Mastering Uncertainty: Optimizing Real-Time Systems for Robustness and Resilience

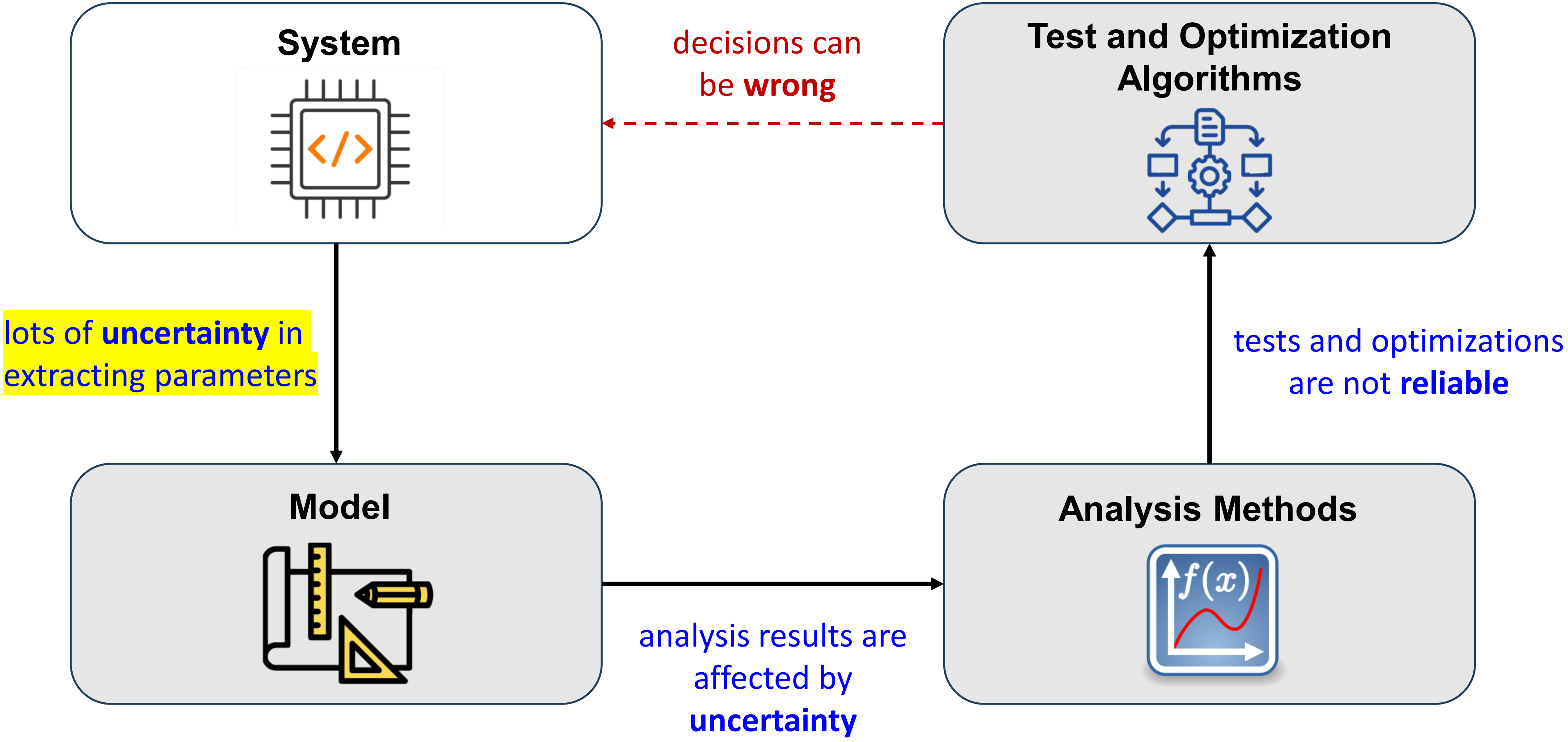
Alessandro Biondi

Scuola Superiore Sant'Anna, Pisa, Italy

Sant'Anna
Scuola Universitaria Superiore Pisa

Retis
Real-Time Systems Laboratory

Design & Test Loop



The Elephant in The Room

- **Let's accept the reality**: In many systems WCETs are unknown
  - They're the same ones we use to motivate most of today's research on real-time systems
  - Response-time bounds cannot be trusted
  - There's a major source of uncertainty in our models

Design for uncertainty

- What's the best configuration to **maximize robustness** in the face of uncertainty?
- How to make a system as **resilient** as possible given uncertainty?
- How **risky** is a system under uncertainty?

Modeling Uncertainty

- Honestly, I still don't have a strong opinion ☺
  - A measure of uncertainty in [0, 1] for each execution time bound?

