SMILIES

Politecnico di Torino

Department of Control and Computer Engineering

DAUIN

DDECS

DESIGN & DIAGNOSTICS of ELECTRONIC CIRCUITS & SYSTEMS

April 27-29, 2026, Bratislava, Slovakia

Experimental Analysis of FreeRTOS Dependability through Targeted Fault Injection Campaigns

Luca Mannella, Stefano Di Carlo, and Alessandro Savino

29th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems

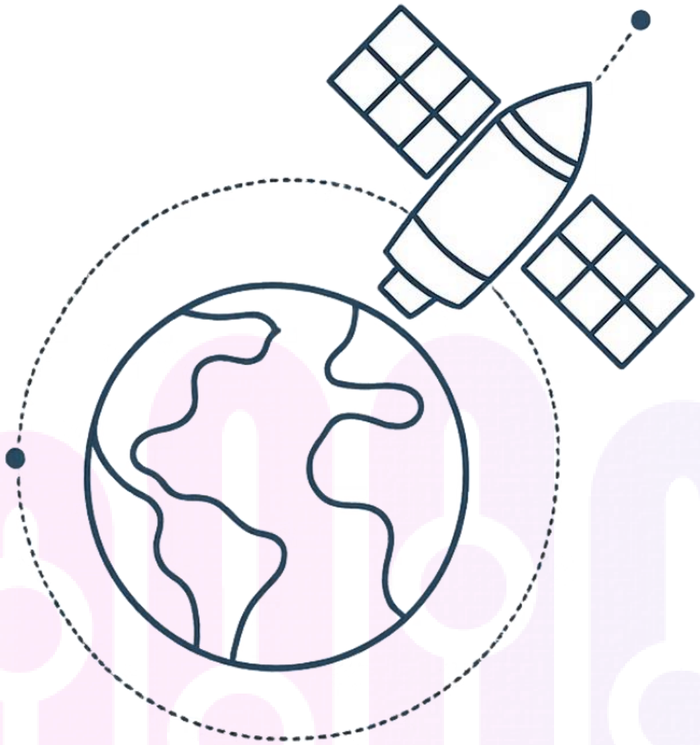
April 27, 2026

Bratislava, Slovakia,

Motivation

The importance of analyzing RTOSes dependability

- RTOSes underpin safety-critical systems in automotive, aerospace, and satellite domains.
- Ionizing radiation can cause **Single Event Upsets (SEUs)**
- As device geometries shrink, radiation vulnerability increases even at ground level.
- In long-duration missions, device aging can progressively degrade components, eventually causing permanent faults.
- Both faults can propagate to kernel-visible state compromising determinism and availability.



Research contributions

1

KRONOS Framework

Novel software-based, non-intrusive post-propagation Fault Injection framework for FreeRTOS
— no specialized hardware required.

