RTNS 2024

TTTech

Taming System Complexity

Global Scheduling and the Time-Triggered Approach

Silviu S. Craciunas

Principal scientist @ TTTech Labs

AIMS 5.0

TTTech project highlights

TTTech Auto safety and robustness in series production

Hardware development until the A-Sample grade

Scalable distributed control system from TTTech Industrial into Vestas 2, 4 and 6 Megawatt turbines

Collins Aerospace Power System and Air Management for the Boeing 787-8, 787-9, 787-10 family

TTTech TTEthernet Switch and End System IP products for the avionics system.

TTEthernet is the “nervous system“ i.e. avionics network platform of the space craft

TTTech Group
Key facts

Founded in 1998, headquartered in Vienna, Austria, with 19 offices in 14 countries worldwide

Products in 1173 production programs

Connected companies: TTTech Auto, TTTech Industrial, TTControl, RT-RK

2,300+
Employees/ subcontractors

60
Nations represented by our workforce

36 ongoing research projects

635,000 ECUs & 82,000 displays

10,000
Vestas turbines

3 million
cars with MotionWise on the road

1.5 billion
operating hours of railway systems

1.9 million
kilometers of slopes

>2 million
kilometers covered by TTEthernet in deep space

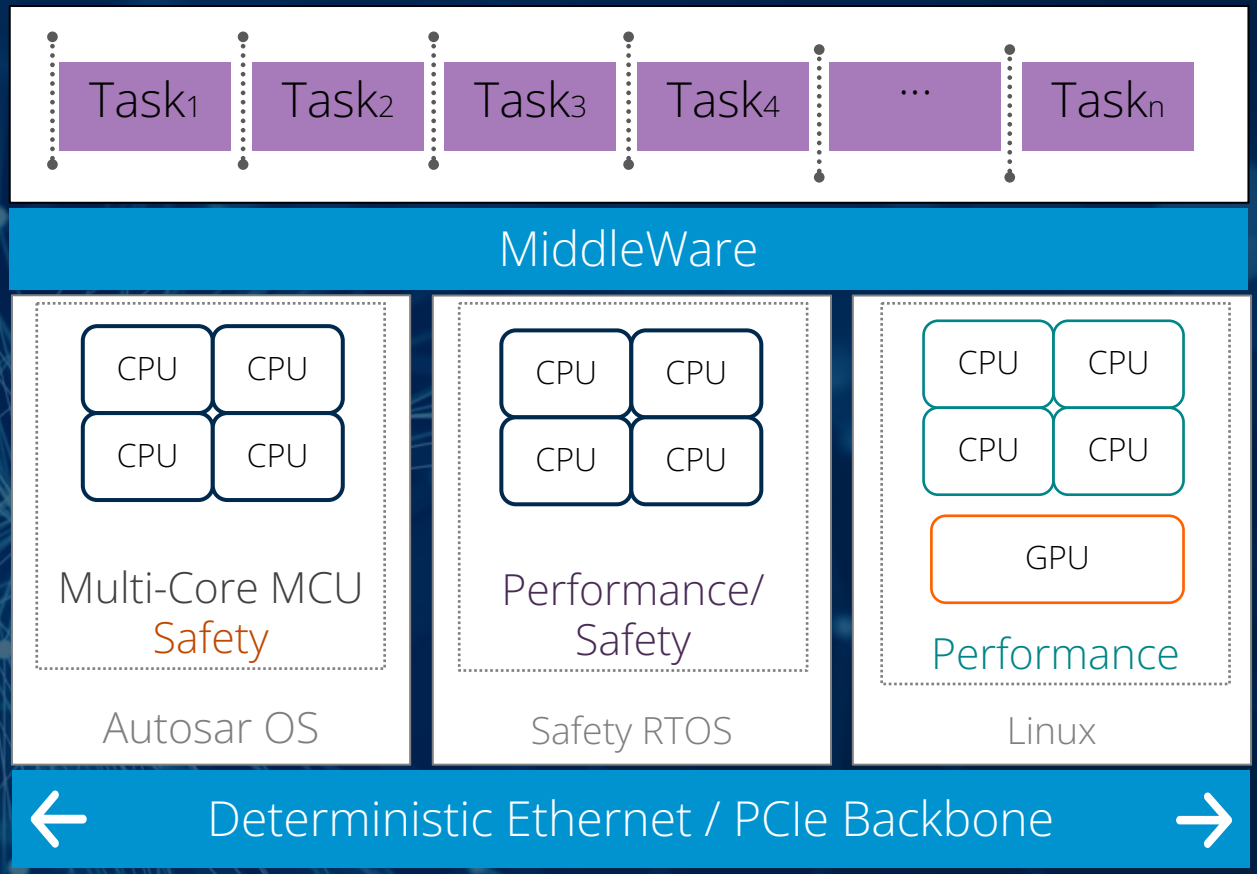
1 billion
flight hours

Numbers as of March 2022

November 8, 2024

Systems and platform

Heterogeneous multi-core multi-SoC platforms featuring a variety of CPUs and GPUs running at different speeds, which are interconnected through either a deterministic Ethernet backbone (TSN) or through PCIe



Large number of safety-critical tasks with increasing code size that have complex timing requirements beyond just deadlines

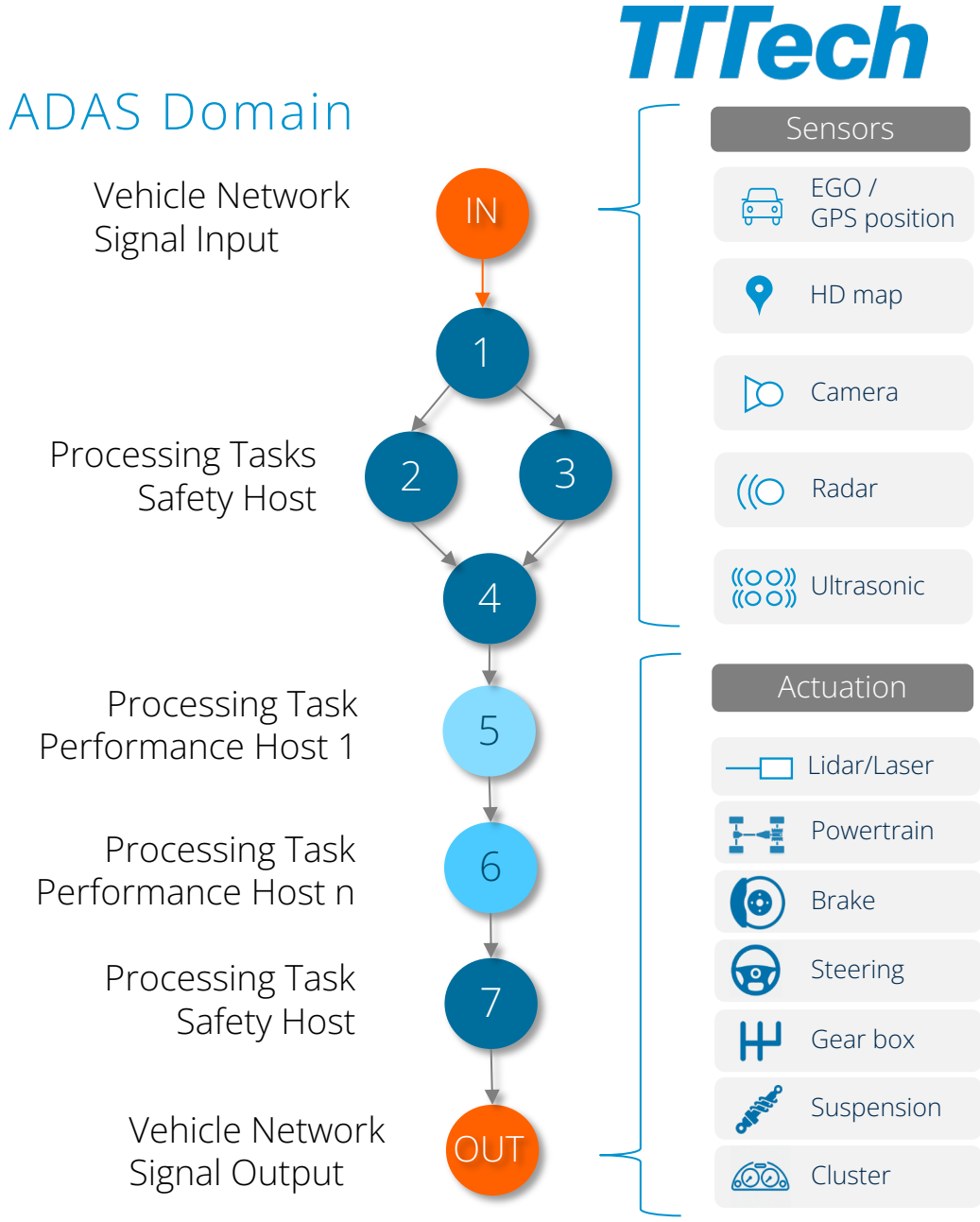
Cause-effect chains

Cause-effect chains are a characteristic of automotive software:

