POLITECNICO
MILANO 1863

DIPARTIMENTO DI ELETTRONICA,
INFORMAZIONE E BIOINGEGNERIA

HEAP LAB

8th International Workshop on Mixed Criticality Systems
@ Real Time Systems Symposium (RTSS 2020)

Fault-Tolerant Real-Time Systems: Challenges and Future Directions

Invited Talk

Federico Reghenzani
<federico.reghenzani@polimi.it>

Rationale

Fault-Tolerant and Real-Time systems

- What is the current state-of-the-art of software fault-tolerant techniques when used in real-time systems?
- How are the real-time and fault-tolerant problems linked?
- How can mixed-criticality play a role in this context?
- What are the current challenges and possible future research directions?

With the contributions of:

- Prof. William Fornaciari, Politecnico di Milano, Italy
- Prof. Zhishan Guo, University of Central Florida, US

Real-Time Systems

Definition

- A (hard) real-time system is a system that must satisfy logical and temporal correctness.

Task model

$$\tau_i = (C_i, T_i, D_i)$$

Worst-Case Execution Time — Period — Deadline

Mixed-Criticality Systems

MC Task Model

$$\tau_i = (\overline{C}_i, T_i, D_i, L_i)$$

Vector of WCETs — Criticality Level

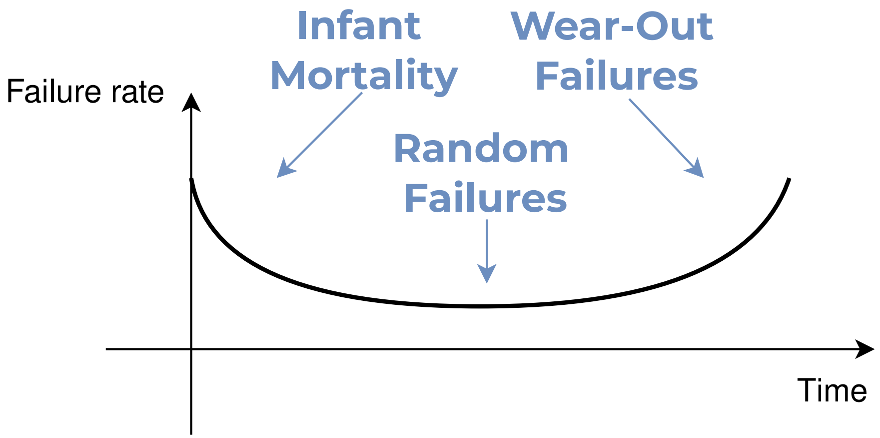
- Each criticality level corresponds to a certification requirement
  - e.g. DAL A, DAL B, …

System mode switch

- When a task overruns one of its WCET, we say that the system "change mode", and it usually degrades the performance of lower criticality tasks

Fault classification

- Permanent Faults
  - They irremediably damage the device, that must be repaired



- Transient Faults
  - Temporary faults, usually modeled with Single Event Upset (SEU)
- Intermittent Faults
  - They appear as bursts of transient faults
  - Caused by environmental effects

    e.g., High-Intensity Radiated Field (HIRF)

Fault sources

Let's focus on Transient Faults

- Main causes:
  - High-energy Particles (α+γ) (e.g., Cosmic Rays)
  - Chip Package Impurities (α)
  - Reflow Soldering Process (α+γ)

Hardware shielding is easy for α but not for γ rays

This is very problematic for space applications

We can improve the manufacturing process, but we cannot shield the system from itself

