Mixed-Criticality Real-Time Systems based on Time-Triggered and Priority-Based Scheduling

Sergio Sáez & Jorge Real
ITI, UPV
Spain

EMC²: Mixed Criticality Applications and Implementation Approaches
HiPEAC 2016 - Prague, January 20th, 2016

Outline

- System model
- Proposed solution
- Time-triggered task behavior and task patterns
- System criticality management
- Multiprocessor execution platforms
- Schedulability analysis
- Conclusions

System model

- Application tasks
  - Periodic tasks with different criticality levels (CL)

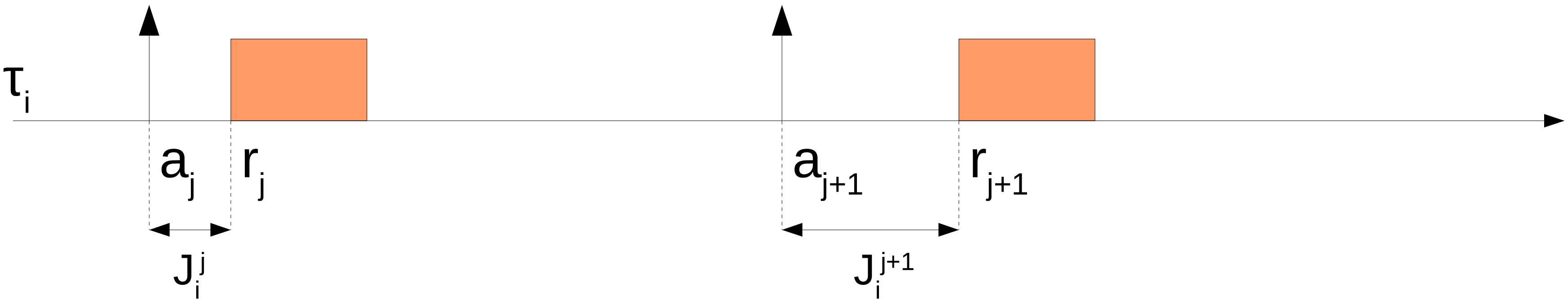
$$\tau_i = \left(CL_i, \vec{C}_i, D_i, T_i \right)$$

  - CL_i: Criticality level of task τ_i
    - The lower the CL_i value, the higher the criticality level.
  - C_i: Worst Case Execution Times → one per CL
    - Different techniques to determine the WCET, or
    - Different behavior for each CL
  - D_i: Deadline
  - T_i: Period or Minimum Interarrival Time

$\left.\right\}$ May also depend on the CL_i

System model

- Application tasks
  - Some tasks have strict release jitter requirements.
  - Unexpected preemptions can also be undesirable.



$$J_i = J_i^{max} - J_i^{min}$$

Absolute jitter

$$J_i^R = max_{\forall j}\left(\left|J_i^{j+1} - J_i^j\right|\right)$$

Relative jitter

  - E.g. Tasks implementing control loops

System model

- Execution platform
  - Shared memory multiprocessor
  - Global Priority-Based scheduler with CPU affinity support
- Application tasks share ...
  - RTOS priority space
  - Tasks with the same CL can share the memory address space
  - Logical and physical resources

Goals

- Execute tasks with different CLs in the same execution platform
  - Timing isolation w.r.t. lower criticality tasks.
  - Support for different MC priority assignment schemes.
- Guarantee the system feasibility in all criticality levels.
  - Depending on the MC model, MC level transitions could also require to be analyzed ≡ operational mode changes.
- For jitter-sensitive tasks
  - Guarantee maximum relative and absolute release jitter
  - Avoid unnecessary preemptions

Proposed solution

A **hierarchical scheduling** scheme based on:

(1) A Time-Triggered scheduling level (TT)

- Tasks using this level are executed according to a predefined time-triggered plan.
- Tasks' jitter is controlled during the construction of the plan.

(2) A Priority-Based scheduling level (PB)

- Tasks using this level are executed according to their priorities (fixed or dynamic).
- This level is **only** activated when spare time is available at TT level.

Time-triggered tasks

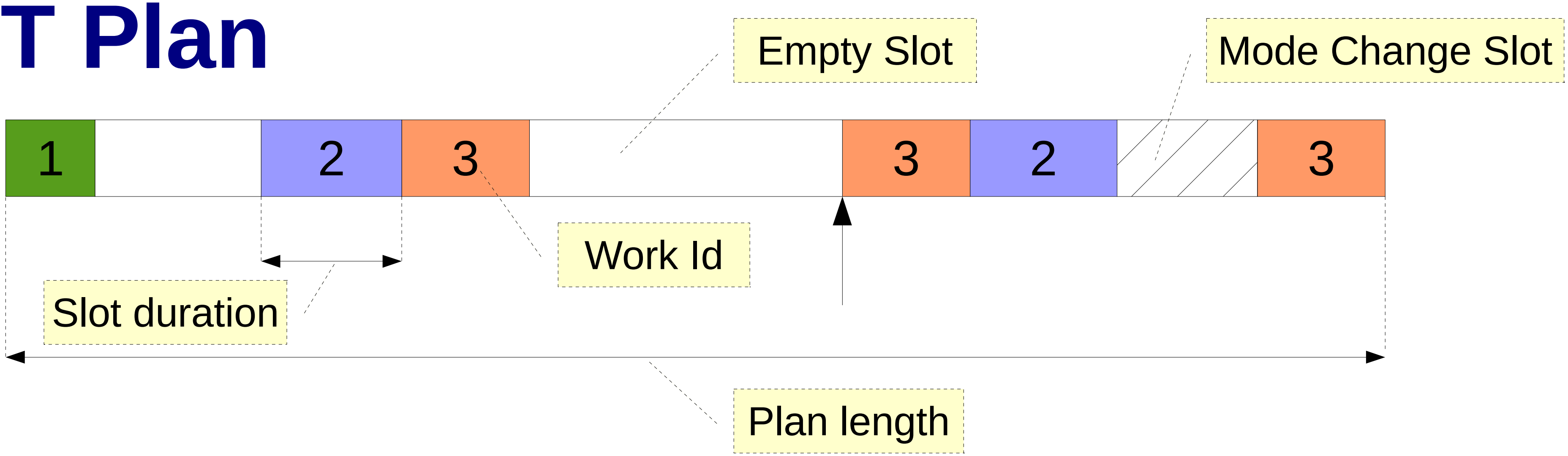
- Tasks are activated following a TT plan.
  - No release jitter is introduced on-line.
    - TT scheduler is simple and predictable.
    - TT tasks cannot be preempted by any PB task.
  - Release jitter w.r.t. original tasks' periods is bounded and perfectly known at run-time.
    - ➔ Corrective actions can be performed within the functional code of the task.
  - The TT scheduler controls that no TT task exceeds is assigned execution time
    - TT tasks do not introduce unexpected interference in the execution of higher criticality priority-based tasks.