- provide additional timing and dependency requirements on the execution of tasks
- can span across multiple activation patterns
- include multiple tasks, even the same task multiple times
- have priorities and end-to-end latencies
- include communication latencies
- task execution and network communication must be aligned across SoCs and CPUs

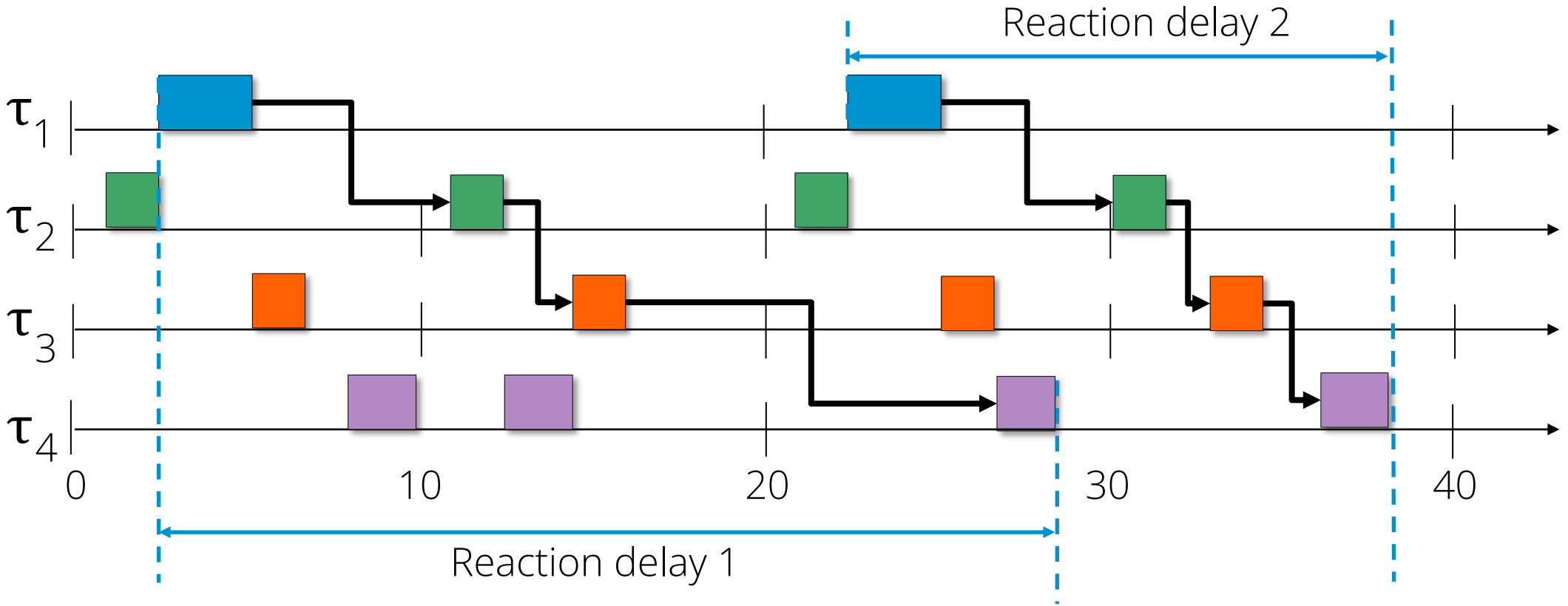
Semantics:

- Reaction delay
- Data age
- Order



Cause-effect chains: reaction delay

After each event, there must be a reaction within the maximum allowed time




Complexity

System complexity – ADAS example

TTTech

Increasing complexity of car software
(lines of code, millions)

+

Increasing complexity of timing requirements

- Critical event/data-driven chains
- Hard real-time and determinism requirements
- Orchestration needs of heterogeneously distributed applications
- Mixed-criticality and Freedom From Interference requirements
- Complex task dependencies

=

Design-time guarantees

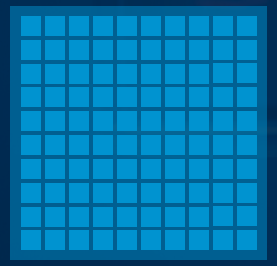
- Testing-based approaches are not sufficient anymore
- Correct by design approach with mathematically-proven guarantees
- This implies more knowledge (e.g., timing budgets) and solutions to complex scheduling/configuration problems

"Design approaches not based on mathematically proven real time scheduling properties are prone to missing deadlines during unusual operational conditions" UL4600 Standard for Safety for the Evaluation of Autonomous Products

Giving design-time guarantees implies solving the configuration/scheduling problem

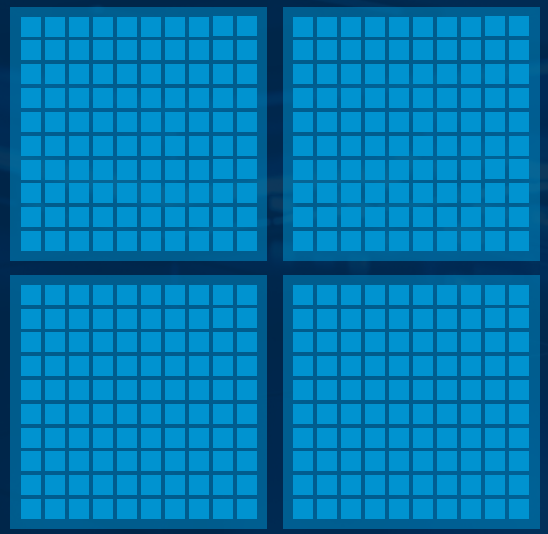
Complexity in numbers

Software demands



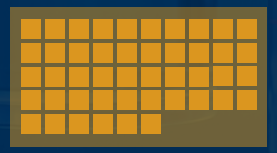
100 Applications

400 Tasks

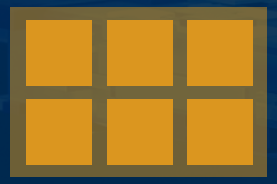


Computation unit resources

46 CPU cores



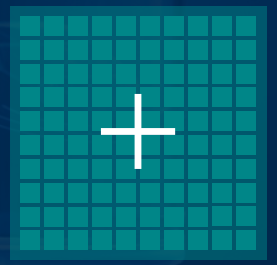
6 Hardware accelerators



2 TSN switches

1024 virtual links

System constraints



100+

time, latency, precedence constraints etc.

Solution space

Theoretical vs Valid configurations

10^{5000} possible configurations

Atoms in the universe 10^{80}

10^{5} correct configurations

How can we solve anything given this complexity?

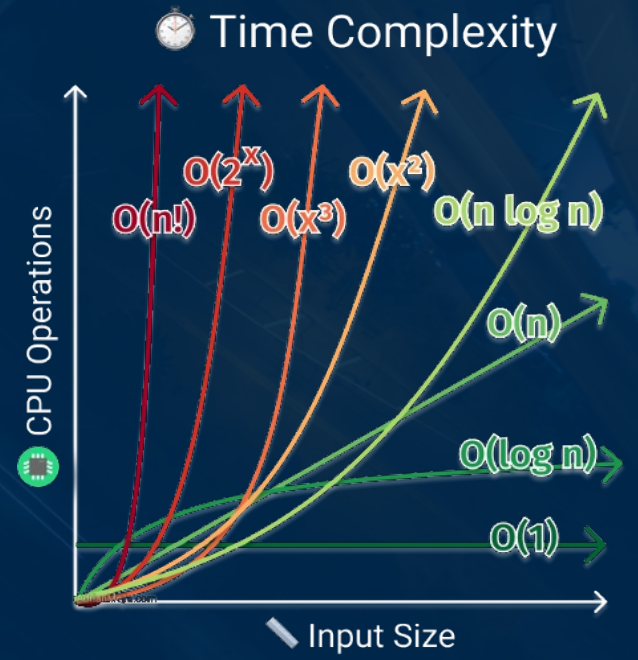
Classical (FP/EDF) approaches

Classical approaches can give design-time guarantees, sometimes even very efficiently.

