TELECOM Paris

IP PARIS

Contributions to Mixed-Criticality Systems

Laurent Pautet

Safety Critical Systems

Definitions

- A function (or task) is assigned a criticality level according to the severity of a fault on its part
- The criticality level determines the acceptable probability of occurrence of faults (in number per hour)
- It determines the development rules to apply according to the criticality level

Avionic classification

Criticality level	Volume of functions	Consequence	Max # of occurrences
E	5%	None	
D	10%	Minor	10^{-3}/h
C	20%	Major	10^{-5}/h
B	30%	Hazardous	10^{-7}/h
A	35%	Catastrophic	10^{-9}/h

Real-Time Systems
Definitions

A real-time system consists in one or more sub-system
that have to react under specified time requirements
to stimuli produced by the environment

A response after a deadline is invalid
even if the response is logically correct

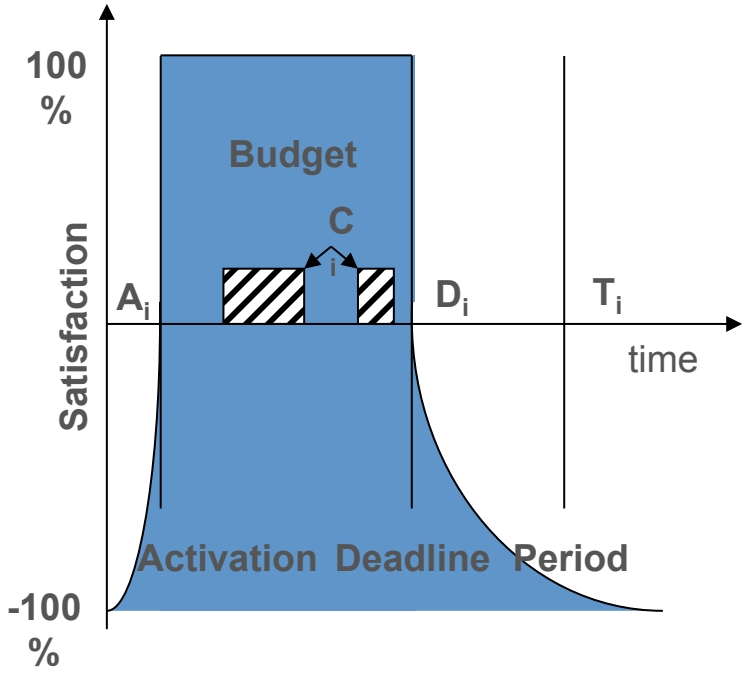
Missing a deadline is considered as a fault

A correct scheduling allocates tasks or functions
to processors such that they respect their deadlines.

Finding a correct scheduling under general assumptions
on the task set is NP-hard.

Real-Time Systems
Task Model

- For a task t_i
  - **T_i : Period.** A task reads periodically or sporadically sensors - exact or minimum delay between task activations (jobs)
  - **A_i : Activation time**. Time when a task can start running. Zero for synchronous tasks, non-zero for dependant tasks for instance
  - **D_i : Deadline**. For deadline implicit tasks, $D_i = T_i$; usually $D_i <= T_i$; when task t2 depends on task t1, typically $D_1 <= A_2$
  - **C_i :** Capacity/**Budget**. Allocated execution time. Usually worst case execution time. But WCET is hard to evaluate (see later)
  - **U_i** : processor **utilisation** of task t_i $U_i = C_i / T_i$

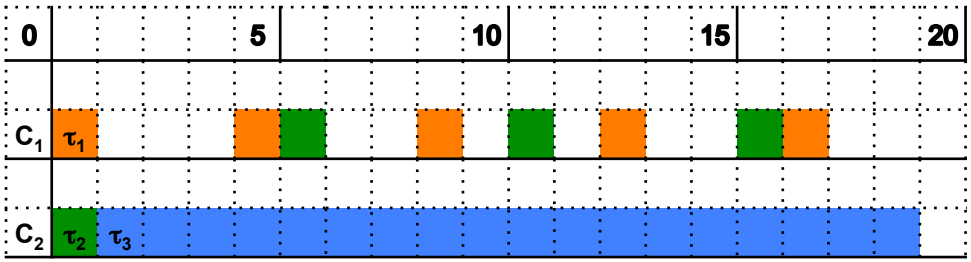


Multiprocessors Scheduling

Two Problems instead of One

- **Uniprocessors** scheduling : time allocation problem (**one** problem)
- **Multiprocessors** scheduling : time **and** space allocation problem (**two** problems)
- Uniprocessor schedulers adapted to multiprocessors are no longer optimal and
- produce **non-intuitive results** (usage reduction -> scheduling becomes invalid)