Priority-based tasks

- PB tasks are activated periodically.
- PB scheduling provides
  - A flexible concurrent model
  - WCRT can be calculated → system feasibility ensured
- PB tasks are executed according to their priorities when no TT task is active.

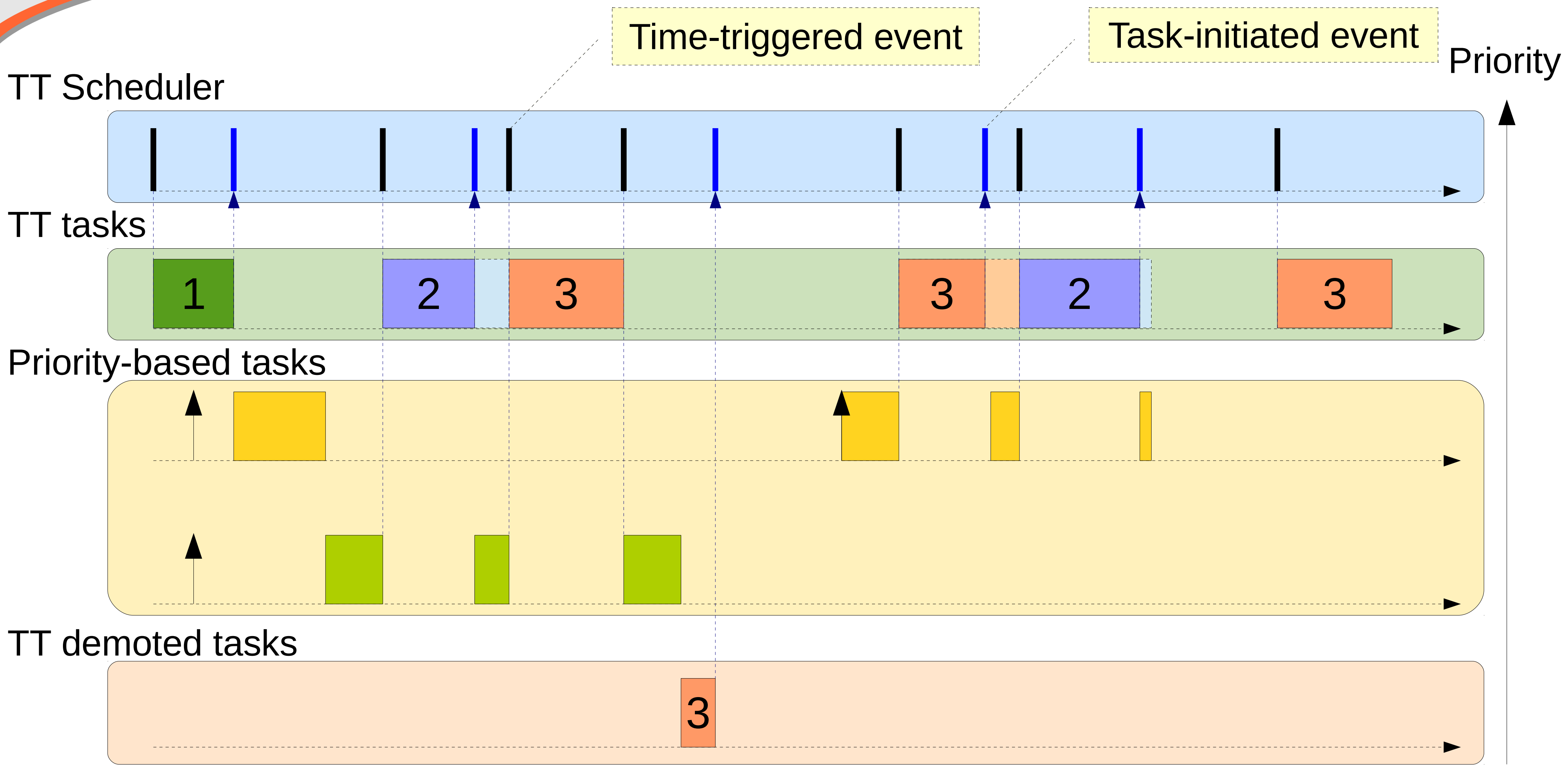
Time-triggered plans