Deadline/rate monotonic, EDF, etc., are efficient ways to assign priorities, but only for specific cases:
- RM is efficient if: release jitter is 0, deadlines are equal to the periods, no dependencies
- DM is efficient if: release jitter is 0, deadlines are less than or equal to periods, no dependencies

Current automotive applications (e.g., ADAS/AD) are more complex:
- providing design-time guarantees implies checking correctness over all n! priority orderings
- e.g., 25 tasks → 15511210043330985984000000 schedulability checks

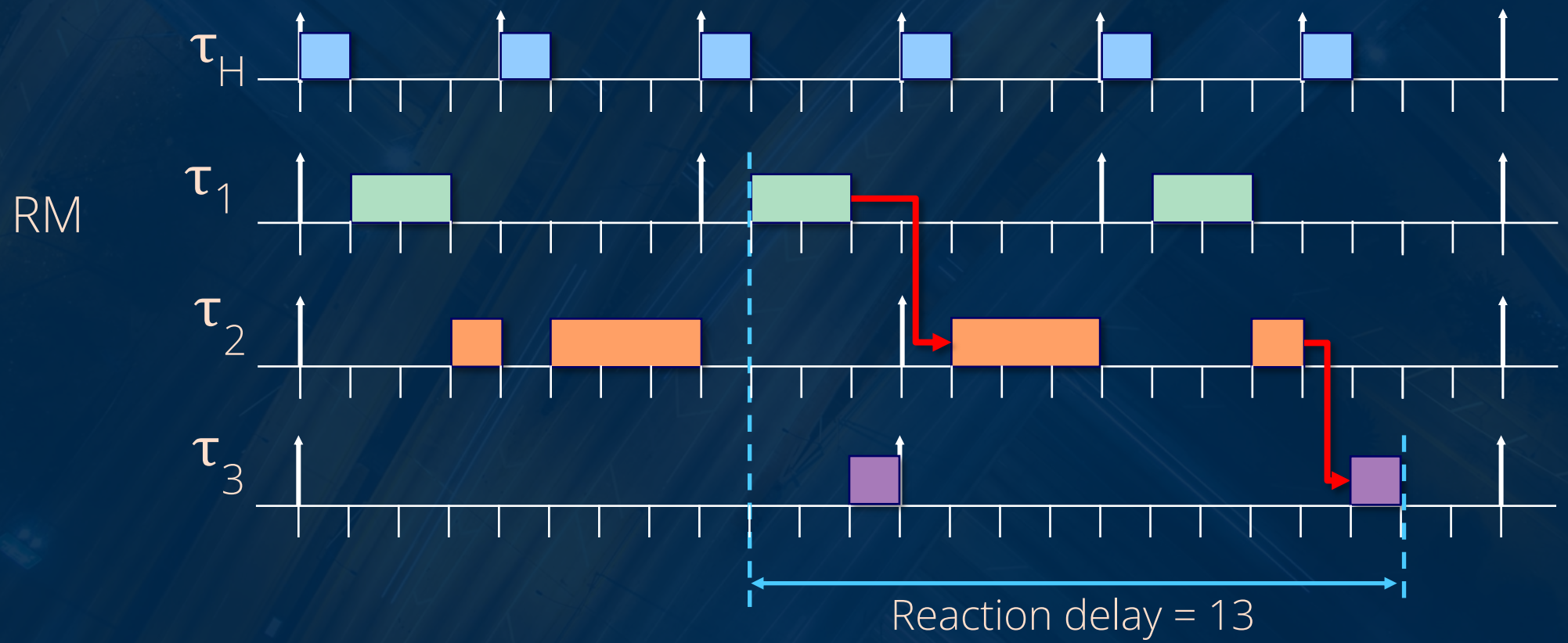
Giving design-time guarantees beyond deadlines is a very complex problem to solve for classical approaches.



Fixed-priority/EDF scheduling wrt. dependency chains

High-priority task: $T_H(1,4)$

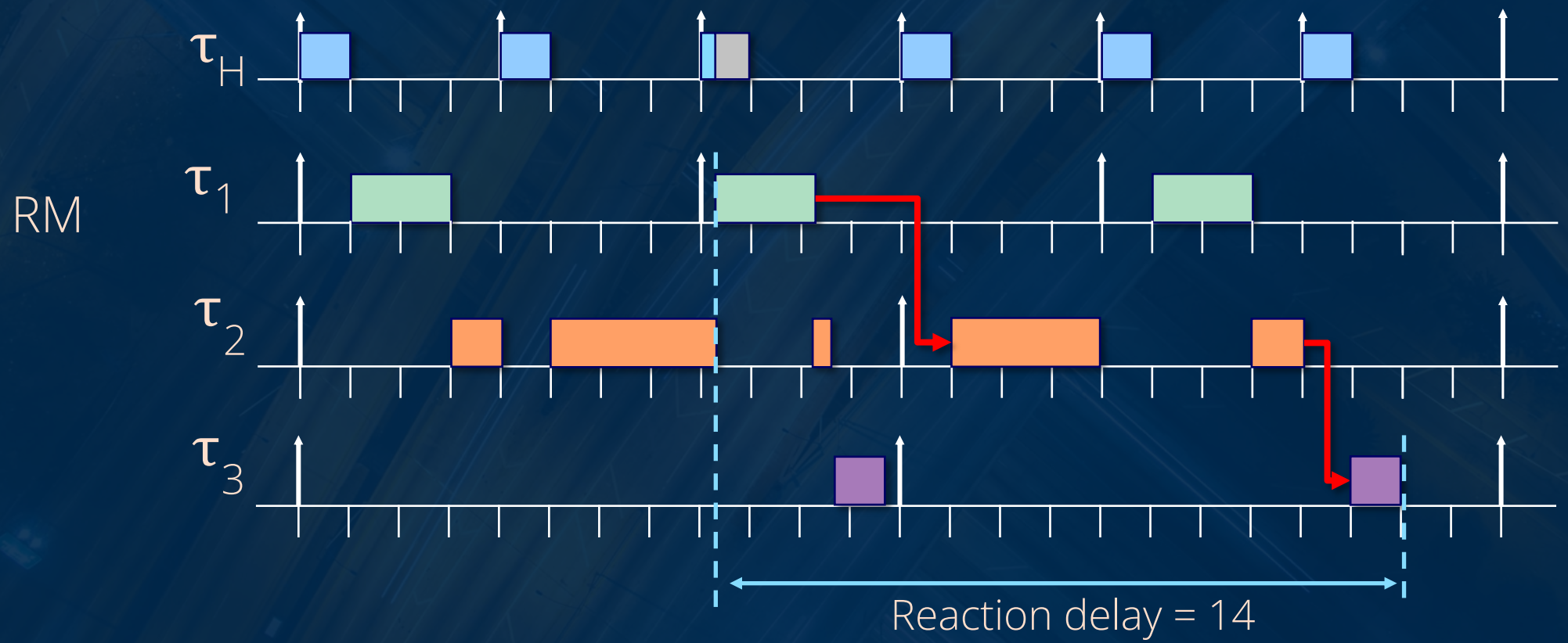
Chain: $T_1(2,8)$, $T_2(4,12)$, $T_3(2,12)$



Fixed-priority/EDF scheduling wrt. dependency chains

High-priority task: $T_H(1,4)$

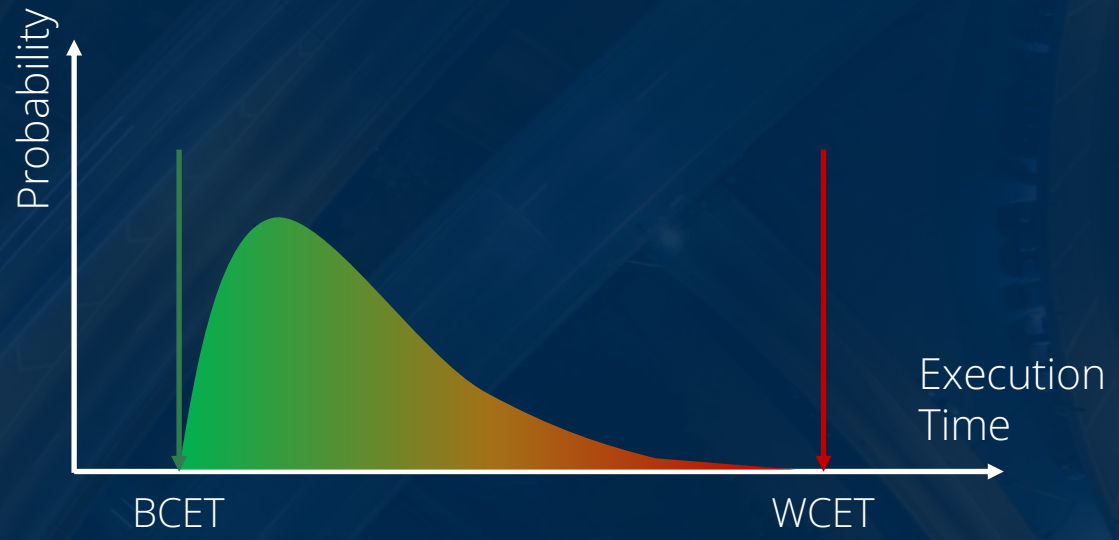
Chain: $T_1(2,8)$, $T_2(4,12)$, $T_3(2,12)$



Complexity of runtime states

WCET is used for generating schedules and testing schedule correctness (also for FP)

