO mode



(d) HI mode

Classic Mixed-criticality model: when a Timing Failure Event occurs...



- How often are LO services interrupted?
- Do HI tasks actually need the timing extention budget?

Research objectives

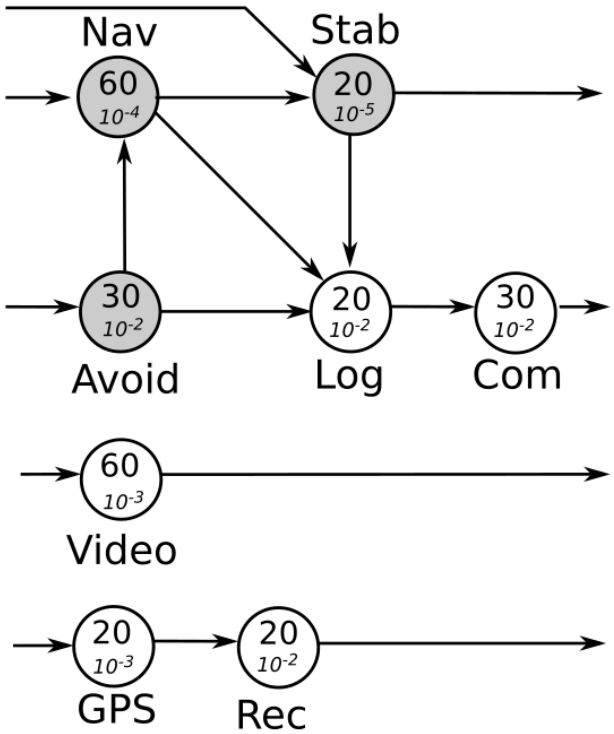
Measure the availability rates of LO criticality services

- Find a formula to compute the availability.
- Simulate the execution of the system.

Improve availability rates of LO services

- Lift pessimism about mode transitions in Mixed-Criticality.
  - Fault propagation model.
- Consider weakly-hard real-time tasks.

Fault Model: failure probabilities



- Failure probability p_{τ_i} for each task.
- Requested by certification authorities.
- *E.g.* Airborne systems: DO-178B Levels A, B, C, D and E. Railroad systems: SIL 1, 2, 3 and 4.

Availability formula for LO criticality services

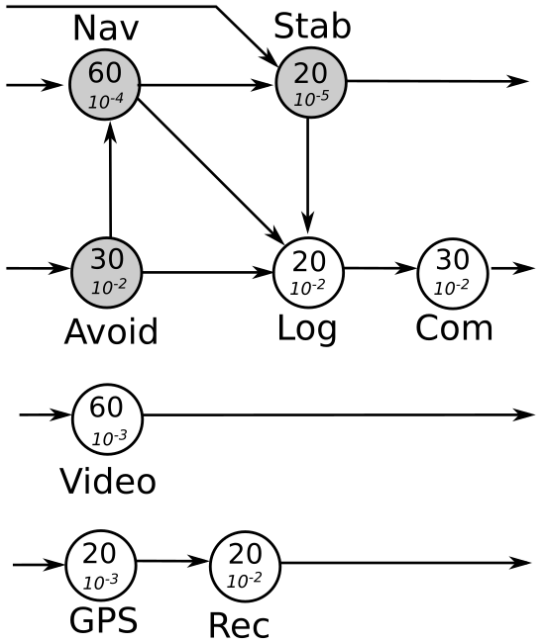
- Availability of a task: its failure probability p_{τ_i} + failure probabilities of tasks executed before it: $pred(\tau_i)$.
- Scheduling tables for the LO mode[4][5] to find the predecessors.

$$A(\tau_i) = 1 - (p_{\tau_i} + \sum_{\tau_j \in pred(\tau_i)} p_{\tau_j}). \quad (1)$$

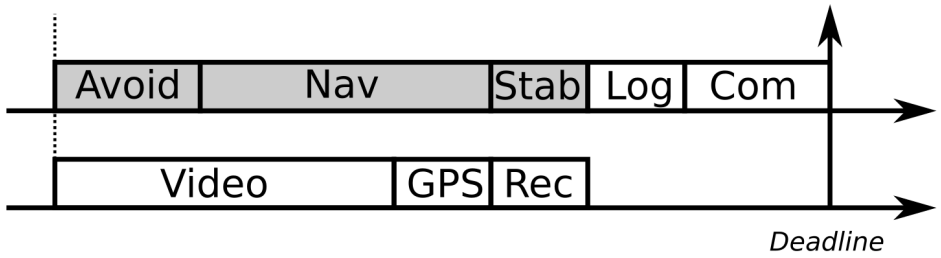
[4] Sanjoy Baruah. "The federated scheduling of systems of mixed-criticality sporadic DAG tasks". In: *Real-Time Systems Symposium*. 2016.