2

Extensive FI Campaign

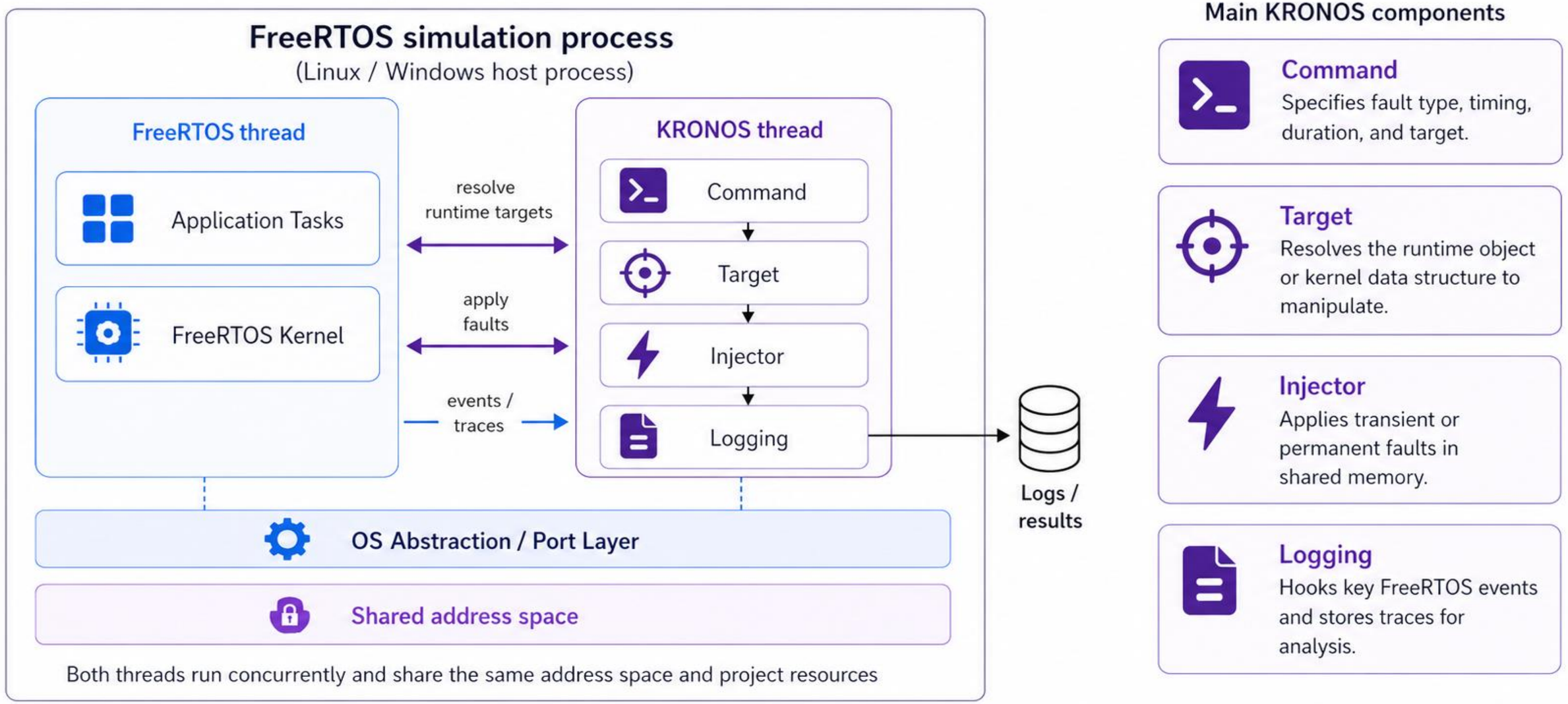
83,916 injections across all core FreeRTOS kernel components, covering both transient and permanent fault models.

3

Vulnerability Analysis

Identifying the most critical kernel data structures for early-stage safety assessment.

KRONOS Architecture



Fault Models

⚠ We are not modeling the physical fault generation; *we assume the fault has already propagated to OS-visible kernel state*

Transient Faults

- One-time bit flip at the chosen time and memory location.
  - SEU-like behavior
- No additional enforcement after the flip.
- Implemented applying a XOR operation, with a specific mask, to the target

Permanent Faults

- Initial set to 0 or 1 depending on the stuck-at to enforce at the chosen location, then fault persistence is enforced.
- FreeRTOS kernel sources are pre-patched so that, after relevant variable updates, a helper re-applies the faulty value.
  - Prevents the fault from disappearing when the kernel overwrites that variable later.
  - Models stuck-at behavior over the rest of the execution.