At runtime, tasks execute for less than the WCET



Example:
BCET = 1ms
WCET = 2ms
Microtick = 1us

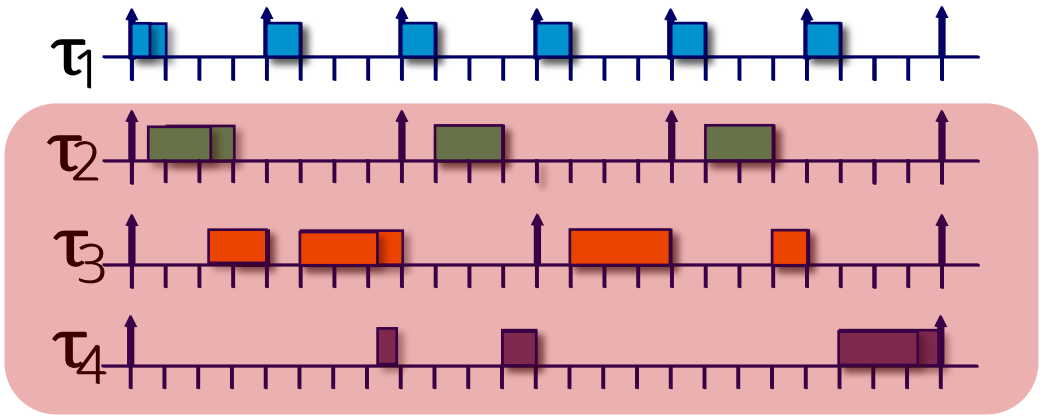
1000 possible execution patterns

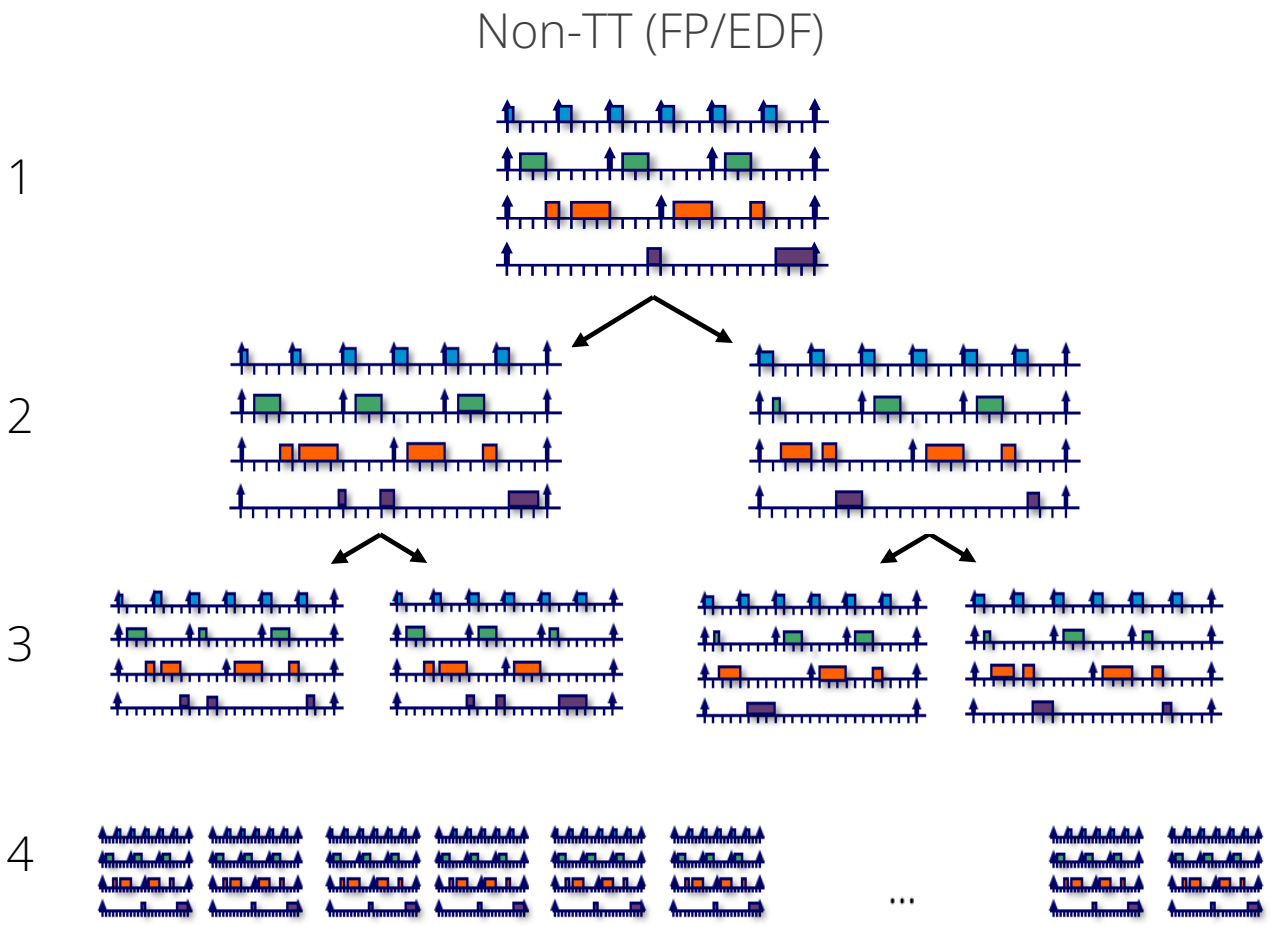
What happens if one task executes for less than the computed WCET?

Complexity of runtime states

What happens if at some point in time, one task executes for less than the computed WCET?

Classical mechanisms
- changes the execution of all other tasks
- state change for the entire system



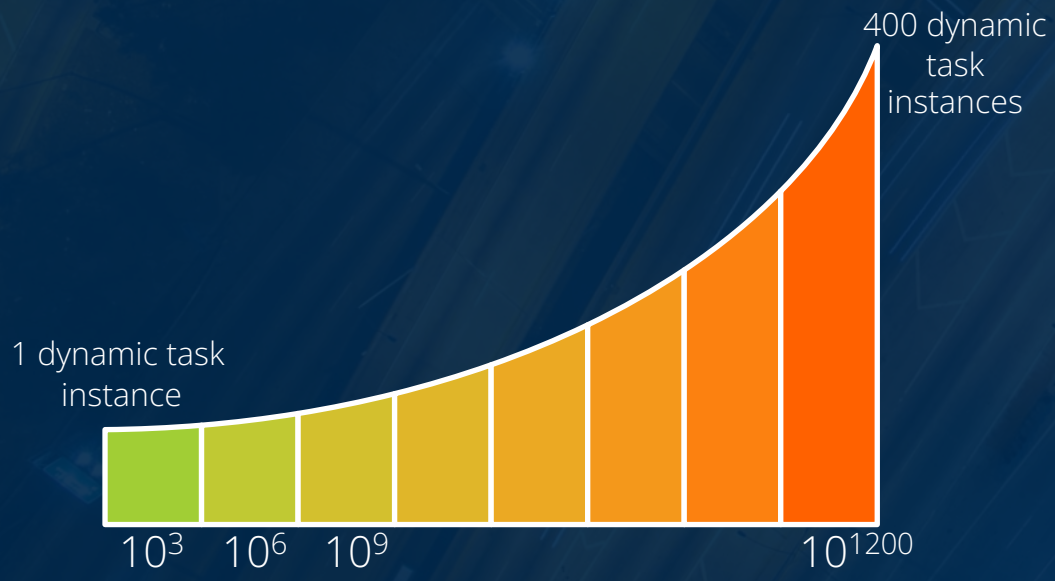


State space explosion

How many system states do I have to potentially check/look at to give design-time guarantees?

Non-time-triggered (FP/EDF/etc.):

- changes the execution of all other tasks
- state change for the entire system



Number of possible assignments of tasks to cores

×

Number of priority orderings

×

Number of system states

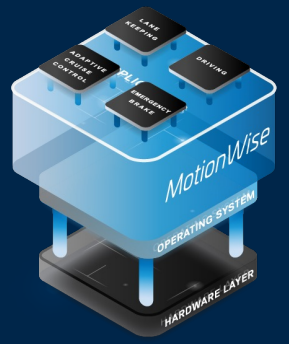
$52^{400} \times 400! \times 10^{1200} = 10^{2755}$

Time-Triggered Scheduling

The Time-Triggered advantage

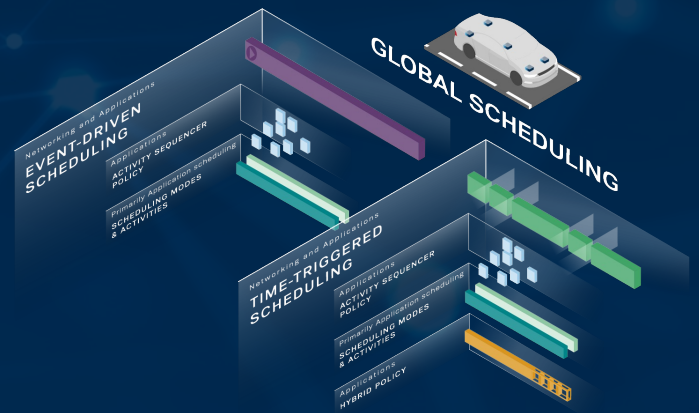
TTTech

Time-Triggered provides correct-by-design real-time guarantees!