[5] Roberto Medina, Etienne Borde, and Laurent Pautet. "Directed Acyclic Graph Scheduling for Mixed-Criticality Systems". In: *Ada-Europe International Conference on Reliable Software Technologies*. 2017.

Formula applied to our example



(a) Architecture



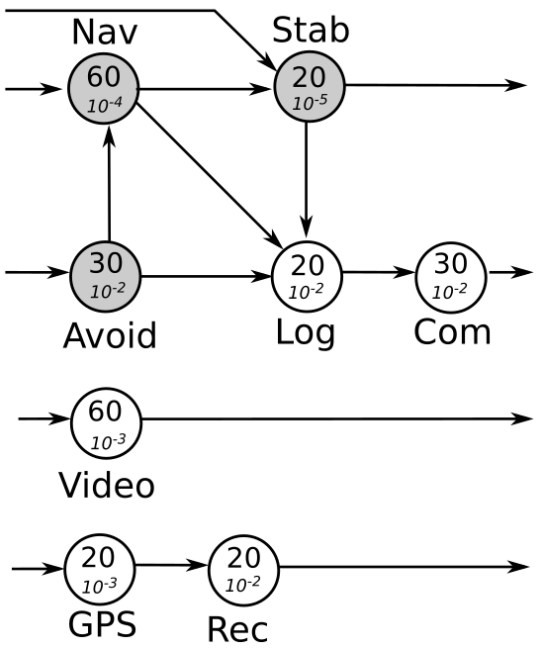
(b) LO scheduling table

Availability for the *Com* task:

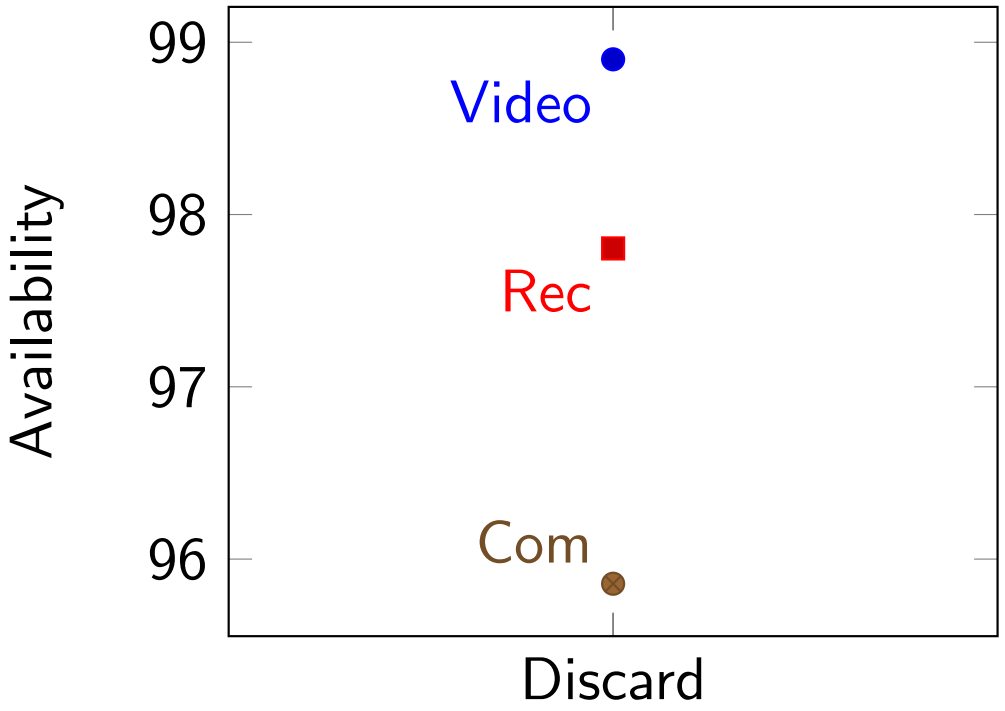
$$A(Com) = 1 - (10^{-2} + \sum_{\tau_j \in pred(Com)} p_{\tau_j}).$$

Where $pred(Com) = \{Avoid,\ Nav,\ Video,\ GPS,\ Stab,\ Rec,\ Log\}$.