	Ti	Ci	Ui
τ_1	**4**	**1**	**0.25**
τ_2	5	1	0.20
τ_3	20	18	0.9



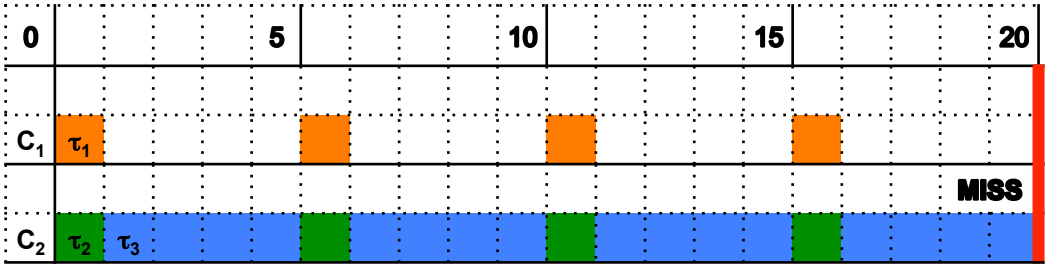
- Multiprocessors scheduling is **very** active research domain …
- As multiprocessor systems are replacing uniprocessor systems (strong industry issue)

Multiprocessors Scheduling

Two Problems instead of One

- **Uniprocessors** scheduling : time allocation problem (**one** problem)
- **Multiprocessors** scheduling : time **and** space allocation problem (**two** problems)
- Uniprocessor schedulers adapted to multiprocessors are no longer optimal ...
- produce **non-intuitive results** (usage reduction -> scheduling becomes invalid)

	Ti	Ci	Ui
τ_1	**5**	**1**	**0.2**
τ_2	5	1	0.20
τ_3	20	18	0.9

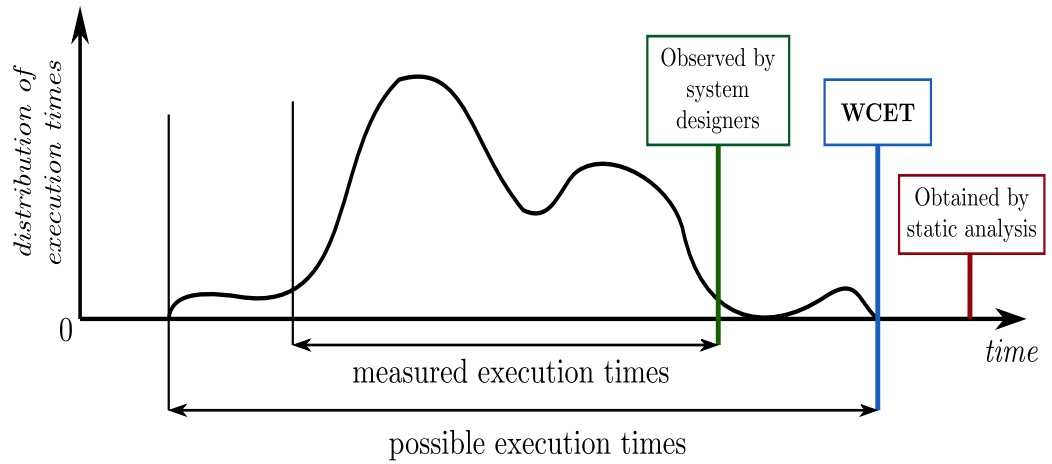


- Multiprocessors scheduling is **very** active research domain …
- As multiprocessor systems are replacing uniprocessor systems (strong industry issue)

Worst Case Execution Time

Different WCET Analysis

- **Measurement analysis**: optimistic since the worst case scenario may not be in the scenario subset. This may lead to **overruns**.
- **Static Analysis**: pessimistic since this worst case scenario may never occur or very rarely. This leads to budget overestimation and system **oversizing**.
- Worst case scenario is even **more difficult to estimate for multi-processors** since interferences between tasks oftenly occur due to shared HW ressources. Multiprocessor systems are replacing uniprocessor systems.



Mixed Criticality Systems (MCS)

General Principles

- **Objective** : Execute / mix functions of different criticality, such as survival functions (high critical - HI) and mission functions (low critical - LO).
- Traditional critical systems use federated architectures allocating functions of same criticality to partitions / machines dedicated to a given criticality level. This results in **oversizing** the architecture (wasted resources).
- **Approach**: Allocate to HI functions less ressources than the worst case scenario (optimistic) would require. Execute LO functions as required.
- When ressources are missing for HI functions (a worst case scenario really occurs), LO functions are degraded or stopped as their failure rate is greater and their ressources are reallocated to HI functions.
- This mode change must be seamless as the system cannot afford to restart.