Advantages of the Time-Triggered approach:

- **Complex timing requirements**: cause-effect chains, different types of jitter
- **Temporal isolation**: no starvation possible, temporal isolation between all tasks
- **Determinism/Predictability**: increased stability and testability, no unwanted run-time effects, many system properties become predictable, e.g., locks, task pre-emption
- **Re-simulation**: Equivalent behaviour between cloud and embedded target
- **Synchronization to communication**: stable real-time behaviour of cause-effect chains
- **Compositionality/Integration**: adding or modifying tasks can be done incrementally, system integration of SWCs from different suppliers is easier
- **Schedulability**: more correct configurations can be realized



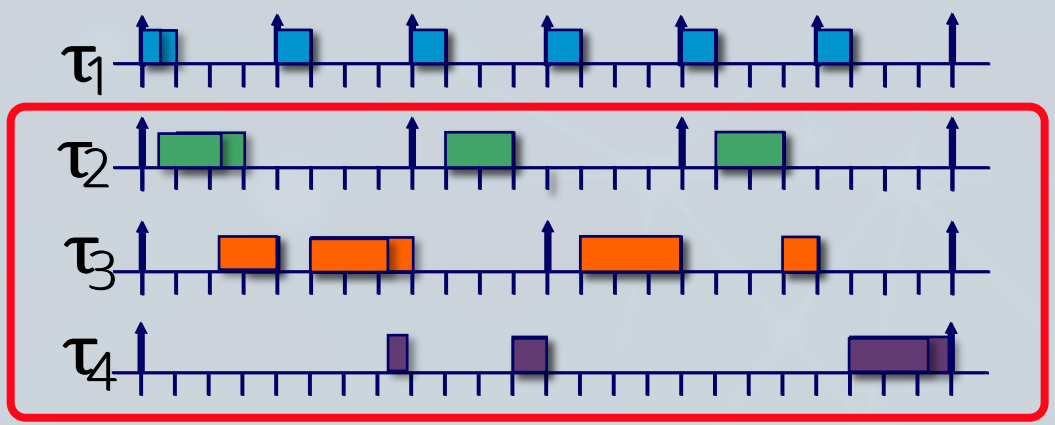
For a more in-depth comparison between FPS and TT please see [Xu2000]

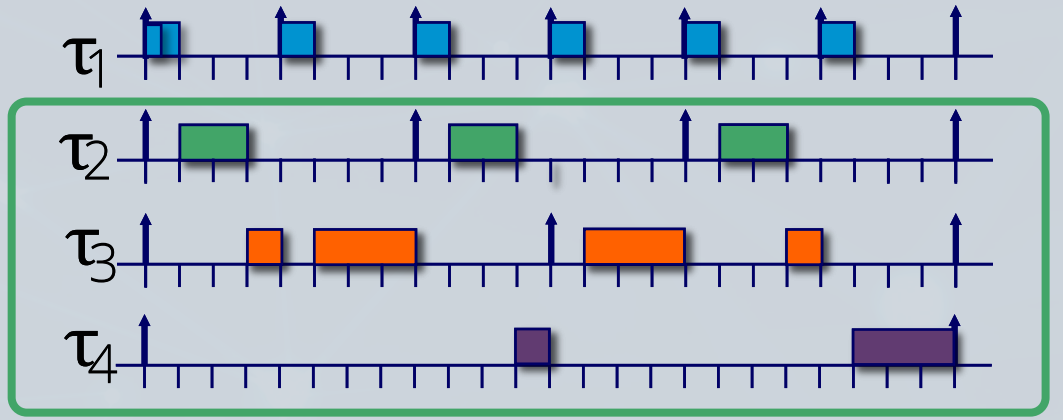
TT vs. FP/EDF

What happens if at some point in time, one task executes for less than the computed timing budget?

Non-TT (FP/EDF)
- changes the execution of all other tasks
- state change for the entire system

Time-Triggered
- execution of all other tasks remains the same
- no state change



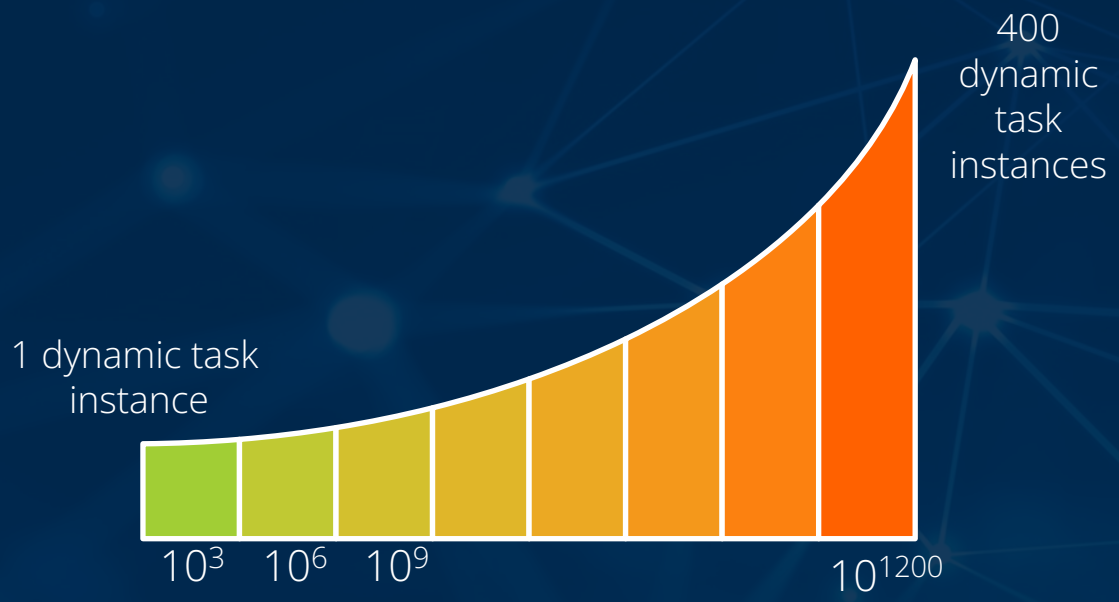


State space explosion

How many system states do I have to potentially check/validate?

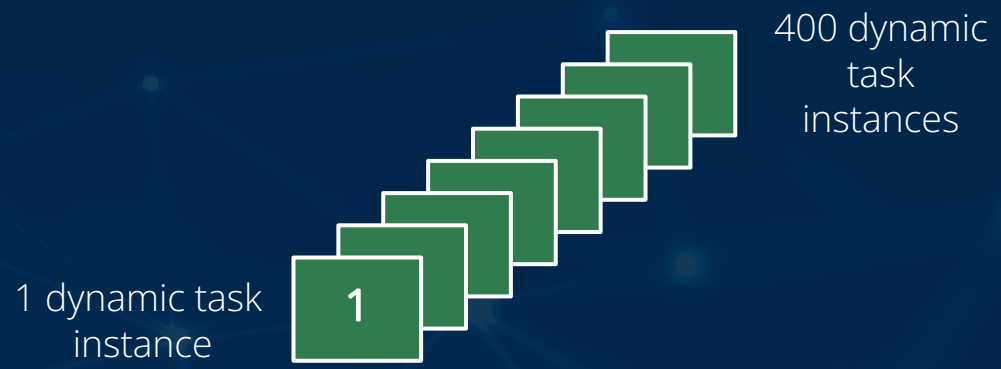
Non-time-triggered (FP/EDF/etc.):

- changes the execution of all other tasks
- state change for the entire system



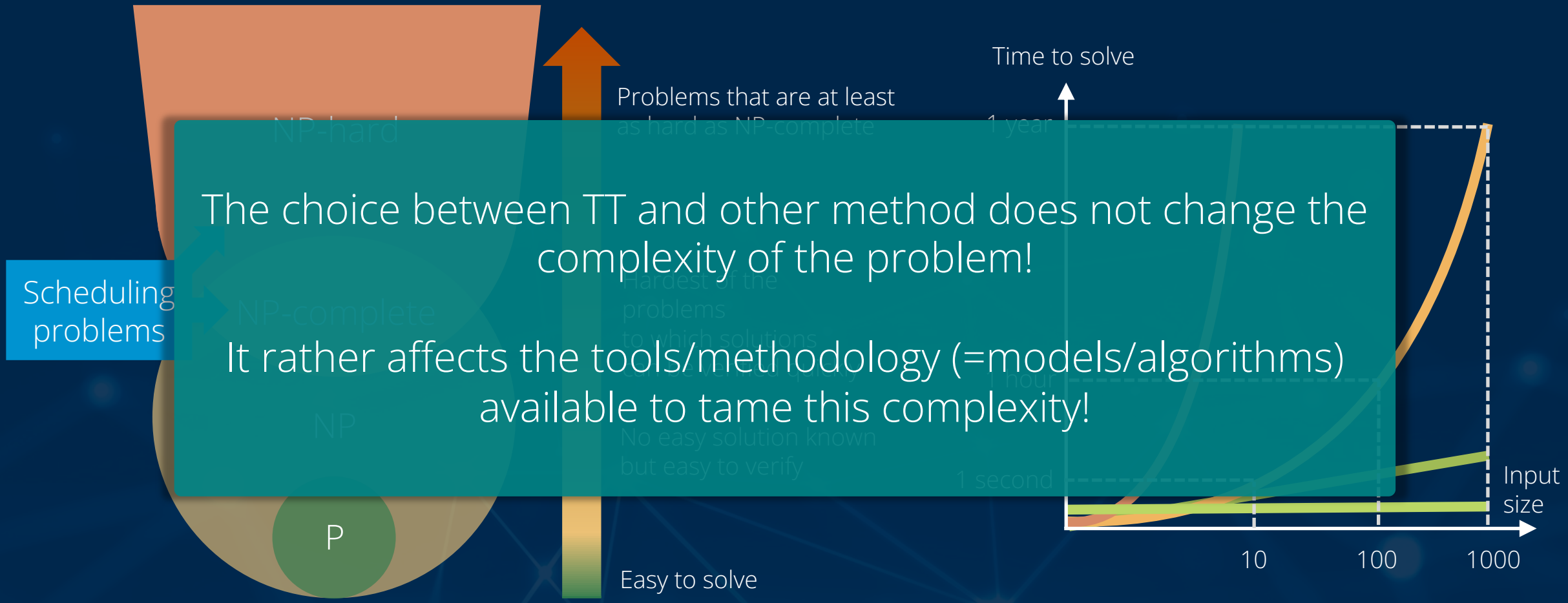
Time-triggered:

- execution of all other tasks remains the same
- no state change



BUT: Finding the static schedule table is still a very complex problem to solve

Complexity classes



The choice between TT and other method does not change the complexity of the problem!