TT Plan



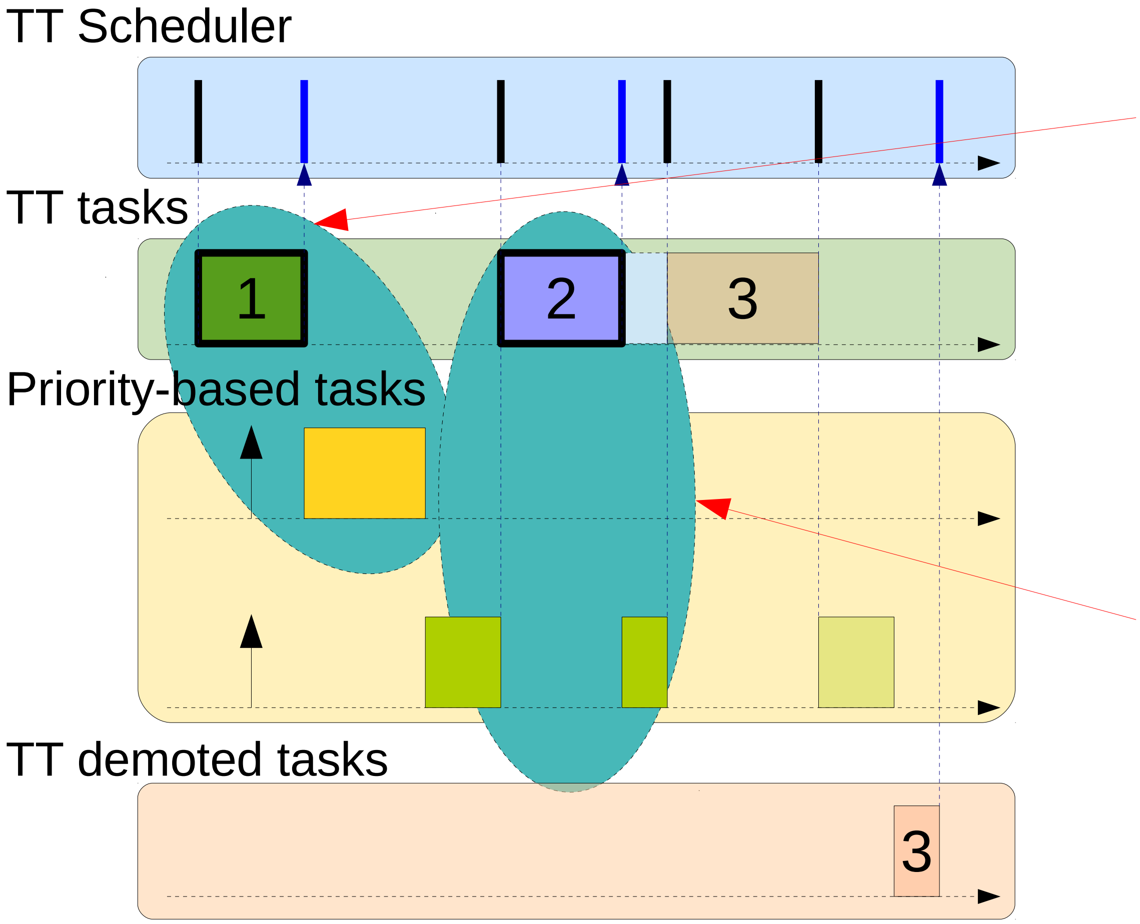
- A Time-Triggered plan is a cyclic sequence of time slots
  - **Regular work** slots to execute jitter-aware tasks.
  - **Empty** slots to allow priority-based tasks to execute.
  - **Mode Change** slots to serve pending mode change requests at predictable instants.