Experimental Setup

Injection Targets

1. Global Variables

   Scheduler state, tick counters, priority flags **(12 targets)**

2. Pointers

   pxCurrentTCB, delayed/timer list pointers **(16 targets)**

3. Lists

   Ready, delayed, pending, suspended, timer lists (**20 targets**)

4. Current TCB Fields

   Priority, stack pointer, notification state **(15 targets)**

Campaign Parameters

- Benchmarks: **5 TACLeBench Apps**
  - (i) SHA, (ii) FFT, (iii) CUBIC, (iv) HUFF_DEC, (v) ADPCM_ENC
- **Confidence**: 99%, margin 5%
  - Using the formula from *Leveugle et al.* (DOI: 10.1109/DATE.2009.5090716)
  - FI per location: 666
- **Total Injections**: 83,916
  - 666 * 63 targets * 2 fault models
- **Tick rate**: 1 kHz
- 7 priorities levels

Experiment outcome Classes

Standard

- Benign:
  Correct result, on time
- Delay:
  Correct result, deadline missed
- SDC:
  Wrong result, no error raised
  (i.e, silent error)
- Hang:
  No completion (deadlock/livelock)
  i.e., golden run time x 3 (i.e., 300%)
- Crash:
  Abrupt termination

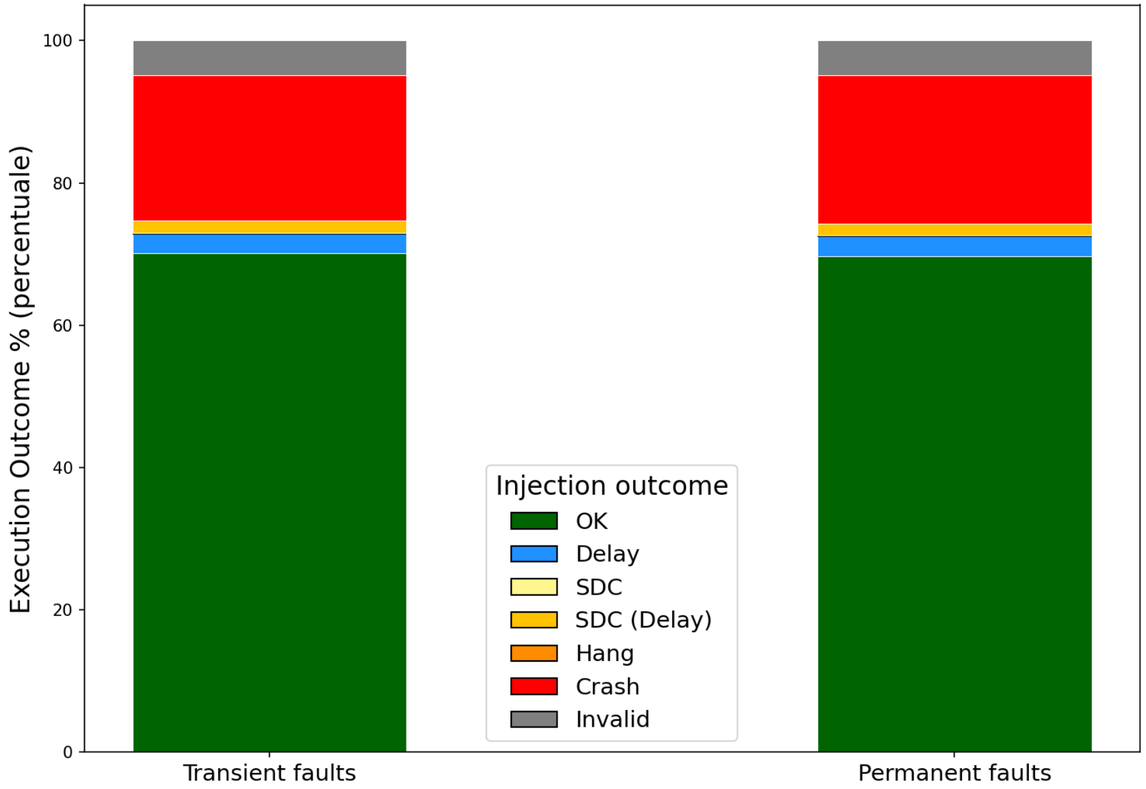
Additional Classes

- SDC + Delay
  Silent error + deadline missed
- Invalid:
  Target not active at injection time