First availability computation



(a) Architecture



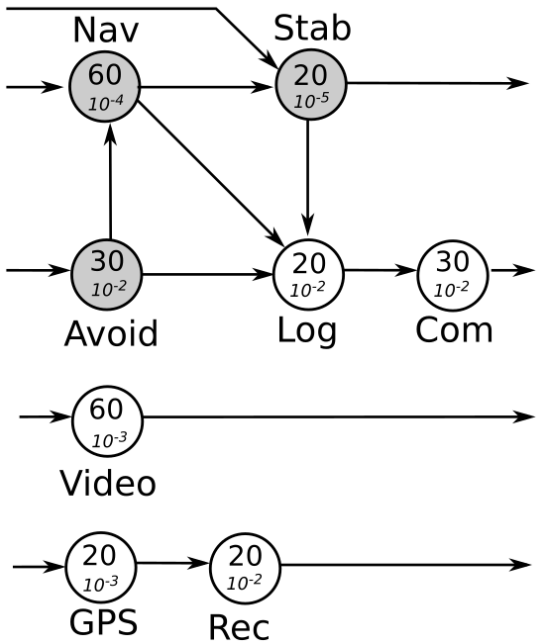
(b) Results

- Pessimistic mode transitions + multi-core architectures.
- Not very good results for *Com* and *Rec*.

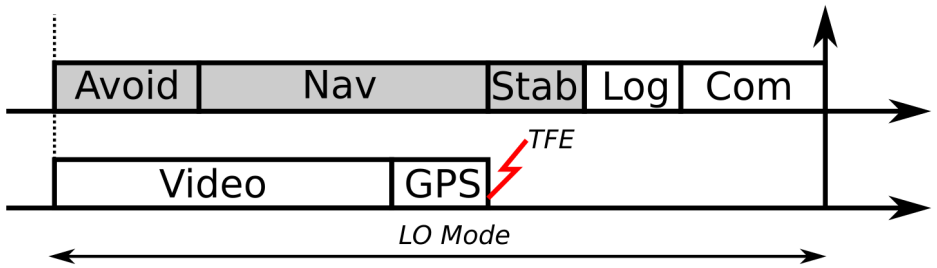
Can this availability rate be improved?

Fault propagation model: improving availability (1/2)

- **Only interrupt communication dependent tasks.**
- Unaffected services can still be delivered.
- Switch to HI mode only when HI tasks have a TFE.



(a) Architecture



(b) Fault propagation

Fault propagation model: improving availability (2/2)

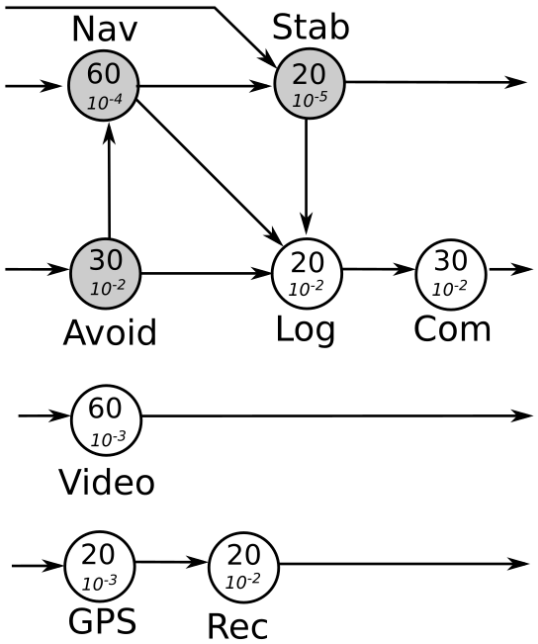
Availability depends on p_{τ_i}, on its graph predecessors and on HI tasks executed before.

$$A(\tau_i) = 1 - (p_{\tau_i} + \sum_{\tau_j \in pred(\tau_i)} p_{\tau_j}). \quad (1)$$