Mixed Critical Systems
Task Model

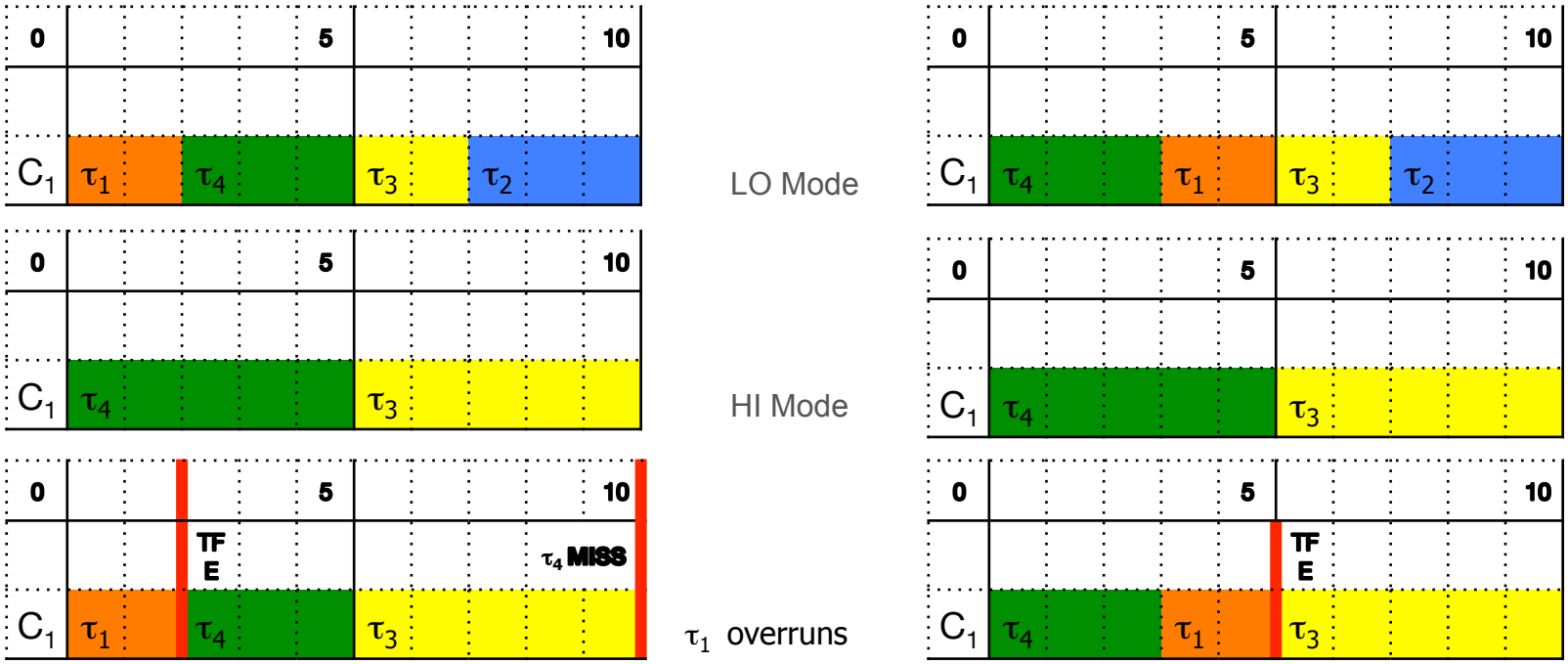
- Set of HI (high criticality) and LO (low criticality) tasks
- Timing budgets for tasks with $C_i(LO) < C_i(HI)$
  - $C_i(LO)$: optimistic timing budget (e.g. measurement analysis)
  - $C_i(HI)$: pessimistic timing budget for HI tasks (e.g. static analysis - WCET).
- Execution modes:
  - LO criticality mode (initial mode):
    Both HI and LO tasks are executed with their LO timing budgets
  - HI criticality mode:
    only HI tasks are executed with their HI budgets. LO tasks are degraded or discarded. **Their budgets are reallocated to HI tasks for their HI budgets**.
- Execution mode change (from LO mode to Hi mode)
  - When a HI or LO task overruns its budget, a Timing Failure Event (TFE) occurs ; the system triggers a seamless mode change to HI mode