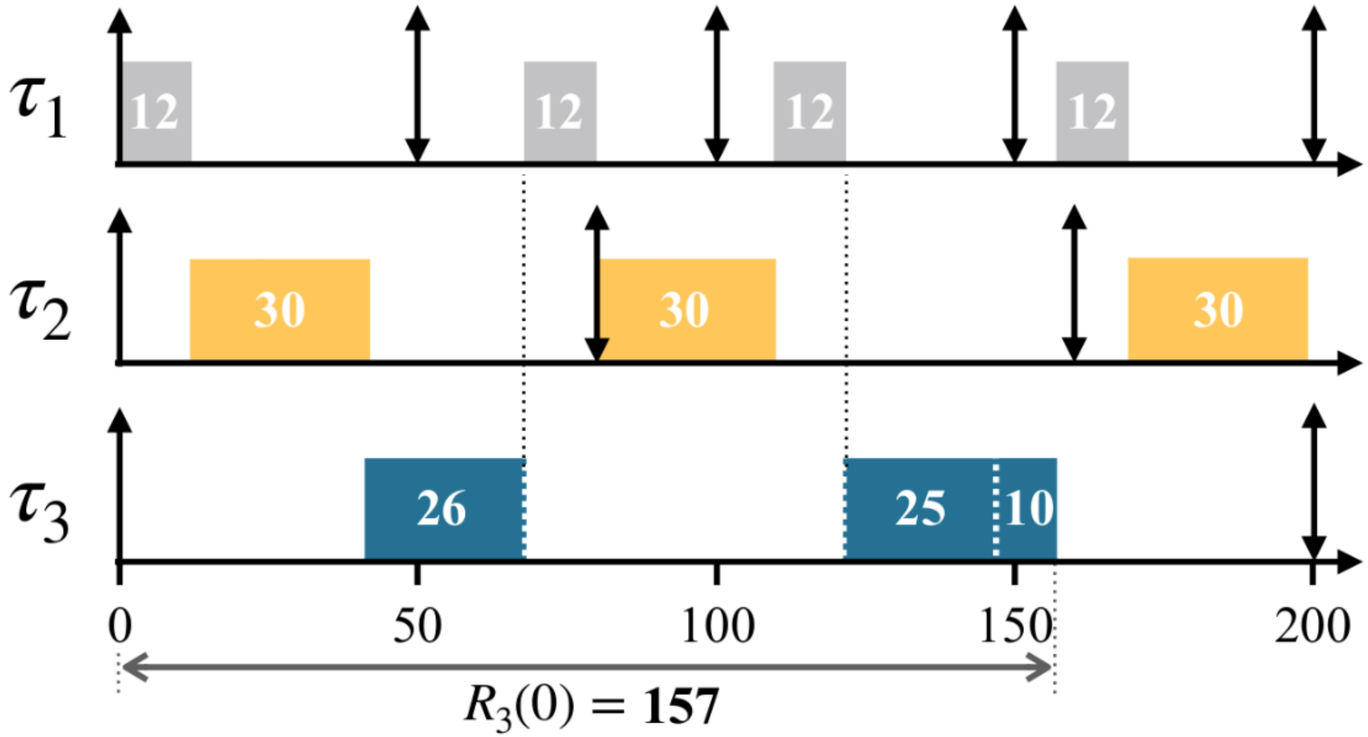
- **Pragmatic observations**:
  - Execution time estimates can be obtained by measurements (*nominal exec. times*)
  - Code complexity affects uncertainty
    - More branching → more uncertainty (e.g., consider branch prediction)
    - More paths → unpredictable cache hit/miss patterns → more uncertainty
  - Memory access affects uncertainty
    - More memory accesses → more opportunity for contention → more uncertainty
  - Coverage affects uncertainty
    - Less coverage during measurements → more uncertainty

Motivating Example

From Matteo Zini's presentation @ RTSS 2024

For example, consider this simple limited-preemptive taskset:

Task	Period	NET
τ_1	50	<12>
τ_2	80	<30>
τ_3	200	<26,25,10>



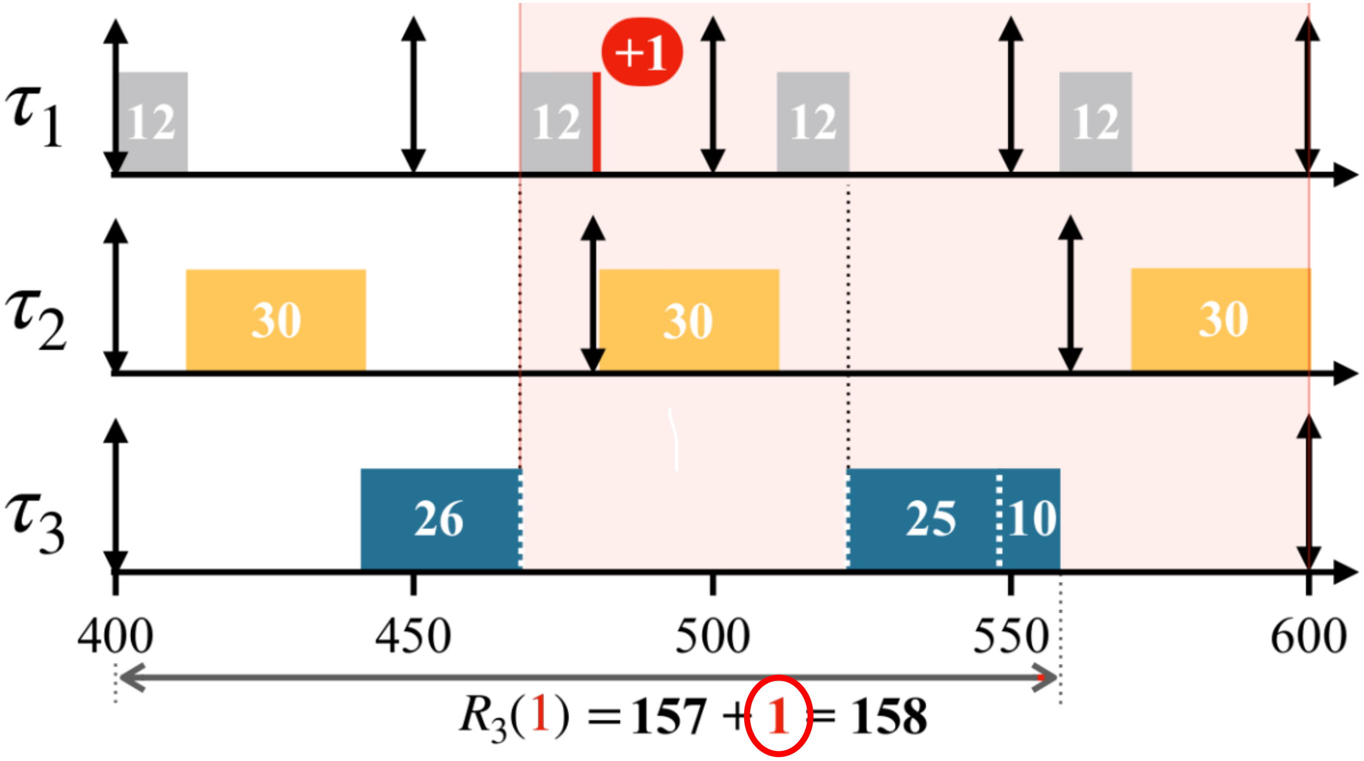
M. Zini, F. Marković, D. Casini, A. Biondi, and B. Brandenburg, "In Search of Butterflies: Exceedance Analysis for Real-Time Systems under Transient Overload", Proceedings of the 45th IEEE Real-Time Systems Symposium (RTSS 2024), pp. 229–242, December 2024.

Motivating Example

From Matteo Zini's presentation @ RTSS 2024

We **add 1 unit of exceedance** to the second job of task τ_1

Task	Period	NET
τ_1	50	<12>
τ_2	80	<30>
τ_3	200	<26,25,10>



τ_3's response time increased by 1 time unit

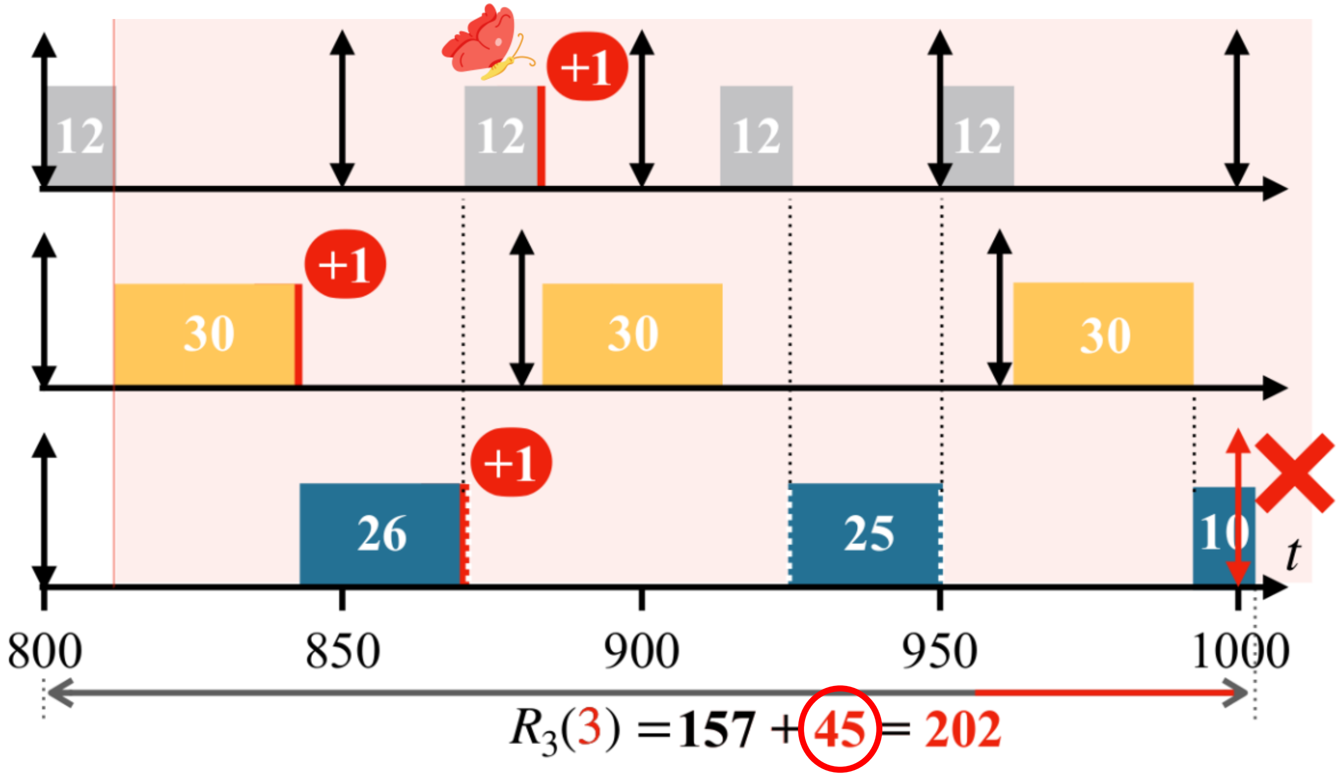
M. Zini, F. Marković, D. Casini, A. Biondi, and B. Brandenburg, "In Search of Butterflies: Exceedance Analysis for Real-Time Systems under Transient Overload", Proceedings of the 45th IEEE Real-Time Systems Symposium (RTSS 2024), pp. 229–242, December 2024.