How experiments are executed

1. An injection time is selected
   - For this campaign the same injection time was selected for all the injections
2. A CSV file with all target locations is generated by the command module
3. The target module converts the CSV to memory location
4. Each target kernel variable is selected by the injector
   - E.g., scheduler lists, counters, task control fields
5. A specific byte and bit are flipped at runtime by the injector thread.
6. The run is compared against a golden execution to classify the outcome according to previously reported classes

Overall Campaign Results



Key Takeaway

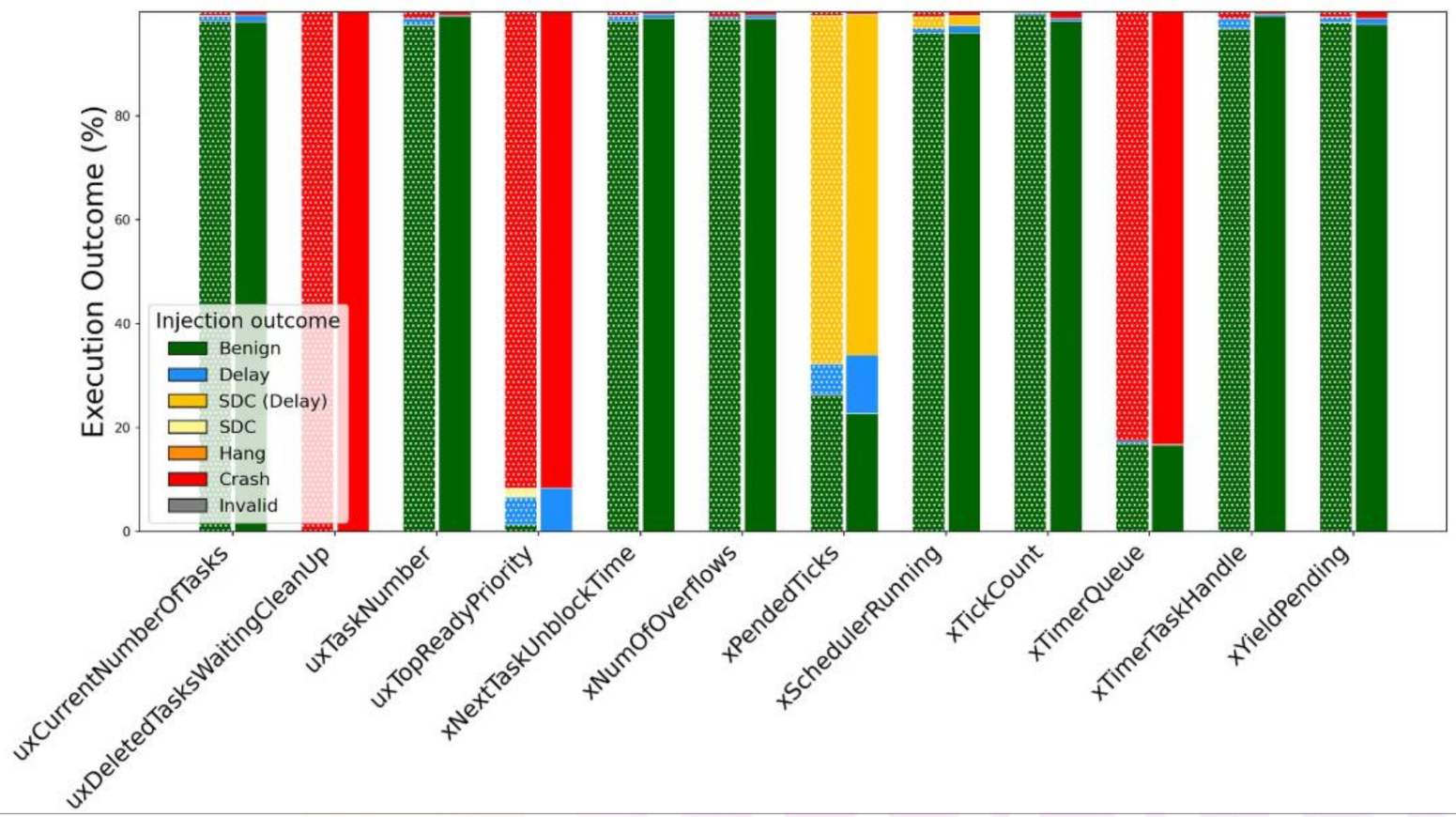
- ~70% of runs completed **successfully**
- >20% resulted in **system crashes**
- **SDC rate below 2%**, often with timing deviations
- Transient and permanent faults show **comparable failure rates**
  - critical corruptions cause immediate, irrecoverable failures regardless of persistence