Mode Change Issue

Three Problems instead of One

- Finding correct scheduling for LO or HI mode is complex
- Finding HI and LO schedulings such that they enforce correct mode changes is even more complex
- The expensive solution requires 2 cores ((2+3+5+5)/10)

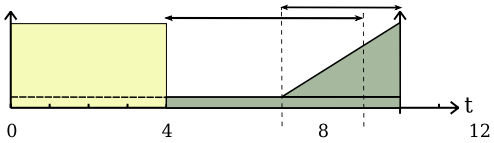
	Ti	CL	C(LO)	C(HI)
τ_1	10	LO	2	0
τ_2	10	LO	3	0
τ_3	10	HI	2	5
τ_4	10	HI	3	5



Contributions to Mixed Criticality Systems

Energy Efficiency

- Objectives : Optimize idle times to activate the most energy efficient states
  Hypothesis : energy consumption is a linear function of execution time
- Problems :
  - The deeper the low power state, the lower the power consumption, but the longer the transition time to return to the active state (Break Event Time - BET)
  - Changing state also requires energy



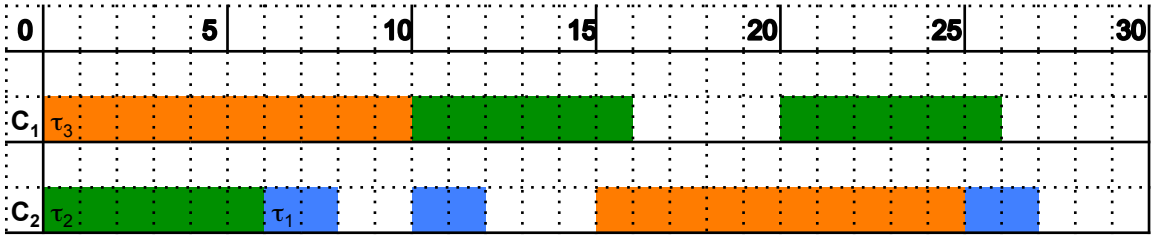
- V. Legout, M. Jan, L. Pautet: Scheduling algorithms to reduce the static energy consumption of real-time systems. Real Time Syst. (2015)
  - Offline : generate static scheduling from a MILP which minimizes energy consumption
  - Online : use MC to increase idle times, may induce deadline miss & mode change

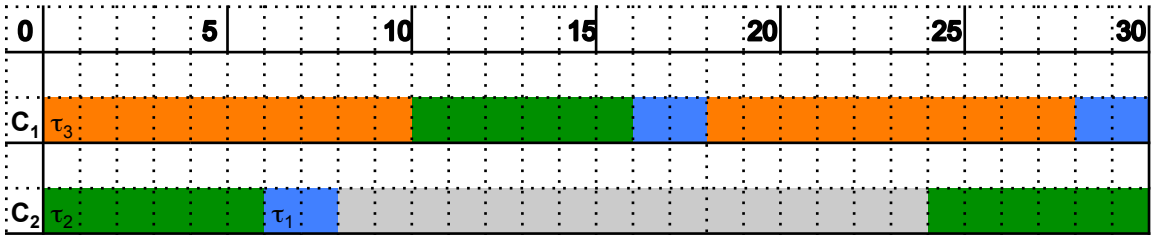
Contributions to Mixed Criticality Systems

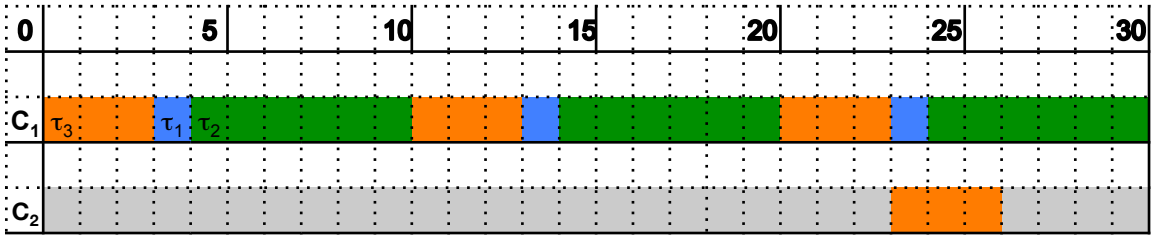
Energy Efficiency

	Ti	CL	Ci(HI)	Ci(LO)
τ_1	10	LO	2	1
τ_2	10	HI	6	6
τ_3	15	LO	10	6

- If BET=5, a traditional scheduling fails to reach low-power state
- An energy aware scheduling can suspend processor for 16-5u
- A MC energy aware scheduling can suspend processor for 27-5u
- If the first job of τ3 overruns in this particular , a deadline miss occurs