Motivating Example

From Matteo Zini's presentation @ RTSS 2024

We add 1 unit of exceedance to the first job of task τ_2 and τ_3

Task	Period	NET
τ_1	50	<12>
τ_2	80	<30>
τ_3	200	<26,25,10>



τ_3's response time increased by **45 time units**!

M. Zini, F. Marković, D. Casini, A. Biondi, and B. Brandenburg, "In Search of Butterflies: Exceedance Analysis for Real-Time Systems under Transient Overload", Proceedings of the 45th IEEE Real-Time Systems Symposium (RTSS 2024), pp. 229–242, December 2024.

Real-Time Systems Laboratory

Response-Time Nonlinearities

From Matteo Zini's presentation @ RTSS 2024

The consequences of **NET exceedance** are not easy to predict:

- NET + 1 ⟹ Response time + 1
- NET + 2 ⟹ Response time + 2
- NET + 3 ⟹ Response time **+ 45** → **Nonlinear** increase!
- …

If we neglect this phenomenon, we might **over-estimate the system's temporal safety margin**

M. Zini, F. Marković, D. Casini, A. Biondi, and B. Brandenburg, "In Search of Butterflies: Exceedance Analysis for Real-Time Systems under Transient Overload", Proceedings of the 45th IEEE Real-Time Systems Symposium (RTSS 2024), pp. 229–242, December 2024.

Response-Time Discontinuities

From Matteo Zini's presentation @ RTSS 2024

Response-time discontinuities are not trivial to predict

Task	Period	NET
τ_1	50	<12>
τ_2	80	<10, 20>
τ_3	200	<26,25,10>

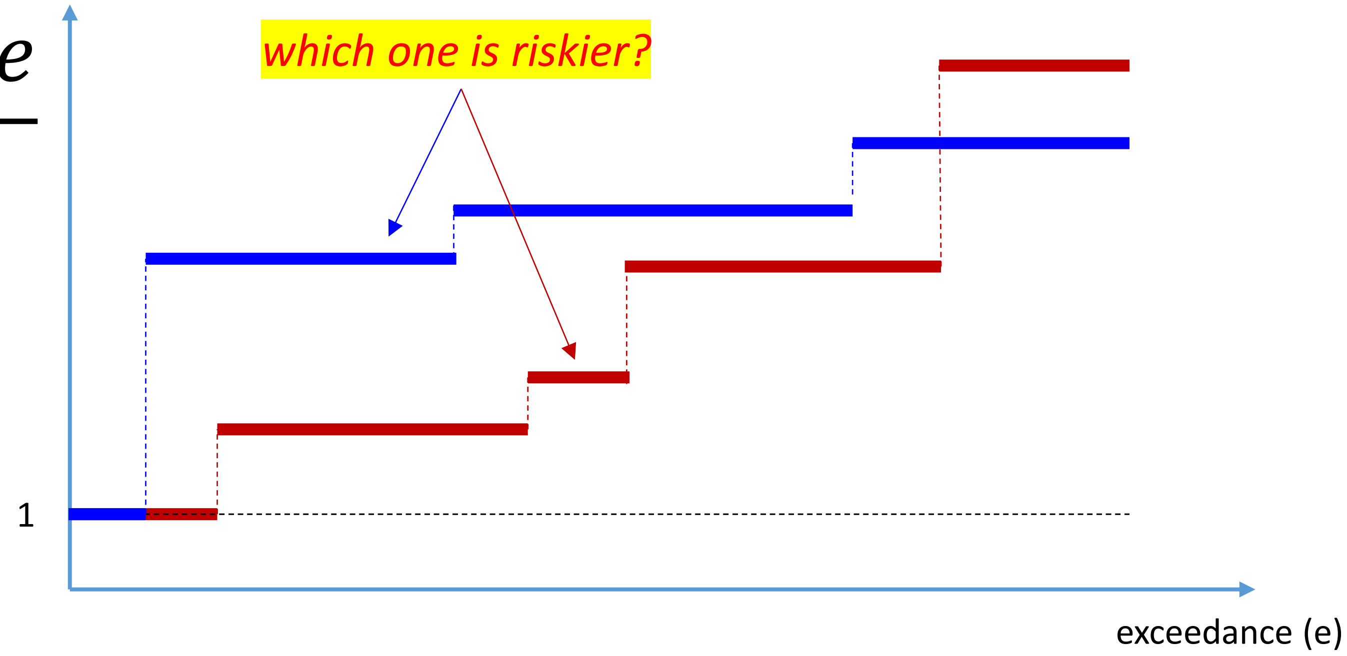
M. Zini, F. Marković, D. Casini, A. Biondi, and B. Brandenburg, "In Search of Butterflies: Exceedance Analysis for Real-Time Systems under Transient Overload", Proceedings of the 45th IEEE Real-Time Systems Symposium (RTSS 2024), pp. 229–242, December 2024.