Fault Type	Benign	Delay	SDC	SDC (Delay)	Hang	Crash	Invalid
Transient	70.16%	2.80%	0.04%	1.69%	0.00%	20.43%	4.88%
Permanent	69.66%	3.00%	0.00%	1.65%	0.00%	20.82%	4.88%

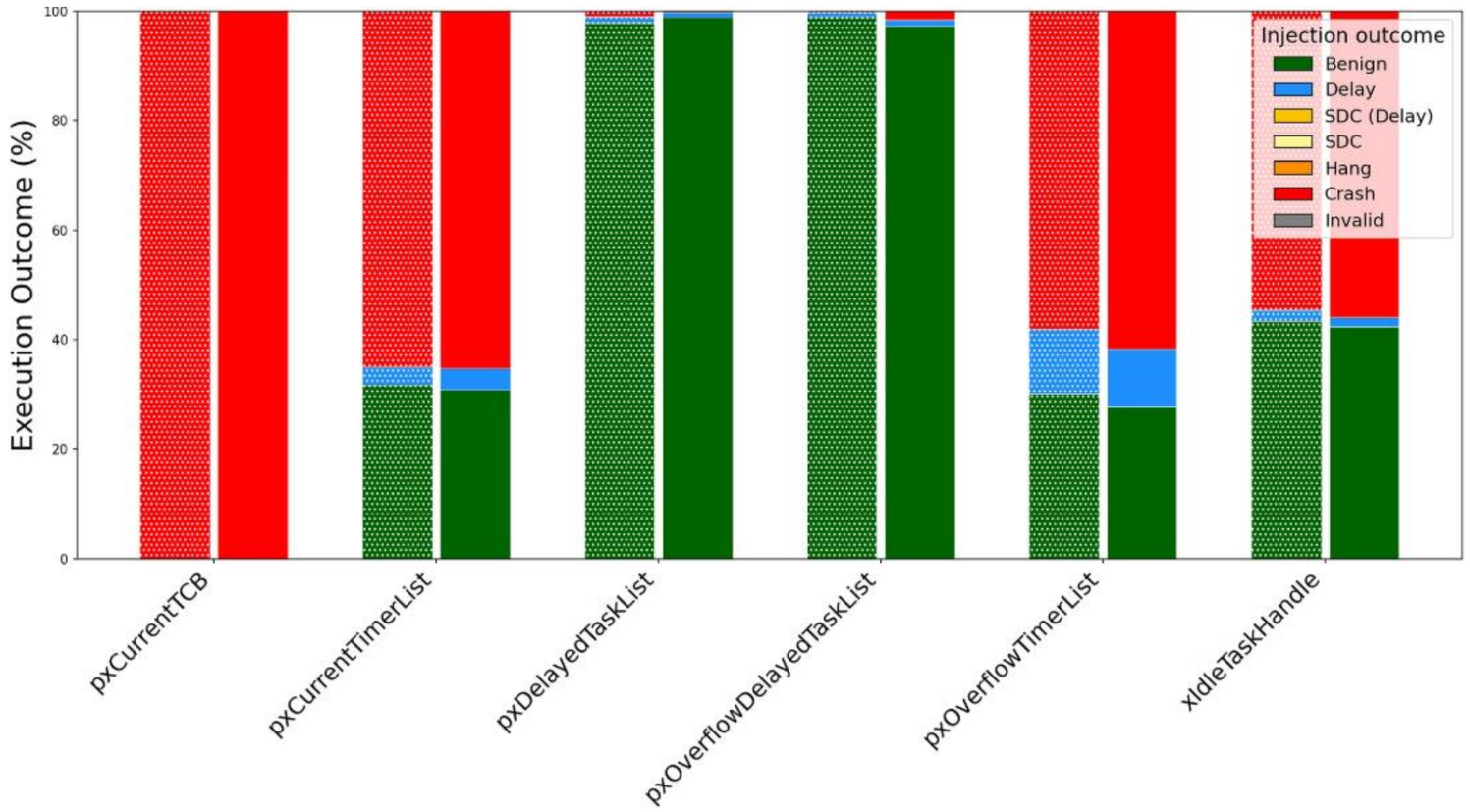
Results by Target Category (1/4)