Execution priority layout



Behavior of Time-Triggered tasks

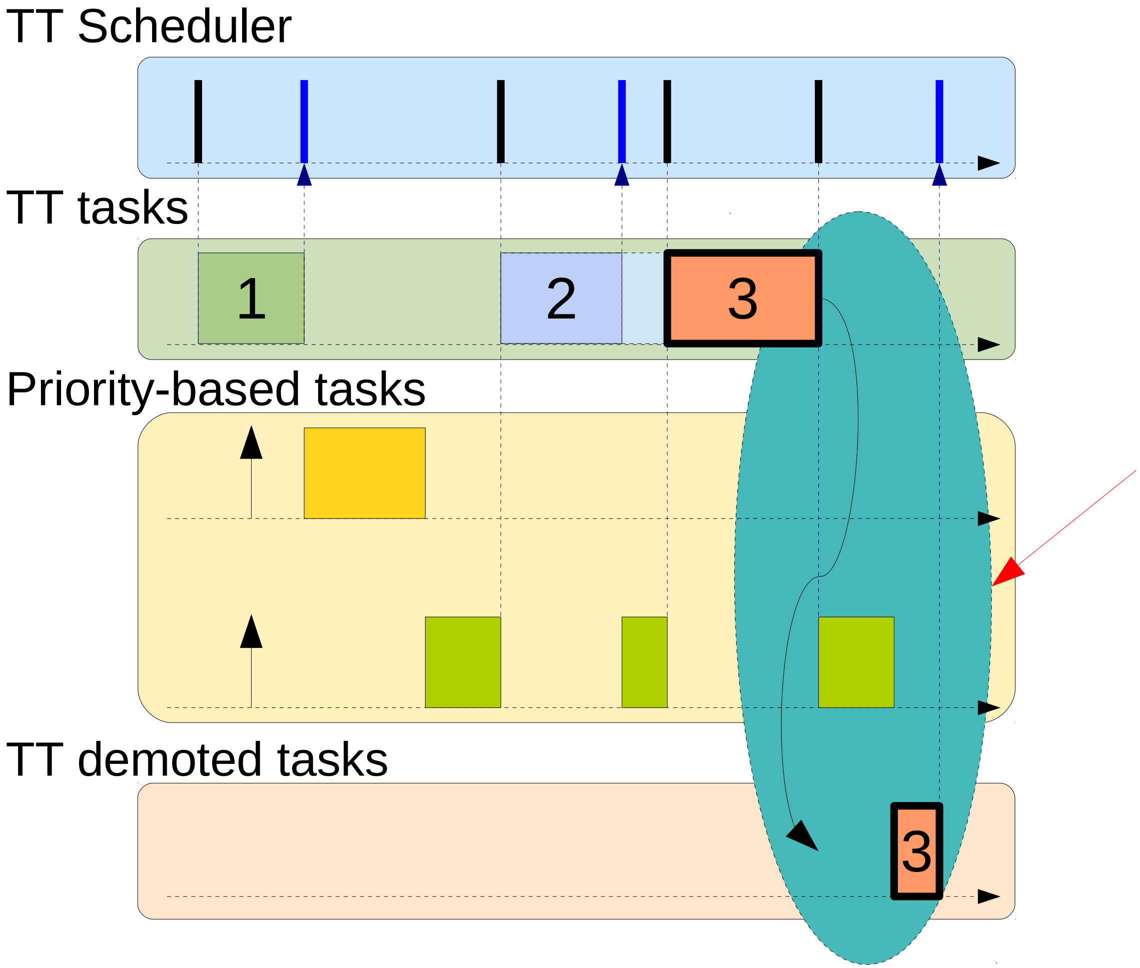
Normal execution



- TT tasks are executed during their respective time slots
- PB tasks are preempted by TT tasks.
- Unused time slots due to early completions are used by PB tasks

Behavior of Time-Triggered tasks

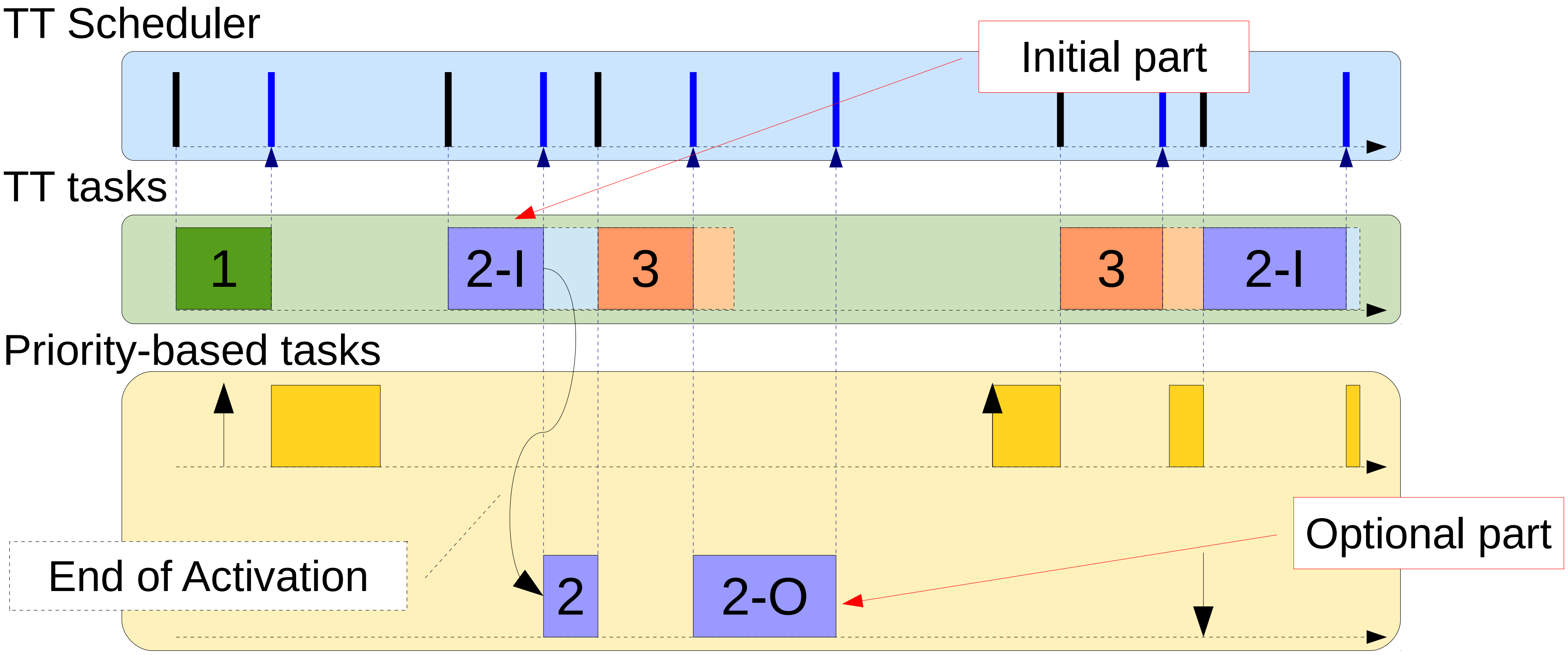
Execution overrun



- TT tasks that exceed its time slot are demoted to a non-disturbing priority.
- PB tasks do not suffer unexpected interference from misbehaving TT tasks.
- System CL can change when a slot overrun is detected.