It rather affects the tools/methodology (=models/algorithms) available to tame this complexity!

⚠ Complexity is not always equal to algorithm efficiency: compare 1.00001^n vs n^{50}

Finding a needle in a haystack the size of the universe

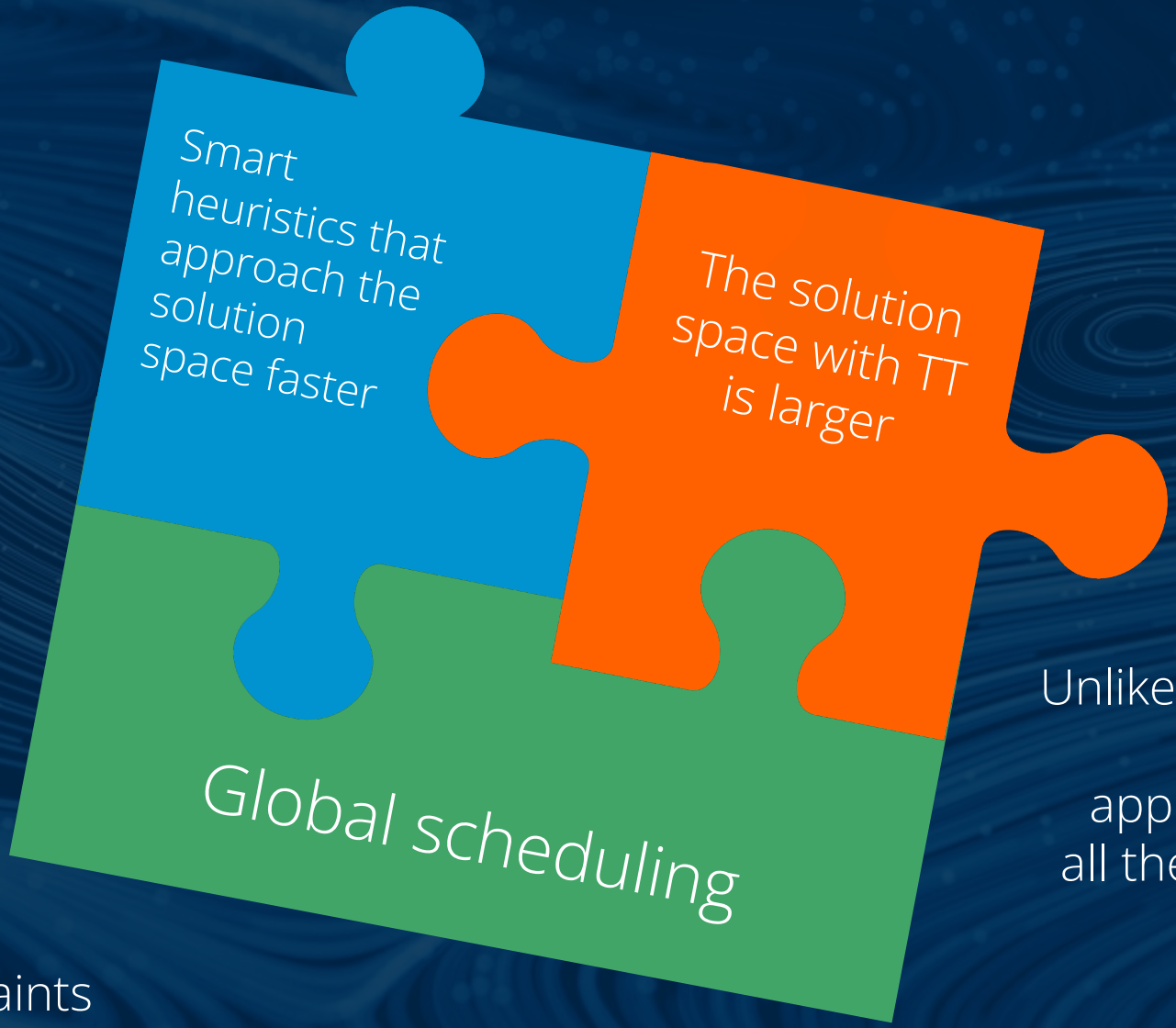
TTTech


Taming complexity

How do we tame the complexity

TTTech

Heuristic algorithms that significantly increase the probability of finding correct schedules



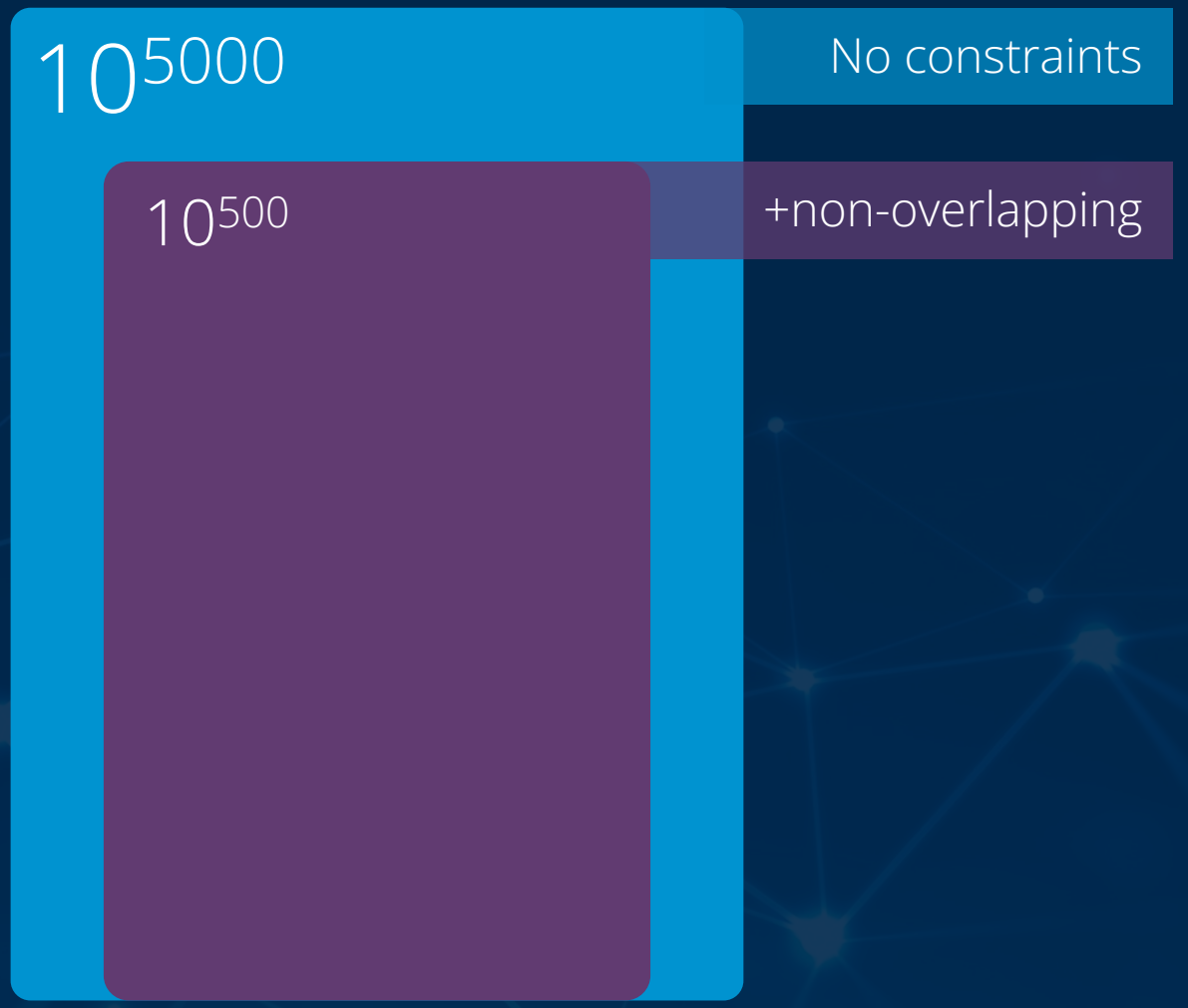
Global scheduling that has a holistic view of the system and constraints

Unlike other techniques, the time-triggered approach can express all the correct solutions (solution space)

Solution space wrt. different constraints

TTTech

- Fixed-priority (e.g., DM, RM)
- FP+ offsets
- Dynamic priority (e.g., EDF)
- Time Triggered