Variables

- uxDeletedTasksWaiting Cleanup, uxTopReadyPriority, and xTimerQueue show **near-total crash rates**.
- xPendedTicks produces SDC with delay.



Results by Target Category (2/4)



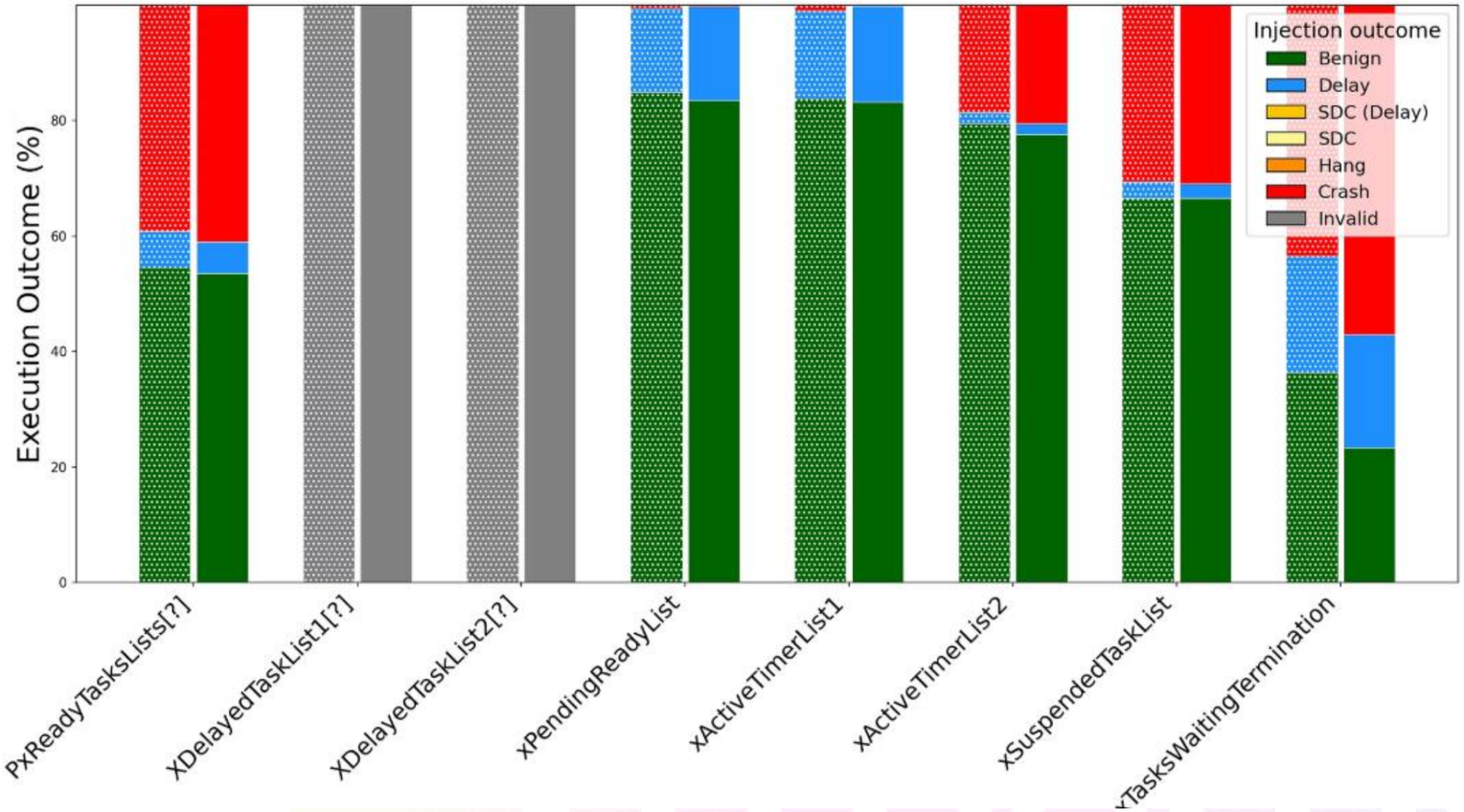
Pointers (Most Critical)