Time-triggered task patterns

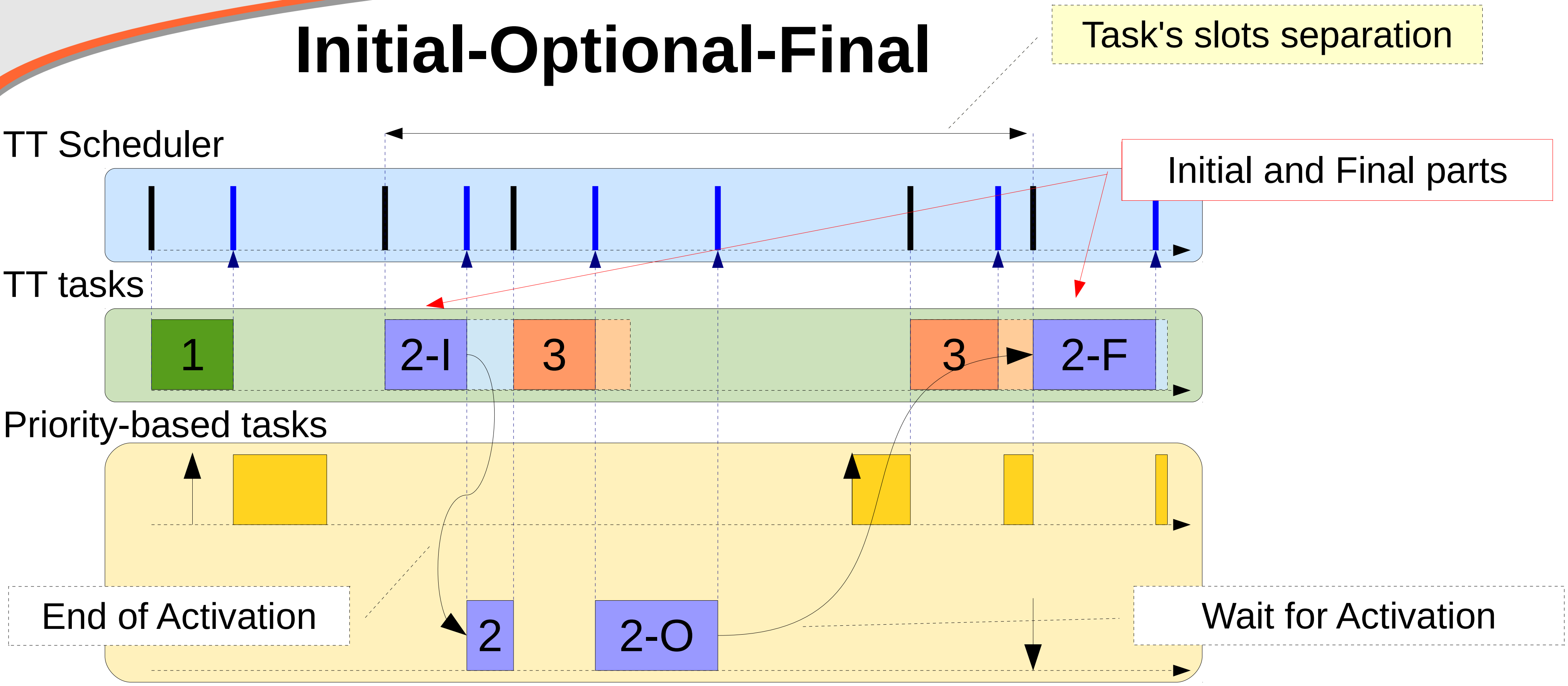
Initial-Optional



- It is equivalent to a sporadic PB tasks activated by a TT task
- It does not require inter-task communication mechanisms.
  → Both parts share the same context.

Time-triggered task patterns

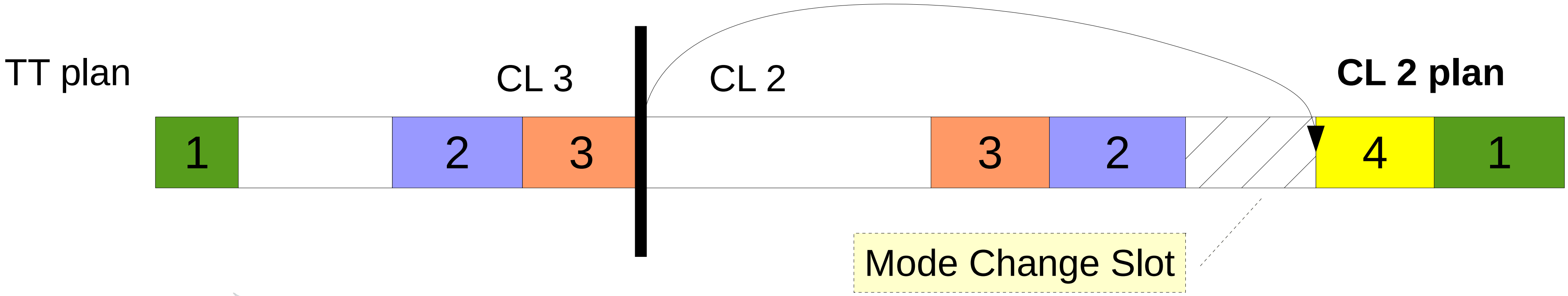
Initial-Optional-Final



- Initial and final parts are jitter-sensitive and mandatory
- Optional part improves control actions, if possible