Risk Factor

- When a system experiences exceedance, the *best* it can happen is a **linear, unitary-slope increase** in response times
  - **Risk** is determined by discontinuous increases of response times (jumps)
  - Hence $\boldsymbol{R_i(e) - e}$ determines risk

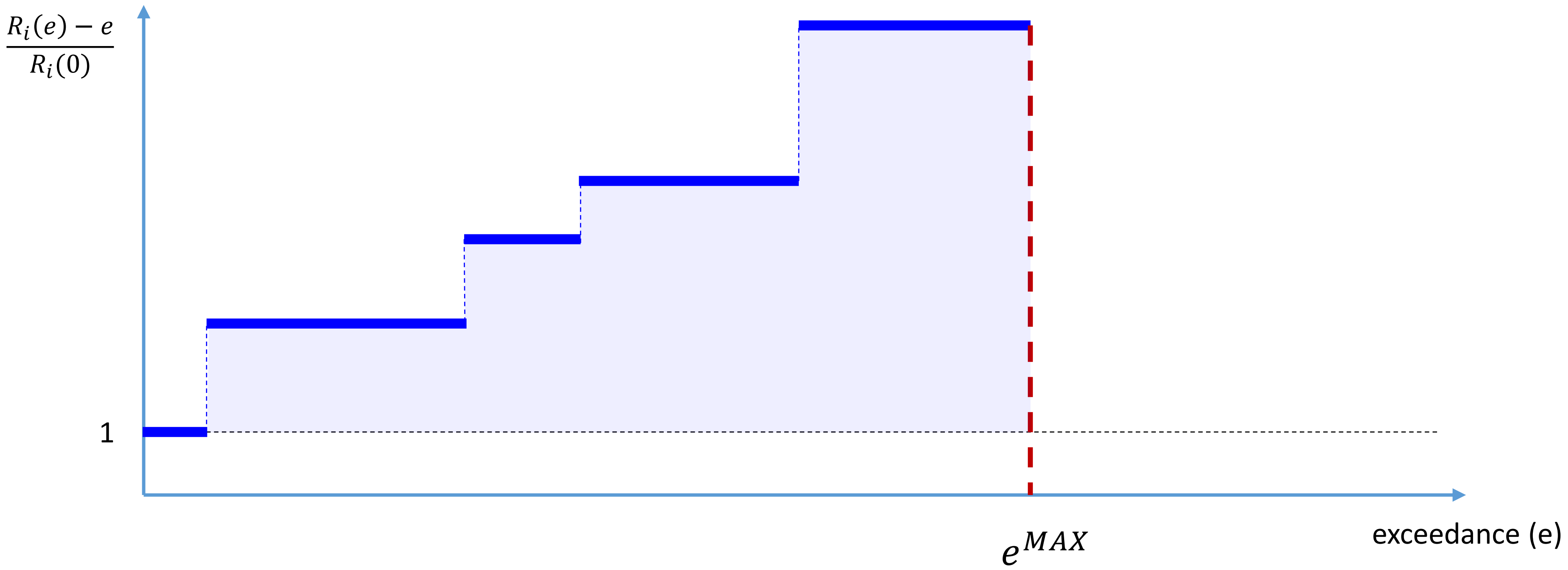
$$\frac{R_i(e) - e}{R_i(0)}$$

normalized to nominal response time (no exceedance)



Risk Factor

$$\gamma_i\left(e^{MAX}\right) = \int_0^{e^{MAX}} \frac{R_i(e) - e}{R_i(0)} - 1 \;\; de$$

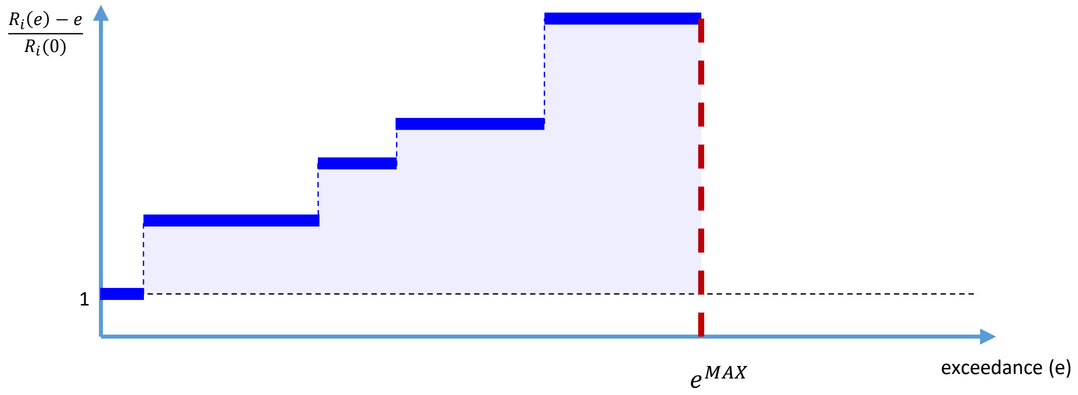


Risk Factor

- **Informal interpretation of risk factor $\gamma_i(e^{MAX})$**
  - It captures *"how much"* exceedance introduces large discontinuous increases
  - It captures *"how quickly"* response times jump with exceedance