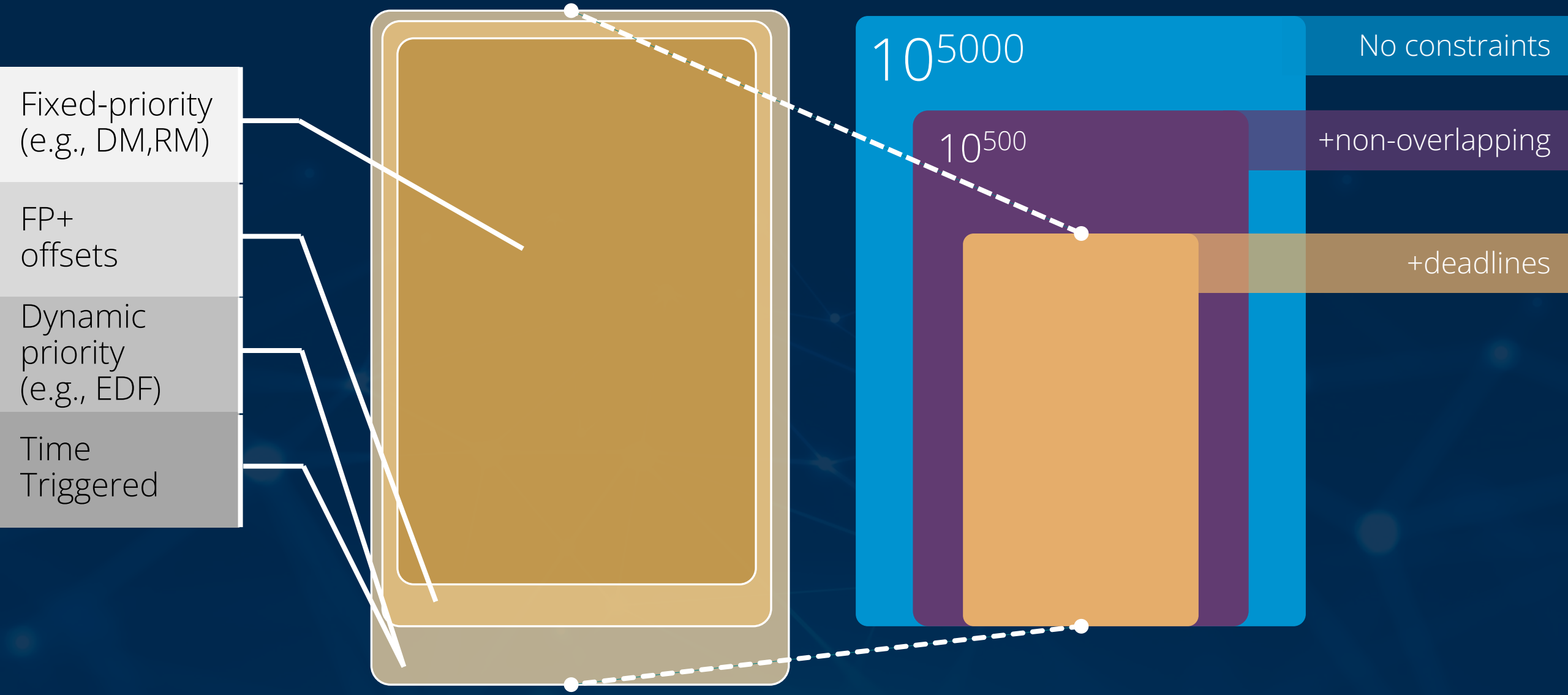


Atoms in the universe 10^{80}

Only TT guarantees the exact position in time.
Other paradigms assign the position at runtime.

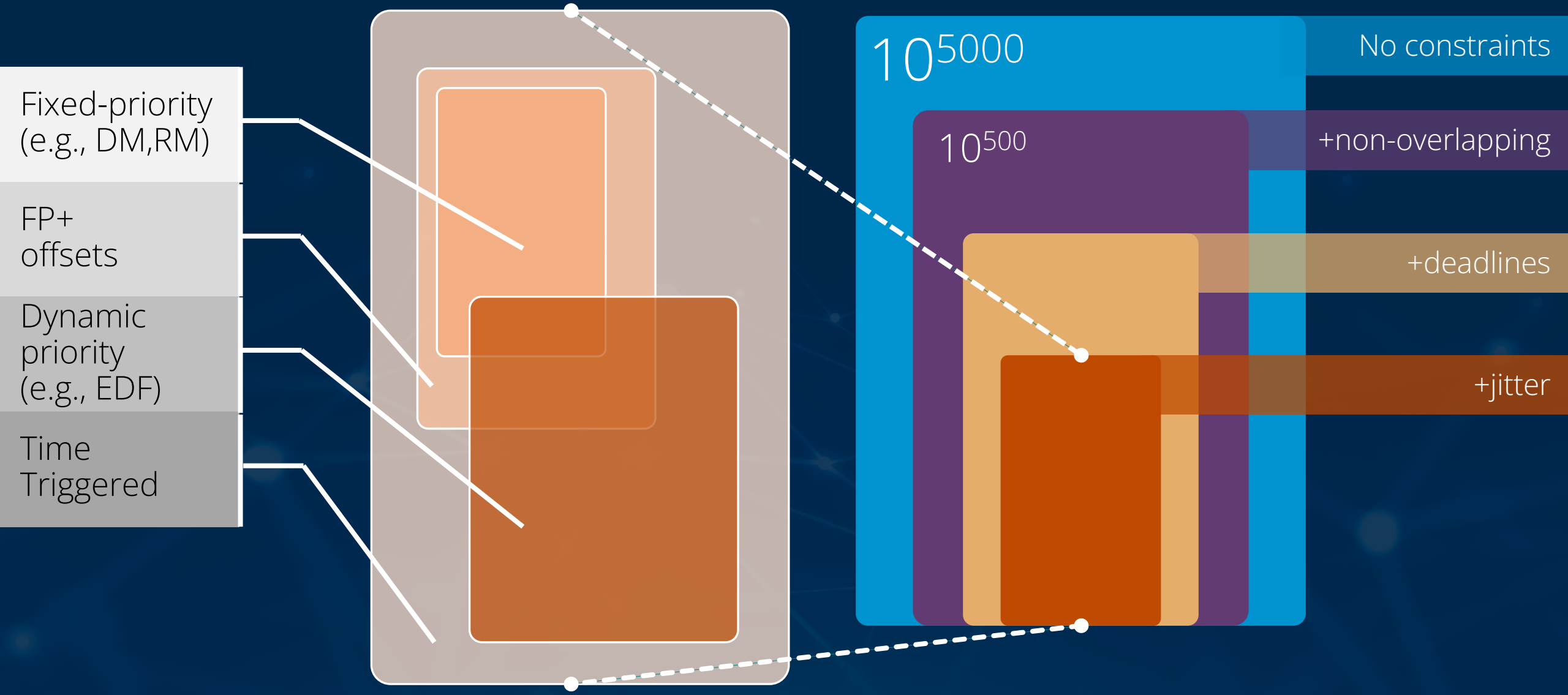
Solution space wrt. different constraints

TTTech



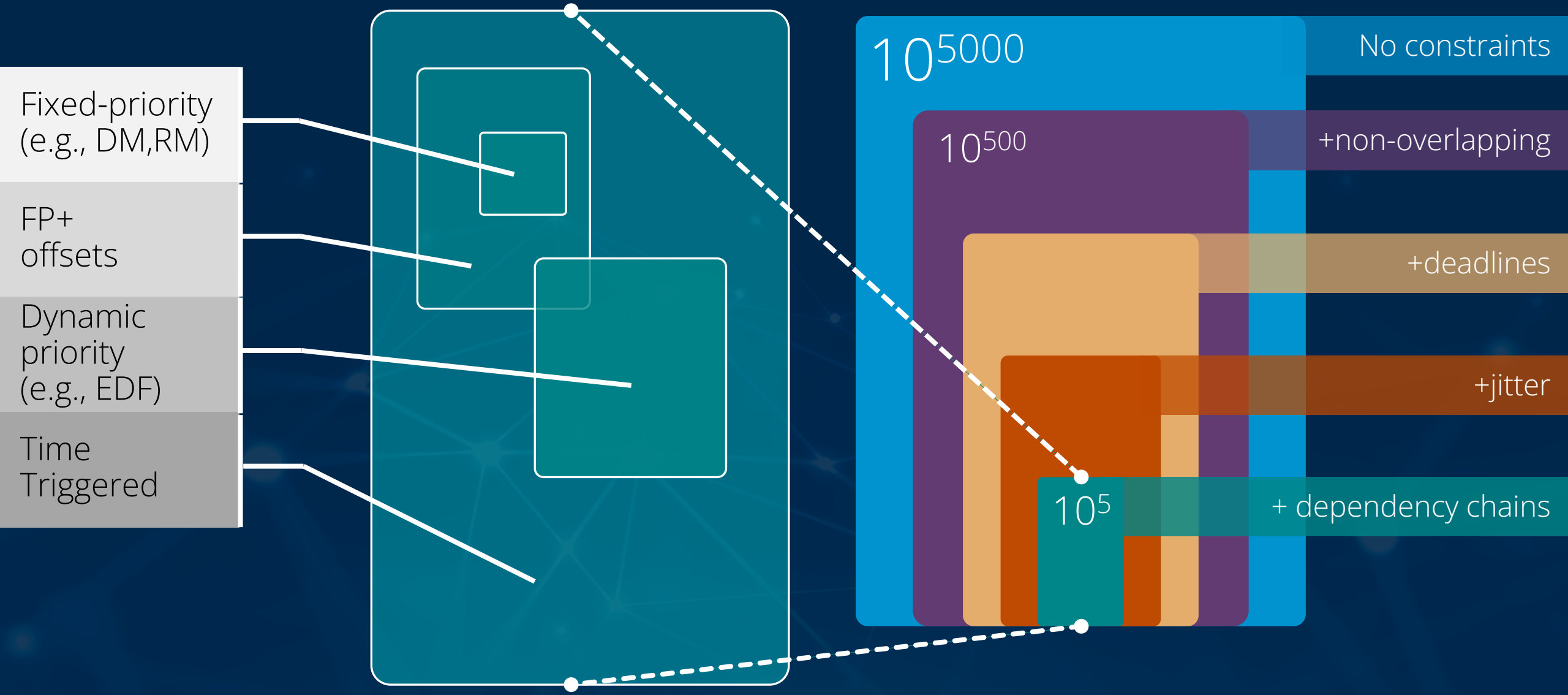
Solution space wrt. different constraints