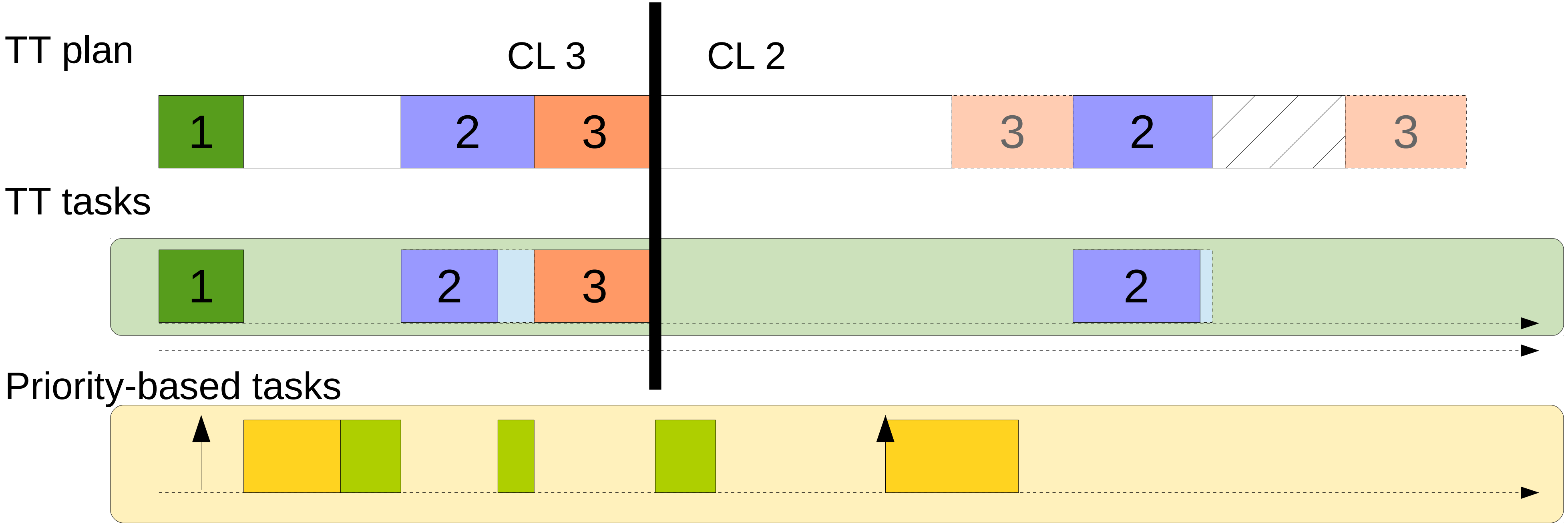
Criticality level changes

- Multiple TT plans and transitional plans
  - One TT plan per system criticality level.
  - Optional TT plans to perform smooth transitions between CL plans.
    - One transient TT plan per each possible CL transition.
  - The new CL plan is not activated until the next Mode Change slot.



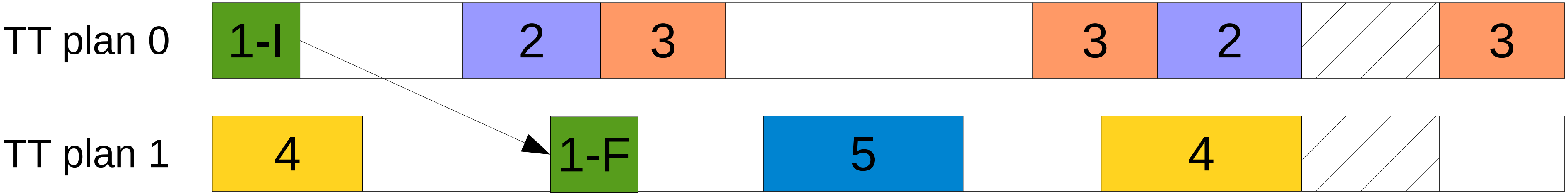
Criticality level changes

- Criticality tagged time slots
  - Each TT task has a criticality level.
  - Time slots with a CL lower than the current System CL are ignored, i.e., treated as *empty slots.*



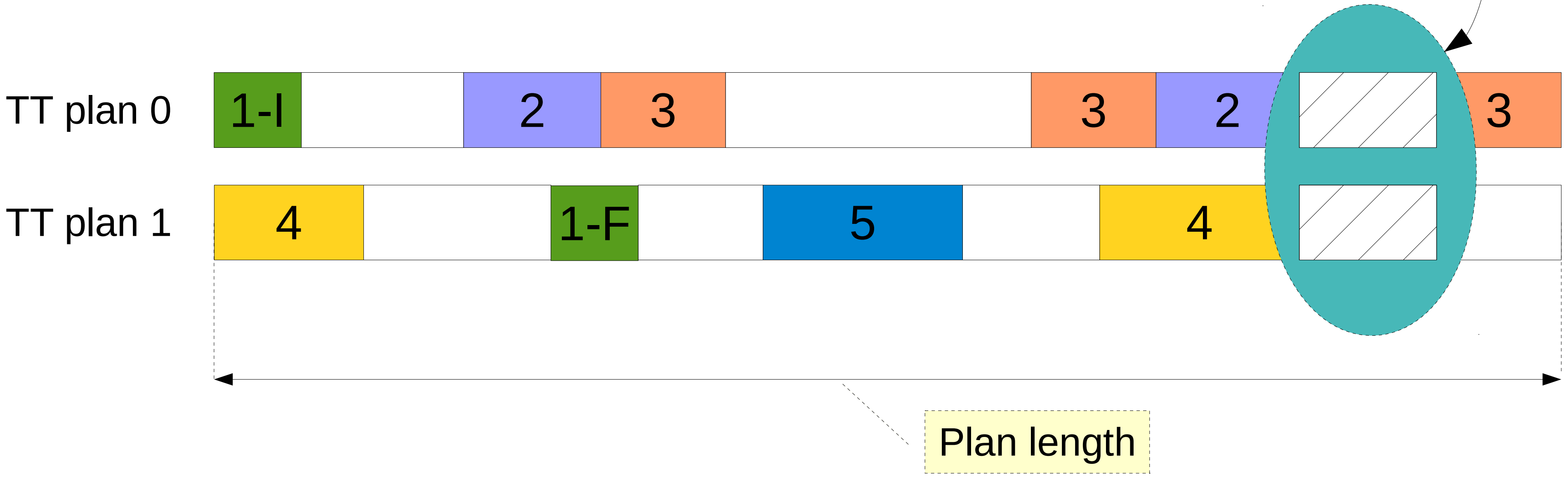
Multiprocessor execution platforms

- One TT plan per processor.
- TT tasks can be partitioned or globally scheduled
  - TT tasks can only migrate from one processor to another to execute different activations.
    - ➔ No time slot preemptions are allowed.
  - Only if necessary to match the required jitter restrictions or if the TT task set cannot be successfully partitioned.