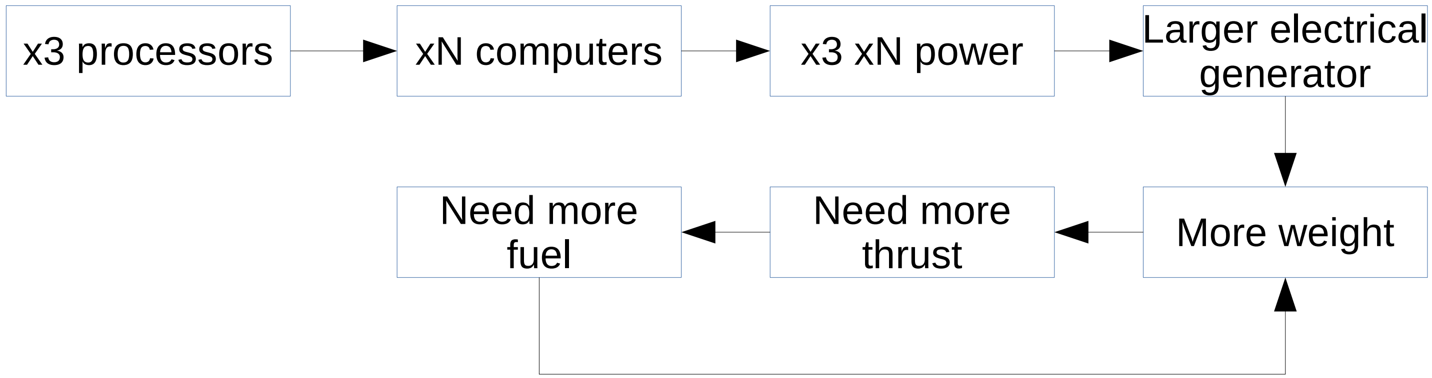
Fault-Tolerant Systems

Hardware fault-tolerance

- The replication of hardware components is the traditional way to achieve fault-tolerance requirements via redundancy
  - ➔ e.g., Voting, Fail-over systems, …

- However, hardware fault-tolerance has cascade effects on development and production costs, weight, energy consumption, thermal dissipation, etc.
  - ➔ Especially problematic for aerospace applications

    (e.g. a LEO transfer costs 3k – 50k$/kg)



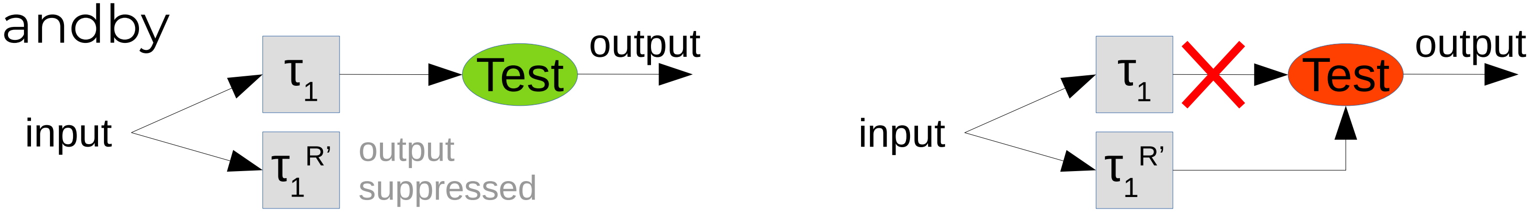
Software FT – Space Redundancy

N-Modular Redundancy

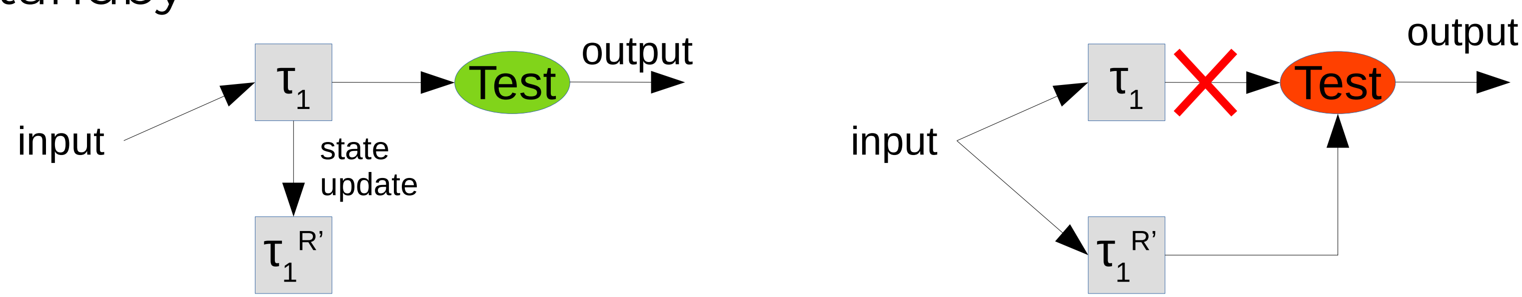
- Similar to hardware replication
- Each task is replicated N times (possibly on different processors) and a **voting** system is applied to their outputs
- It increases by x(N-1) times the system utilization

Reconfigurable Duplication

- Hot-Standby



- Cold-Standby



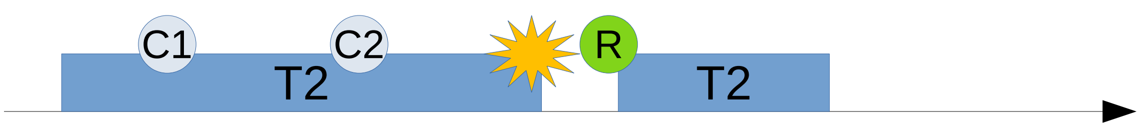
Software FT – Time Redundancy

Re-Execution

- At the end of a job, the job is restarted if an error has occurred
- The job can be restarted multiple times if the failure probability requirement requires so

Checkpoint/Restart

- Periodic checkpoints save the state of the job, in order to resume it in case of fault is detected
- Proper tuning of the checkpoint rate is essential



Many other techniques…

- Forwards Error Recovery, Recovery blocks….