- This definition depends on the maximum expected exceedance e^{MAX}
  - Ouch…yet another parameter?

$$\gamma_i(e^{MAX}) = \int_0^{e^{MAX}} \frac{R_i(e) - e}{R_i(0)} - 1 \;\; de$$



Minimizing Risk Factor

- **Challenge**: *Design real-time systems to minimize risk factor*
  - Either for a selection of tasks or all tasks
  - Weighting risk factor by a trustworthiness/uncertainty level of execution times
  - Considering arbitrary maximum expected exceedance
- It's an optimization problem

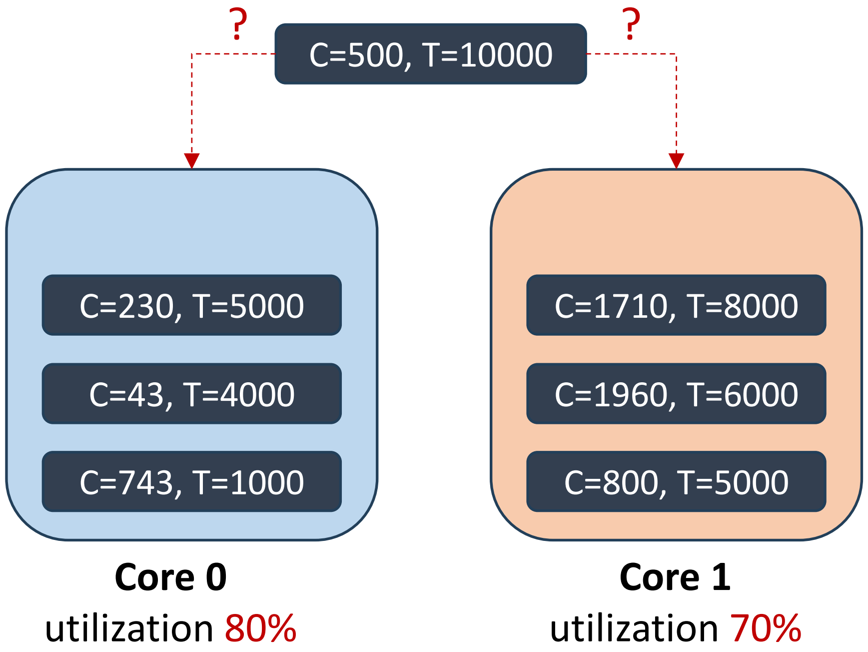
Examples:

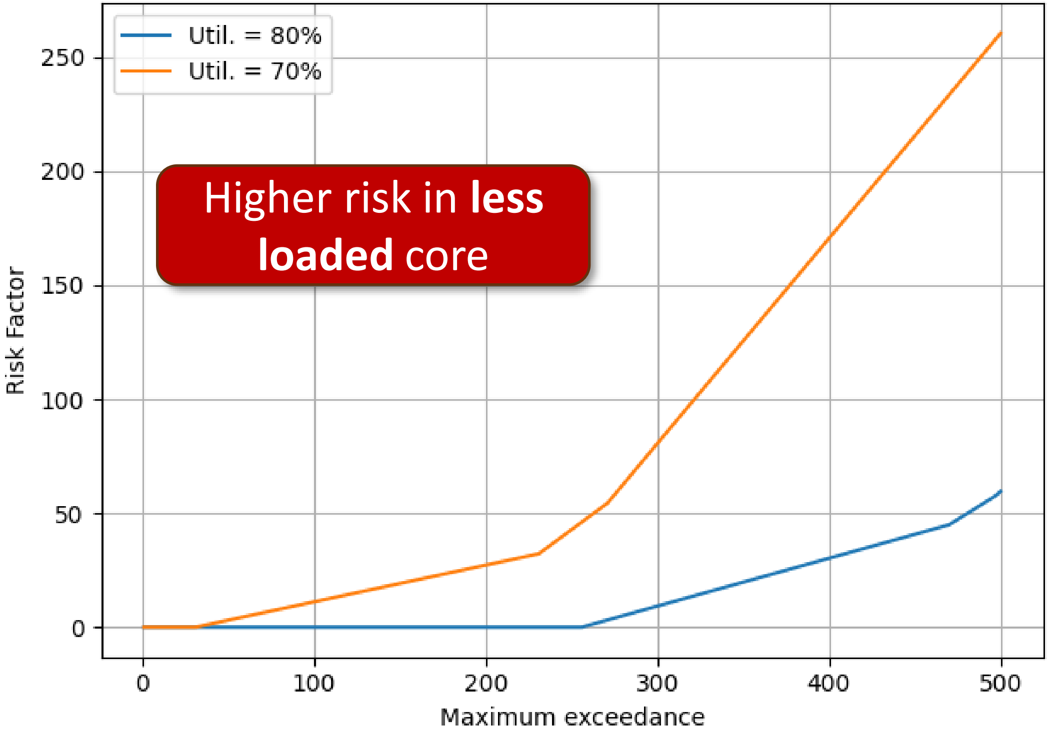
- Partition tasks on multicores according to risk
- Find task periods that minimize risk while securing control performance
- Configure locking protocols to minimize risk
- Configure Logical Execution Time (LET) intervals according to risk
- …