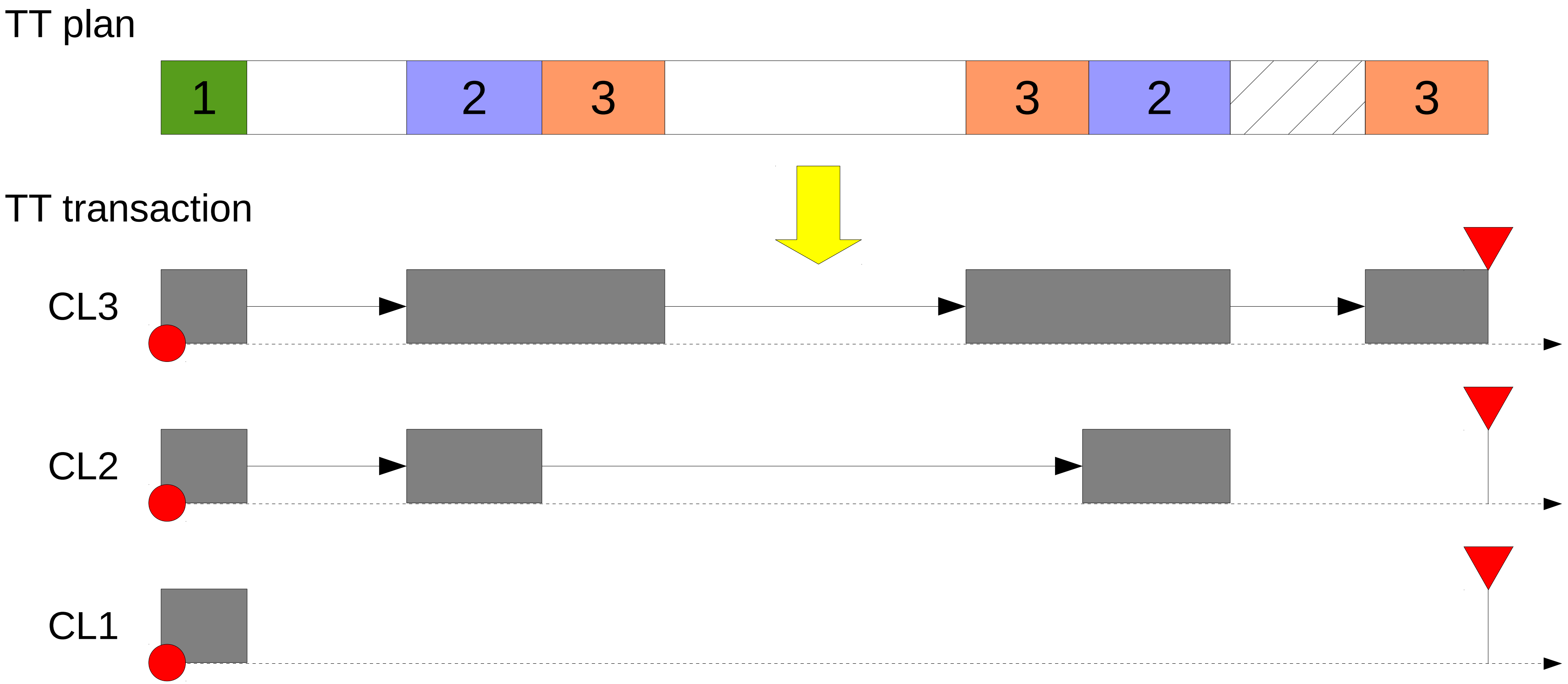


Multiprocessor execution platforms

- TT plan changes have to be coordinated by construction.
  - Mode Change slots has to be synchronized
  - TT plans must have the same length