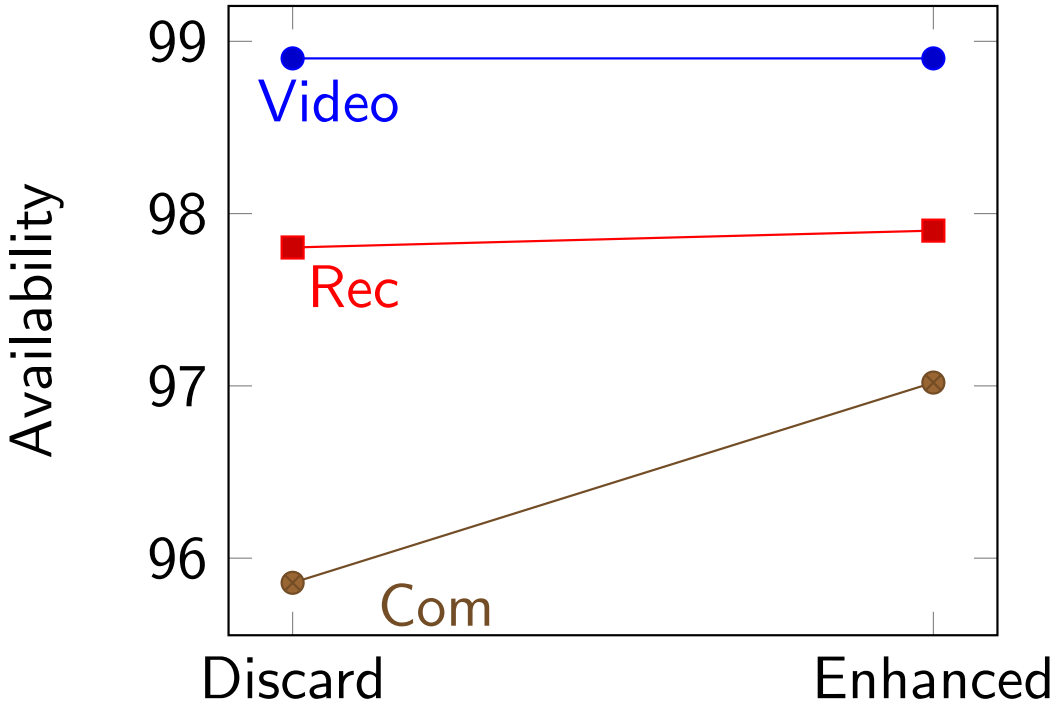
Example: For the *Com* task: $pred(Com) = \{Avoid,\ Nav,\ Stab,\ Log\}$.

$$A(Com) = 1 - (10^{-2} + 10^{-2} + 10^{-4} + 10^{-5} + 10^{-2}).$$

Improving the availability



(a) Architecture



(b) Results

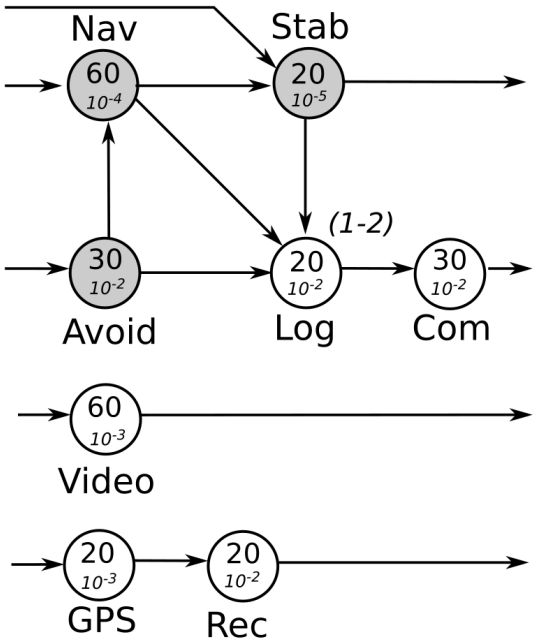
Important availability improvement:

- +0.1% for *Rec*, +1.2% for *Com*.
- Availability often measured at 10^{-5}

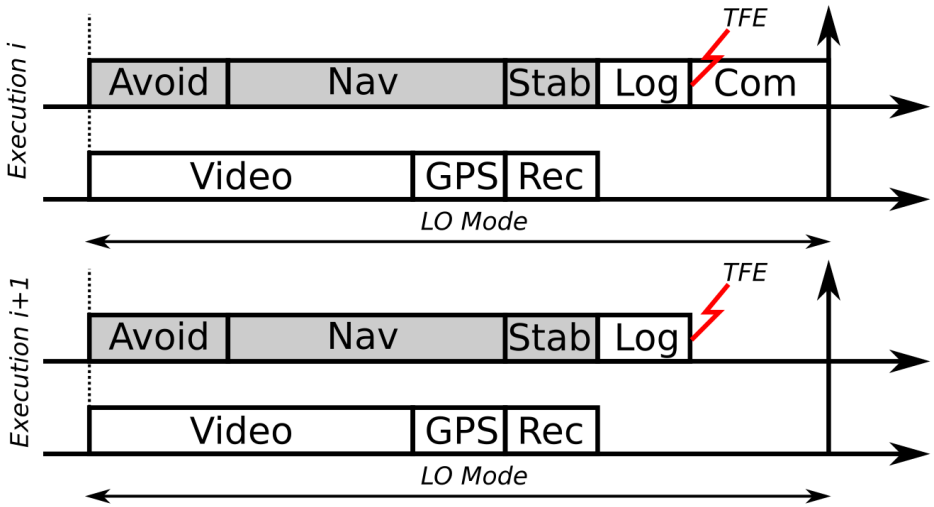
Can we further improve this availability?

Weakly-hard real-time tasks

- Literature only considers hard real-time tasks.
- Incorporate weakly-hard real-time tasks.



(a) Architecture



(b) Example of scheduling

- Tolerate a number m of faults for k successive executions.
- **Problem:** Availability equation cannot be applied anymore.

Availability estimation for LO services

1. Compute scheduling tables for the LO and HI mode.
2. Transformation of the scheduling tables to PRISM automaton[6].
3. Estimate availability rates thanks to simulations of the system.

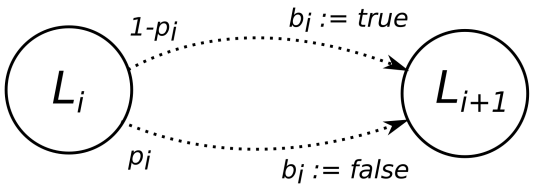
$$A(\tau_i) = \frac{Number\ of\ executions\ of\ \tau_i}{LO_{exec} + HI_{exec}}. \quad (2)$$

[6] Roberto Medina, Etienne Borde, and Laurent Pautet. "Availability analysis for synchronous data-flow graphs in mixed-criticality systems". In: *Industrial Embedded Systems (SIES), 11th IEEE Symposium on*. 2016.

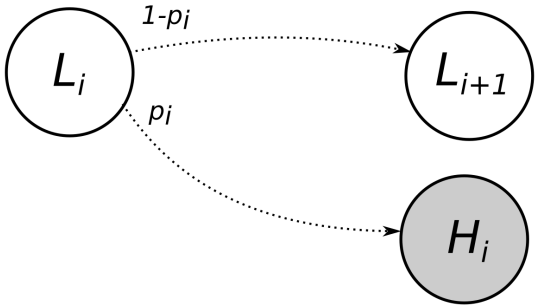
Translation rules to PRISM automata

Why PRISM?

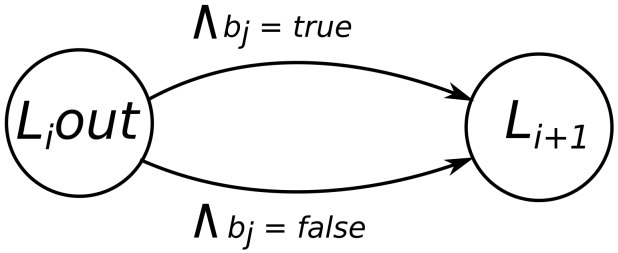
- Capture fault model naturally thanks to probabilistic transitions.
- Represent fault propagation and data production thanks to booleans.