Subject to classical schedulability under nominal execution times

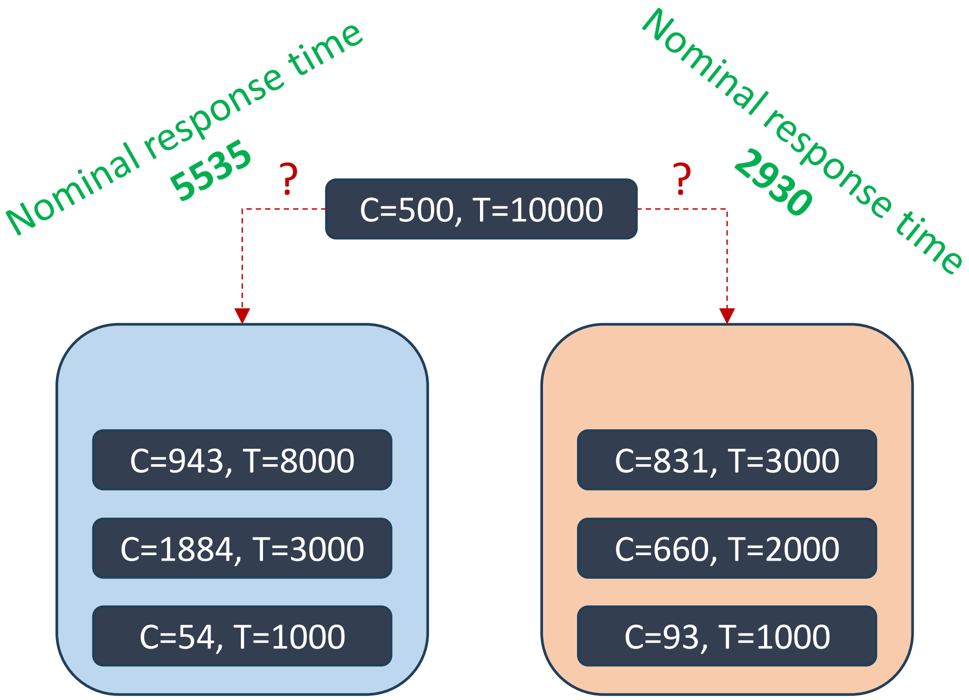
Example: Task Partitioning (1)

- Place tasks to cores to minimize risk factor
  - E.g., for a target, relevant task
- **Partitioned fixed-priority scheduling of Liu&Layland tasks**
  - Even with a simple scheduler and task model, decisions are not obvious



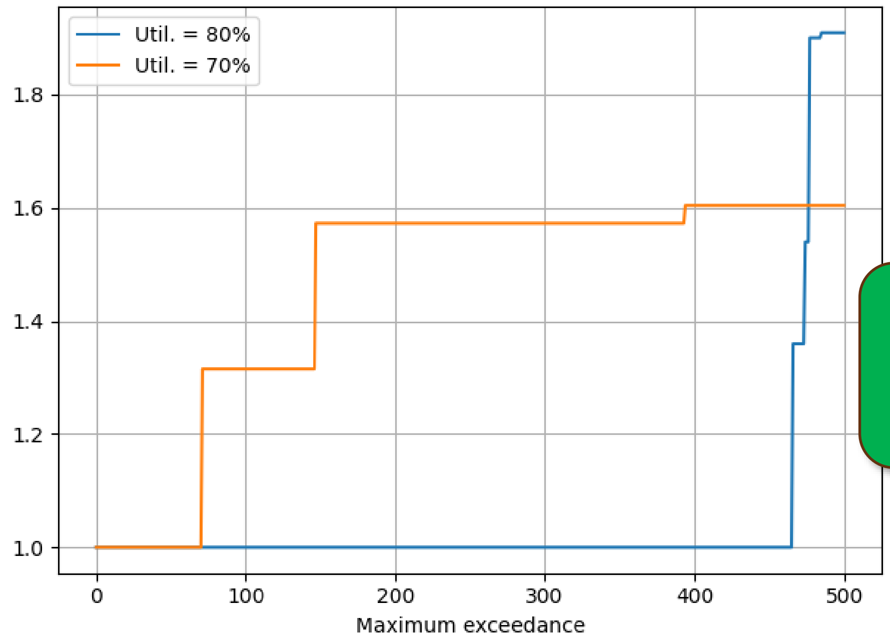


Example: Task Partitioning (2)