- Bit-flips almost always cause crashes.
- pxCurrentTCB crashes 100% of runs.
- pxCurrentTimerList and xIdleTaskHandle cause crashes in >50% of runs.
- No SDC observed.

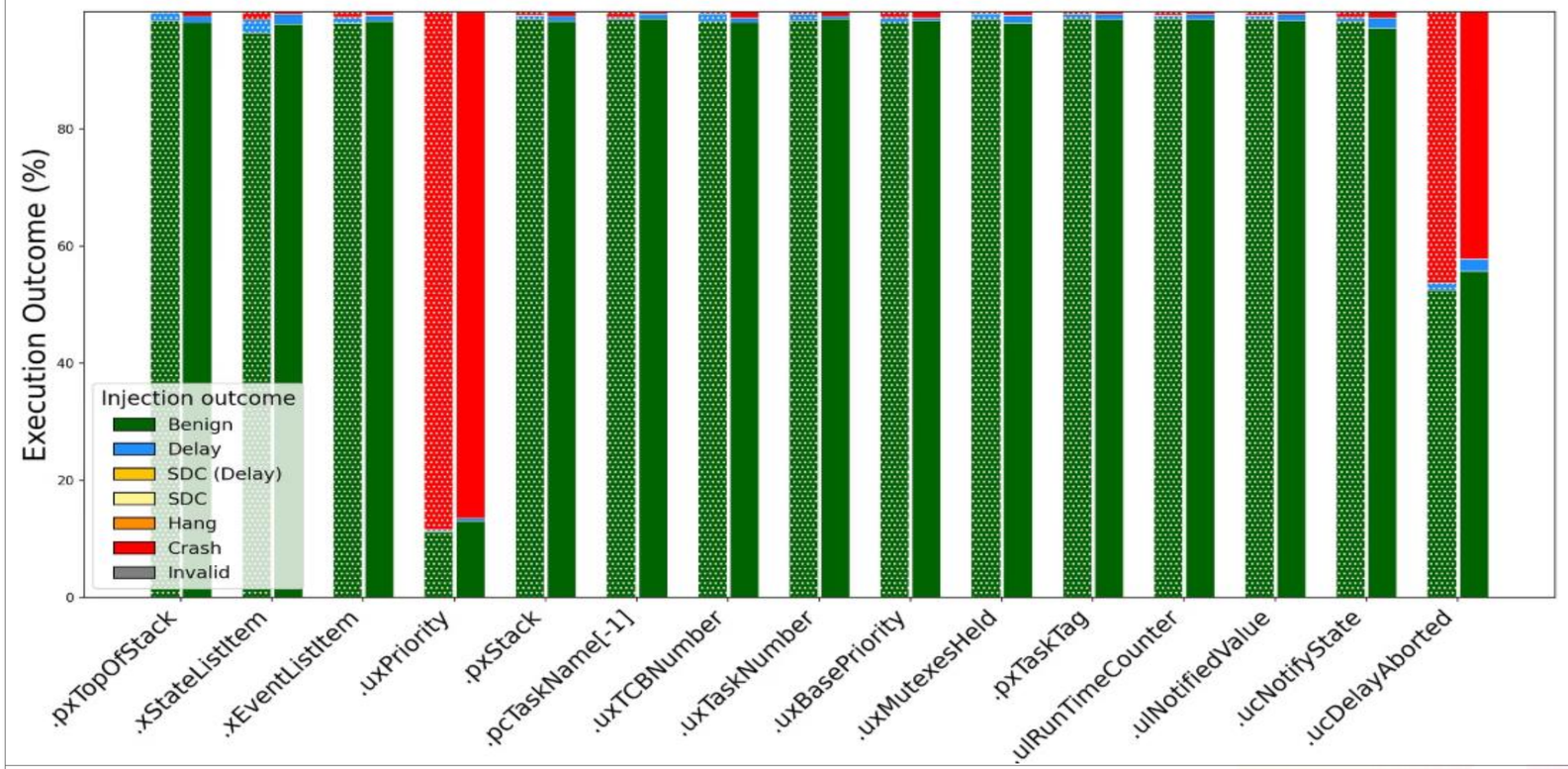
Results by Target Category (3/4)