The research question

How to guarantee **fault-tolerance requirements** while maintaining the utilization at acceptable levels to guarantee **hard real-time requirements**?

State-of-the-Art

Previous works

- Fault-tolerance in real-time systems is not a new topic, the first papers appeared at the beginning of ‘90

- In the last 30 years:
  - Many papers on fault-tolerant distributed real-time systems
  - However, not many papers considered the transient fault tolerance techniques in the context of “traditional” real-time systems

- A few papers on mixed-criticality, but very preliminary works

The interest is increasing

Technology

- Transistors are getting smaller and smaller and then more susceptible to bit flips
- The incresing use of reconfigurable architectures (FPGA) is even more problematic

The interest in Commercial Off-The-Shelf (COTS) devices for aerospace and automotive is increasing

- The switch to COTS is in the critical path for technology achievements for space agencies
  - Ref. ESA's technology strategy 2019

- Software fault-tolerance may be the only way to satisfy the failure requirements in COTS

Fault-tolerance and real-time crosslinks

Impact of fault-tolerant on real-time requirements

- The fault-tolerance requirement to execute more than one time a job (re-execution), the N-MR tasks, the checkpoints, etc. increase the system utilization

Impact of real-time requirements on fault-tolerance

- The larger the execution time, the larger a job is exposed to transient faults in the processor and memory
- The larger the waiting time, the larger a job is exposed to transient faults in the input memory

Possible research directions

Can Mixed-Criticality scheduling be exploited for FT?

- Example with re-execution:
  - Fault probability in a given job (simplified): 10^{-4}/h

Task	Criticality	Failure Requirement	Nr. re-execution	WCET
T_1	LO	10^{-3}/h	0	C_1
T_2	MI	10^{-6}/h	1	$\{C_2, 2C_2\}$
T_3	HI	10^{-9}/h	2	$\{C_3, 2C_3, 3C_3\}$

- In such a setup, system mode switch depends on faults not on the execution time → the probability of mode-switch is known

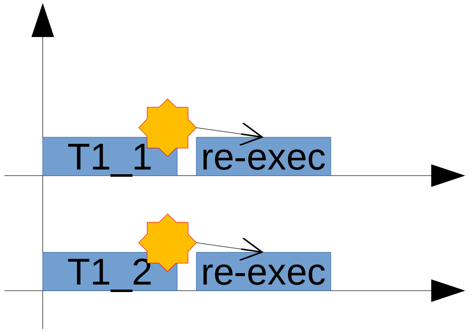
Possible research directions

DVFS and fault-probabilities

- Changing the processor speed modifies the amount of time a task is exposed to faults
- Increases the processor speed decreases the exposure time, but it increases the permanent faults rate due to thermal effects

Composition of techniques

- Can the combination of techniques (e.g., N-MR + re-execution) improve the schedulability while guaranteeing the failure requirements?

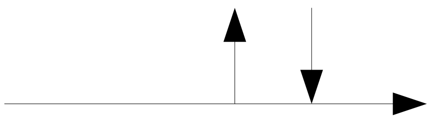


Possible research directions

Sporadic tasks

- Sporadic tasks are associated to "on-demand functions"
  - The probability of failure requirement is expressed as Probabilistic of Failure per Demand and not Probability of Failure per Hour:

    e.g., PFD = 10^{-3} / job
- Does this change the way failure and real-time requirements interact?



OS & Scheduler

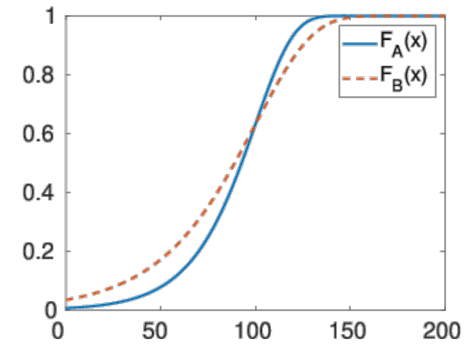
- How to make OS (including scheduler) resilient to faults?
  - Can we apply the same techniques (N-MR, re-execution, …) for OS tasks?

Who guards the guards?

Possible research directions

Probabilistic (worst-case) execution time

- pWCET or pET may provide a statistical characterization of the fault probability less pessimistic compared to the WCET



What about malicious faults and security?

- Can attacks invalidate real-time requirements?
  - e.g., can a DoS attack make the utilization > 1?
  - What about side-channel attacks exploiting timing information?
- How security countermeasures impact real-time requirements?

Conclusions

Thanks for your attention

Questions & Discussion

http://heaplab.deib.polimi.it/wmc2020/