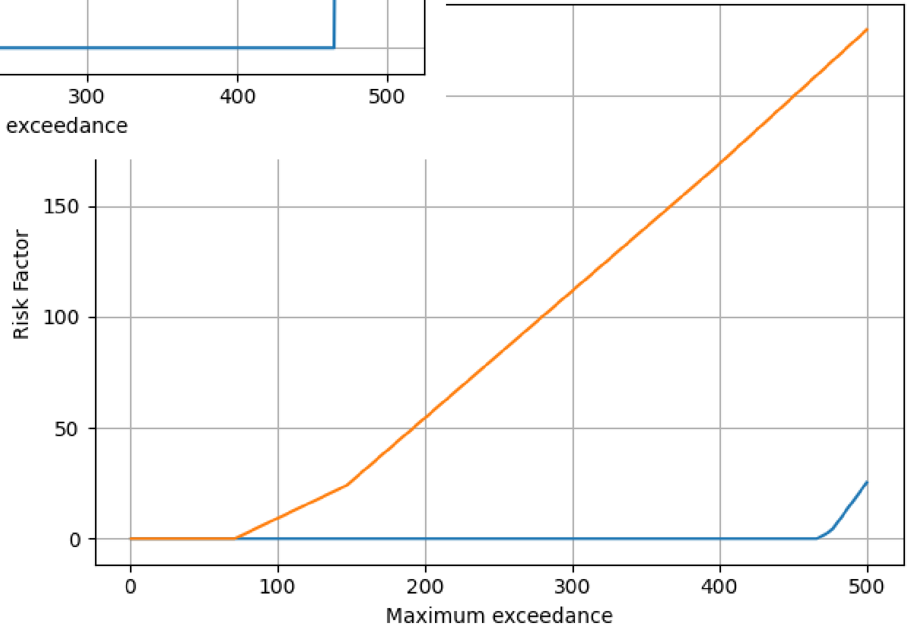


Core 0
utilization 80%

Core 1
utilization 70%



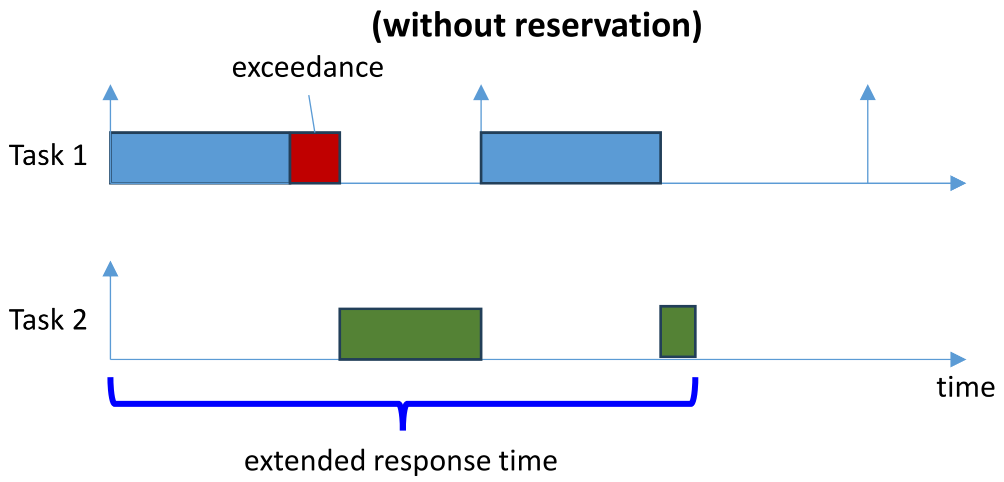
Lower risk in **more loaded** core with **larger** nominal response ime

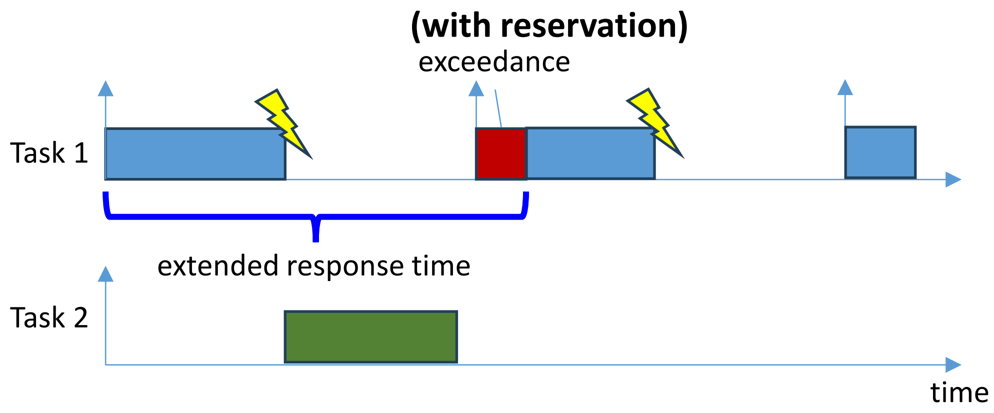


Achieving Resiliency

Since late **90's**

- **Reservation servers** work great to isolate the effects of exceedance
  - It's a way to control uncertainty





- Can we design ad **adaptive** reservation mechanism that minimizes the risk factor?
  - Budget reclaiming is very effective (since early 2000's)

There's space for new reservation algorithms that, jointly applied with budget reclaiming,
limit response-time discontinuities and hence the risk factor

Thank you!

Alessandro Biondi
alessandro.biondi@santannapisa.it