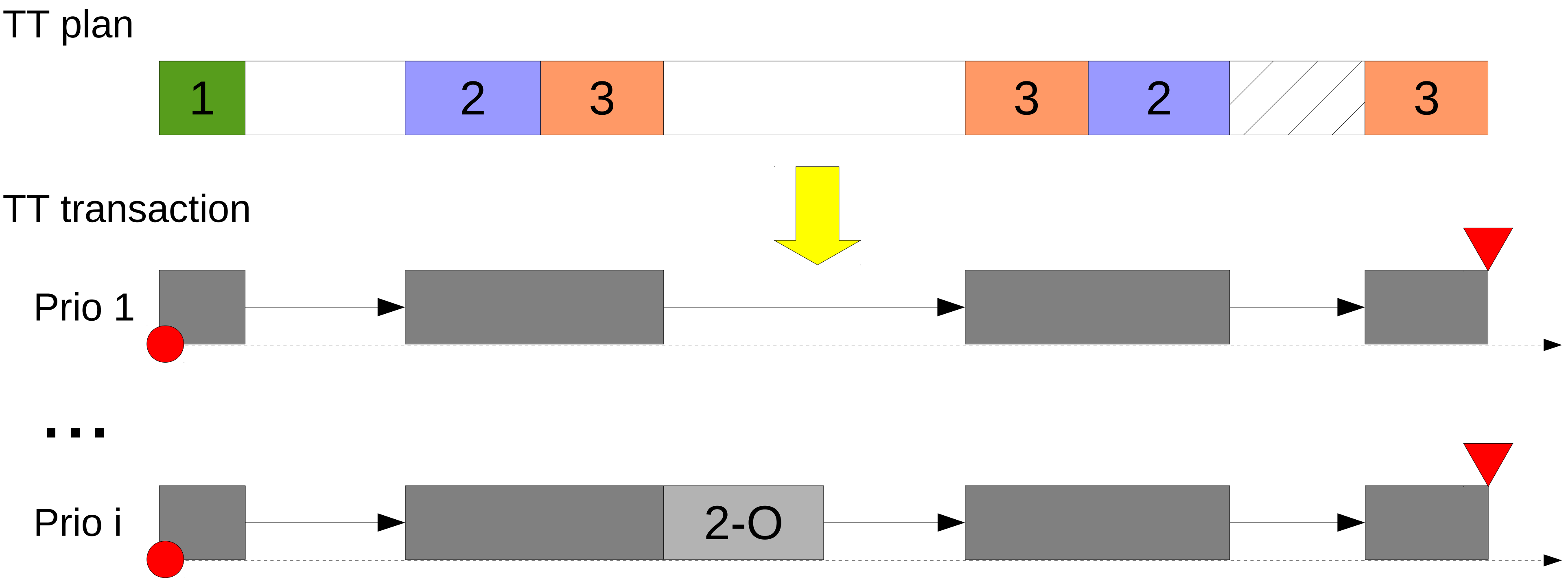


Schedulability analysis



- A TT plan is converted into a periodical transaction for each criticality level.

Schedulability analysis



- Worst case interference due to this transaction can be computed for each priority level.
  - Optional parts only interfere in lower priority levels

Conclusions

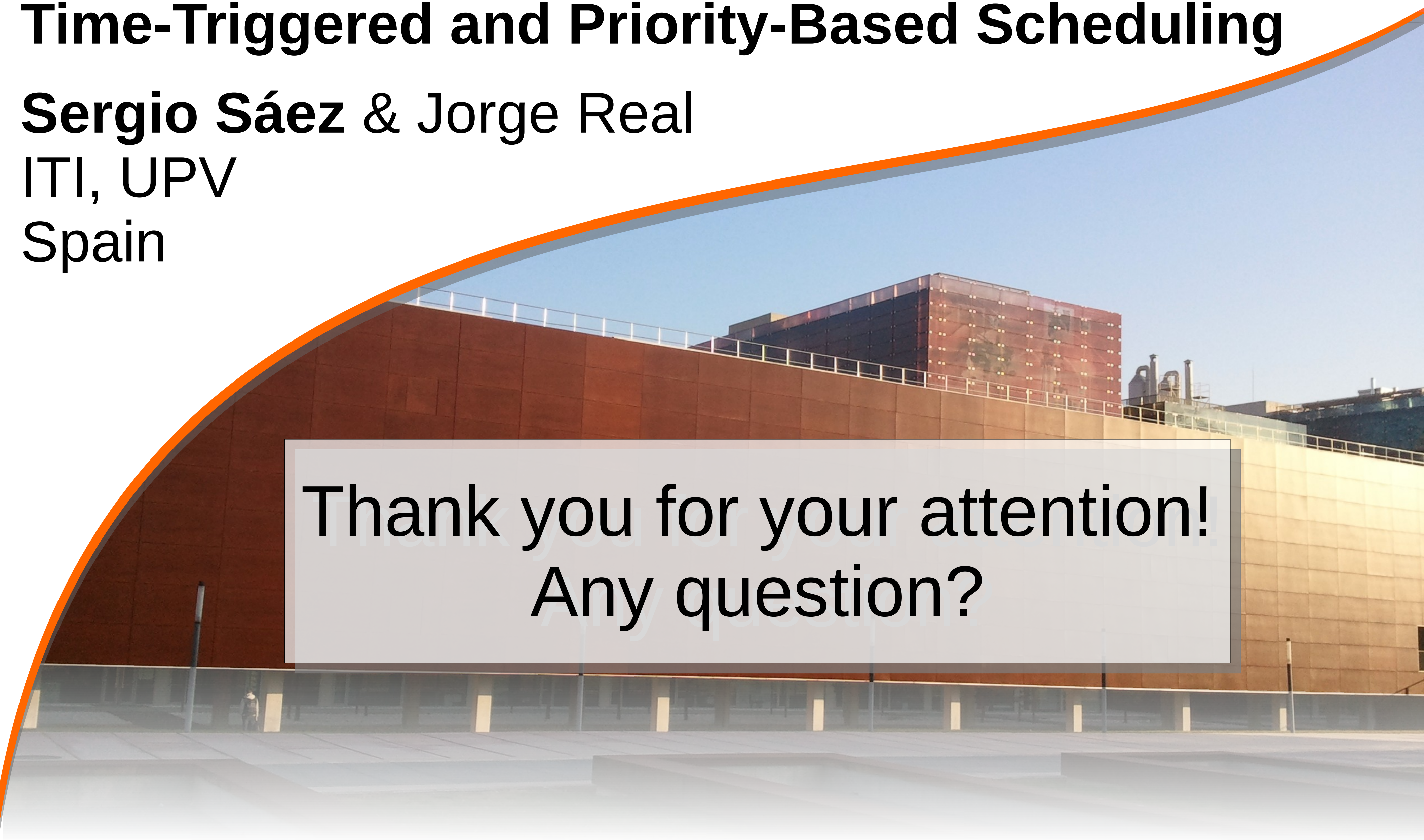
- A hierarchical scheduler is proposed to deal with MC systems that include jitter-sensitive tasks.
- The proposed solution can be implemented in top of a priority-based RTOS.
- The approach does not depend on the priority scheme used to deal with mixed criticalities and can be easily extended to multicore platforms.

Future work

- Incorporate TT plans construction and analysis into the 'art2kitekt' tool chain.

Mixed-Criticality Real-Time Systems based on Time-Triggered and Priority-Based Scheduling

Sergio Sáez & Jorge Real
ITI, UPV
Spain



Thank you for your attention!
Any question?

EMC²: Mixed Criticality Applications and Implementation Approaches
HiPEAC 2016 - Prague, January 20th, 2016