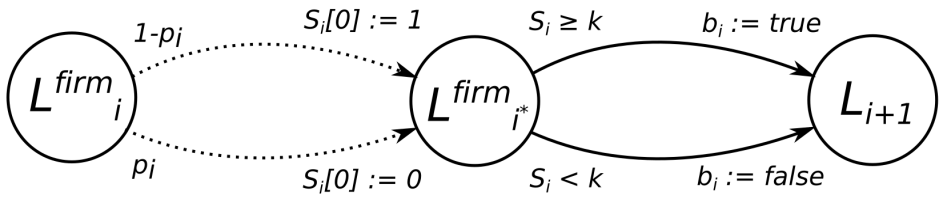
(a) LO task translation



(b) HI task translation

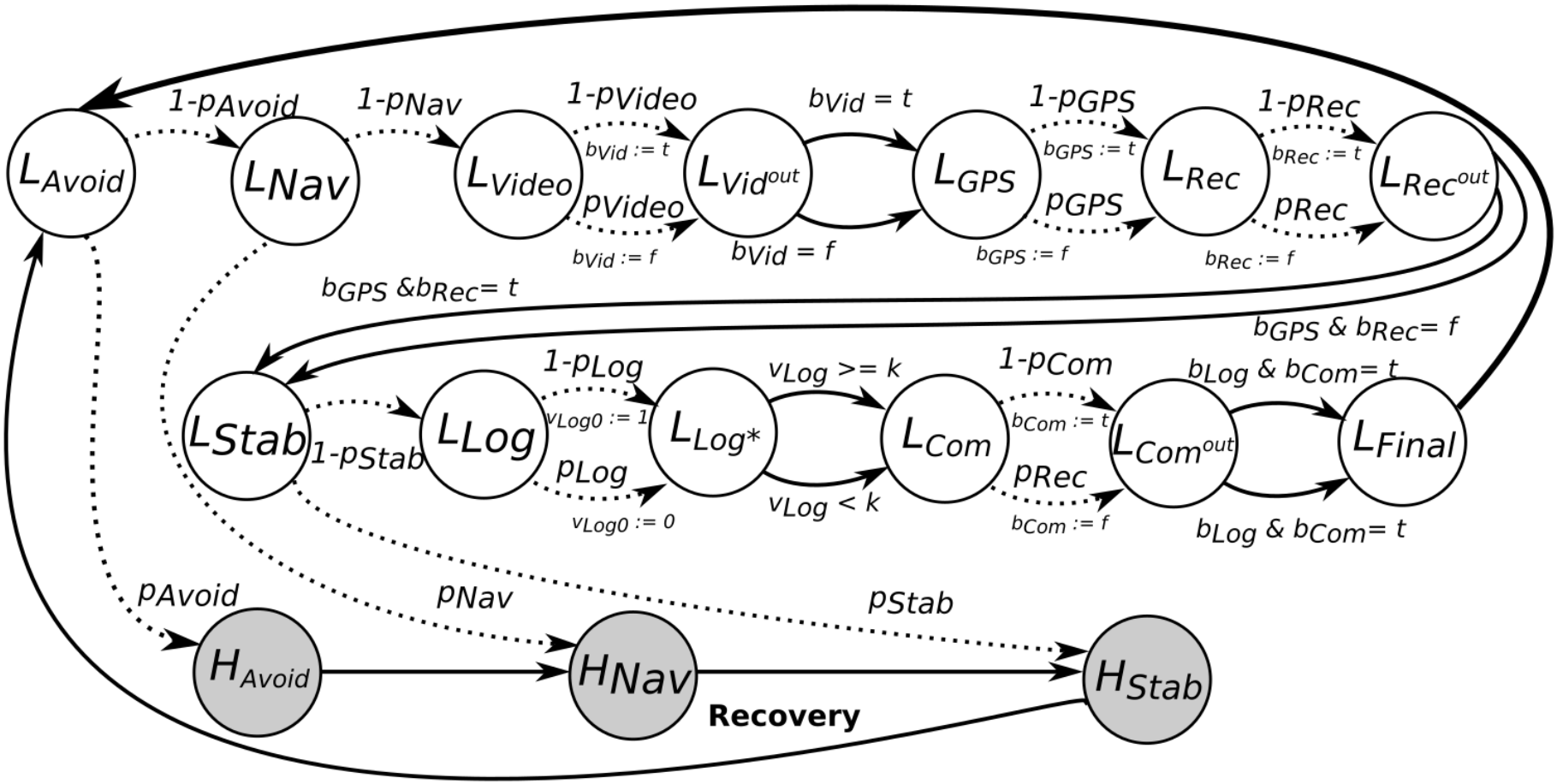


(c) LO output translation

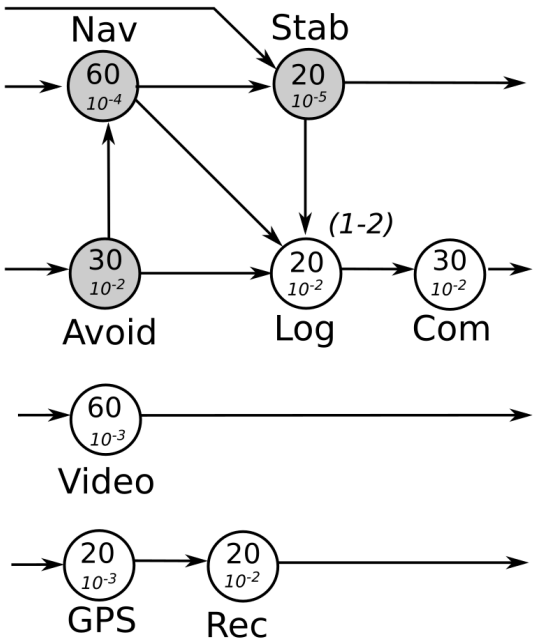


(d) (m-k) firm task translation

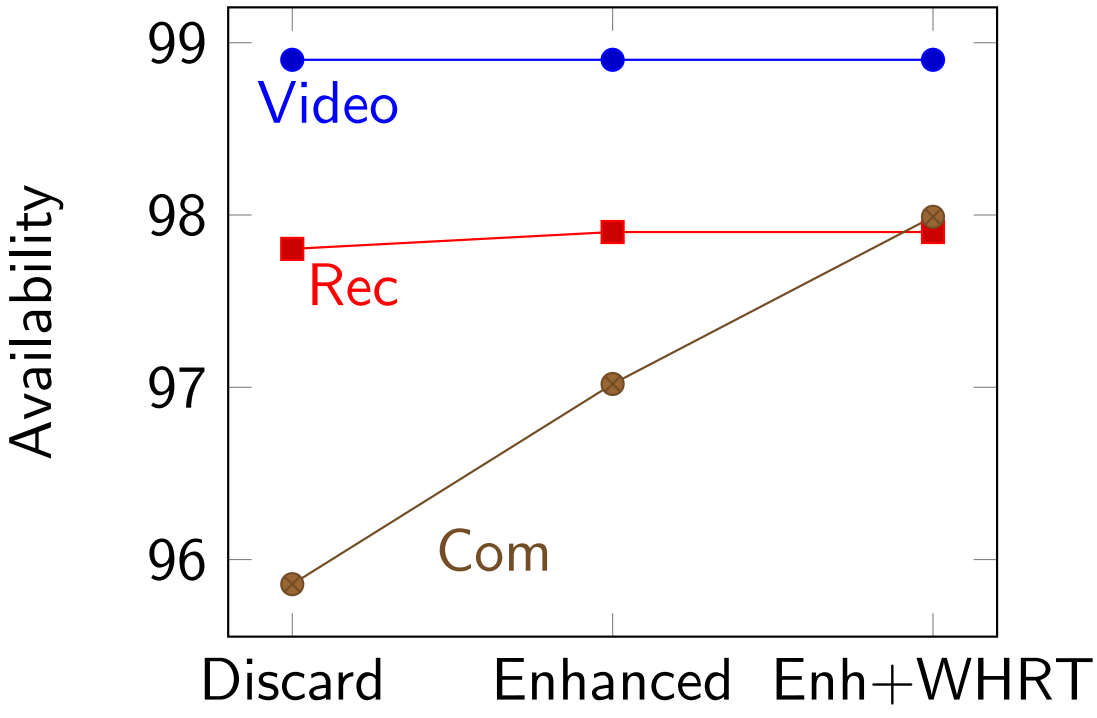
Obtained automaton for our system



Final evaluation of the availability



(a) Architecture



(b) Results

Weakly-hard real-time tasks coupled with our fault propagation model:

- Further improvement in availability: +1% for *Com*.

Conclusion

Defined a method to estimate availability rates

- Defined a formula to compute the availability.
  - Fault model allows to solve this formula.
- Estimate availability thanks simulations of the system.
  - Translation rules to obtain PRISM automata.

Improved the availability rates of LO services

- Improvements to the Mixed-Criticality model: fault propagation.
- Weakly-hard real-time tasks.
- For critical systems 10^{-5} gains are significant.