Lists

- Crash rates range from 20–50%;
  - higher for permanent faults.
- xTasksWaitingTermination is sensitive.
- xSuspendedTaskList tolerates more faults.
- No SDC observed.



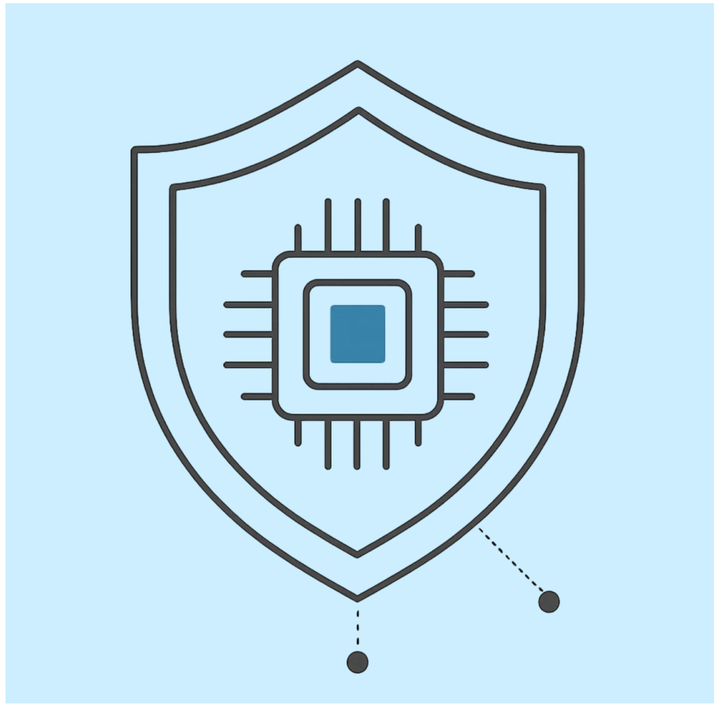
Results by Target Category (4/4)



Current TCB Fields

- Generally limited impact.
- uxPriority causes crashes in >80% of runs
  - underscoring the importance of priority management.
- ucDelayAborted crashes in >40% of runs.
- No SDC observed.

Conclusions & Future Work



KRONOS

- Automated, repeatable, hardware-free kernel-level FI for FreeRTOS.
- Supports both transient and permanent fault models.

Key Finding

- Pointer and scheduler-critical variables are the most vulnerable.
- TCB fields have limited impact
  - except **uxPriority**.

FI campaign extremely fast: avg. wall-clock time = **76.7 s** (σ = 7.1 s)

Future Work

- Release KRONOS as open-source software
- Extend KRONOS to other RTOSes
- Integrate cross-layer models linking OS-visible faults to device-level phenomena

Supported by COLTRANE-V (PRIN 2022) and Space It Up (ASI/MUR, CUP I53D24000060005).

Thank you

for your attention

Luca Mannella, Ph.D.

Post doctoral fellow

luca.mannella@polito.it

Stay Connected

Publications

More about the SMILIES group