Contributions to Mixed Criticality Systems

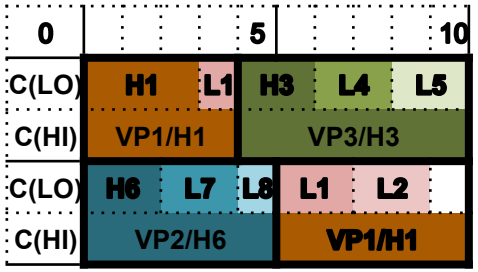
Multiprocessor Efficiency

- Objectives :
  - To guarantee mode change, a simple but expensive solution consists in scheduling HI tasks with their HI budget and LO tasks with their LO budget
  - Minimize # of required processors (thus, reduce CPU, energy and cost)
- Problems :
  - Few multiprocessors schedulers optimise or minimise # of processors
  - Difficult to design schedulers that guarantee safe mode change

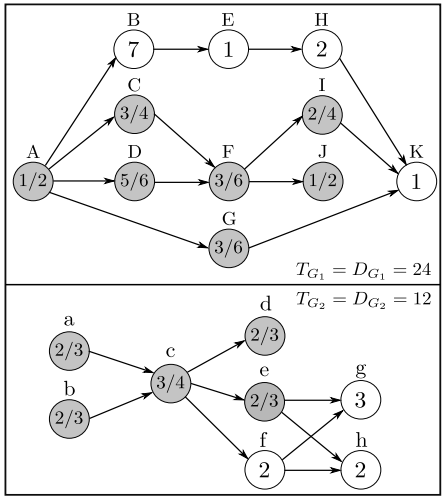
Contributions to Mixed Criticality Systems

Multiprocessor Efficiency

- R. Gratia, T. Robert, L. Pautet: Scheduling of MCS with RUN. ETFA (2015)
- Based on RUN scheduler : only optimal & implementable multiprocessors scheduler ; globally schedule virtual processors which schedule statically allocated tasks



- Contribution GMC-RUN : define a HI task as a virtual processor and use its slacktime Ci(HI)-Ci(LO) to schedule its allocated LO tasks (this allocation problem is NP-hard)
- R. Medina, E. Borde, L. Pautet: Generalized MC Static Scheduling for Periodic Directed Acyclic Graphs on Multi-Core Processors. IEEE Trans. Computers (2021)
- Based on a time-triggered approach (industrial standards): predictive table-driven schedulers
- Contribution GMC-DAG : build scheduling tables by enforcing a safe mode change and an as-late-as-possible strategy to allow both very good acceptance ratio and efficient implementation



Contributions to Mixed Criticality Systems

Kernel Implementation

- Objectives :
  - Efficiently implement MC schedulers (here GMC-DAG) on open-source real-time kernels supporting industrial standards
- Problems :
  - In most schedulability analyses, the mode change cost is overlooked despite that it clearly incurs complex synchronization.
  - The mode transition may induce an asynchronous rescheduling operation across all processors.
  - This rescheduling operation may happen on a processor which is already involved in a scheduling operation.
  - These rescheduling operations have to be managed simultaneously making it more complex to implement with non-atomic mutually exclusive primitives.

Contributions to Mixed Criticality Systems

Kernel Implementation

- L. Pautet, T. Robert, S. Tardieu: Litmus-RT plugins for global static scheduling of mixed criticality systems. J. Syst. Archit. (2021)
  - Add kernel services to make MC schedulers a reality. They have been implemented in LITMUS-RT, a well-known prototype of RT Linux kernel.
  - First service: handle properly instantaneous task migrations and cyclic instantaneous migrations to avoid deadlocks.

    On LITMUS core c1 preempt t1 and let it migrate to c2 only when t2 has migrated from c2; core c2 preempt t2 and let it migrate to c1 only when t1 has migrated from c1
  - Second service: a safe but efficient implementation of mode changes that can handle multiple simultaneous budget overruns.
  - Next step : implement GMC-DAG on top of RTEMS, open source real-time platform, well-known in the industry.

Mixed Criticality Systems
Conclusions

- Objectives of resource reduction
  - Resource : CPU, energy, cost
  - Domain : HI/LO critical systems or autonomous systems such as drones
- Perspectives
  - Efficient schedulers for more general task models
  - Provide decent implementation on well-known RT kernels