TTTech



The time-triggered advantage

TTTech



Problem dimensions

TTTech

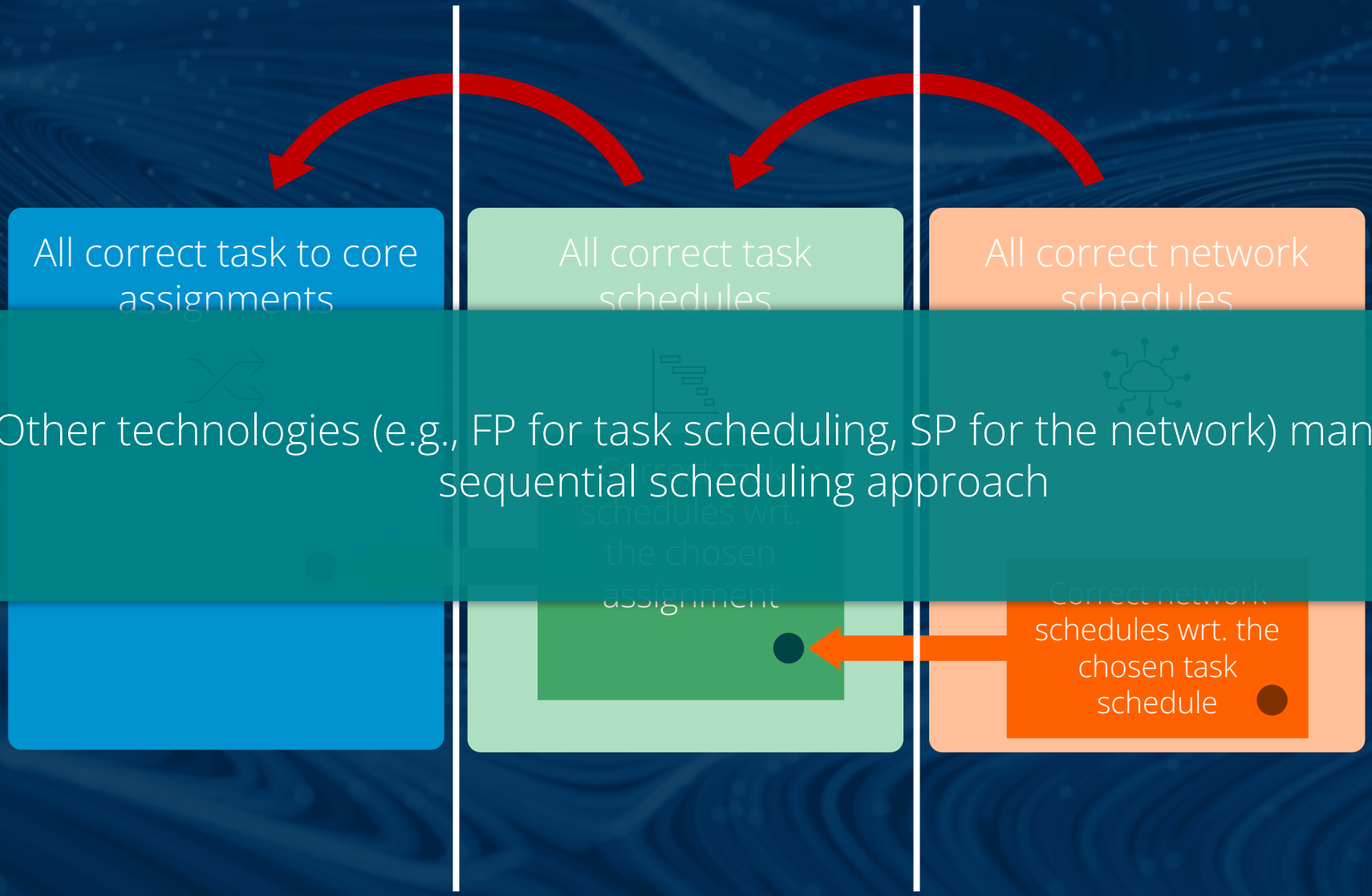
Task schedule

Hypervisor schedule

Task to core assignment

Network schedule

Sequential scheduling

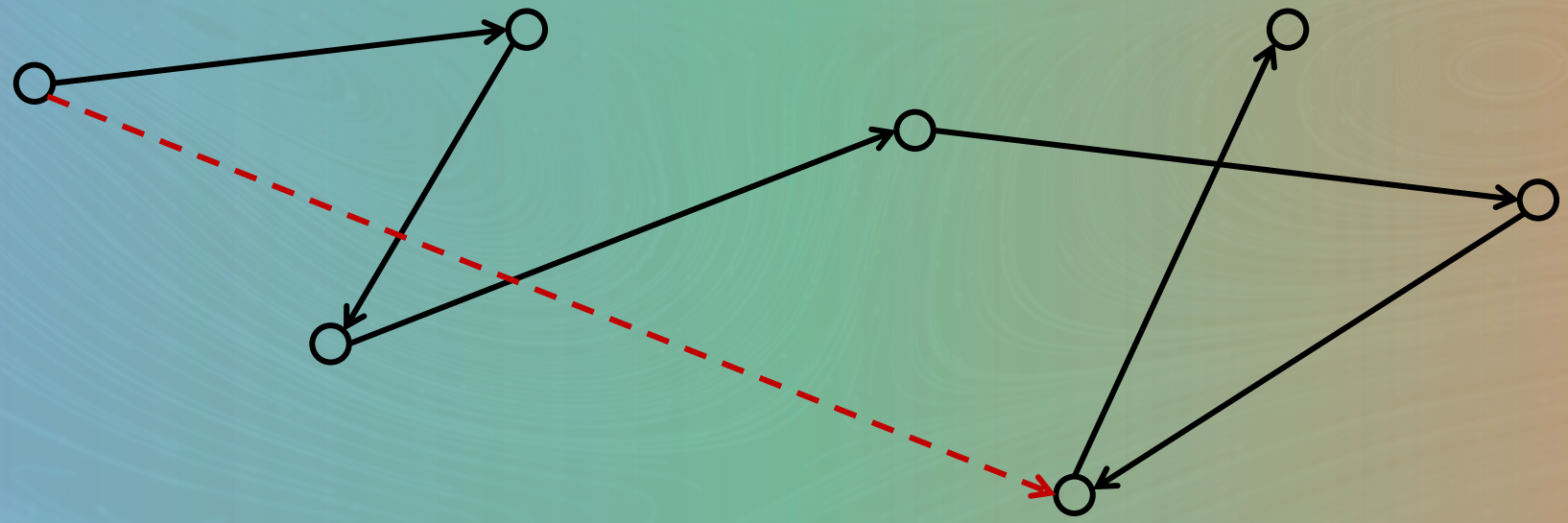


Other technologies (e.g., FP for task scheduling, SP for the network) mandate a sequential scheduling approach

Problem dimensions

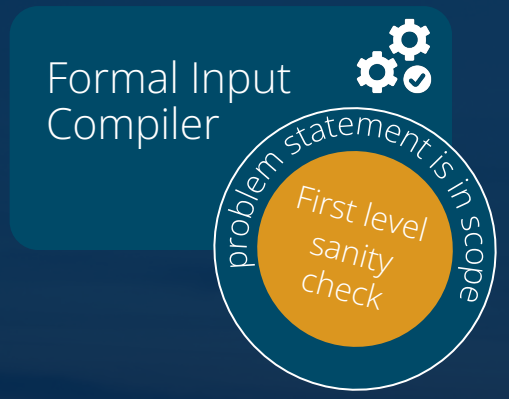
TTTech

All possible correct schedules



Global Scheduling allows us to seamlessly go from one problem dimension to another, do fine-grained improvements, and even find relations between candidate solutions across the different dimensions

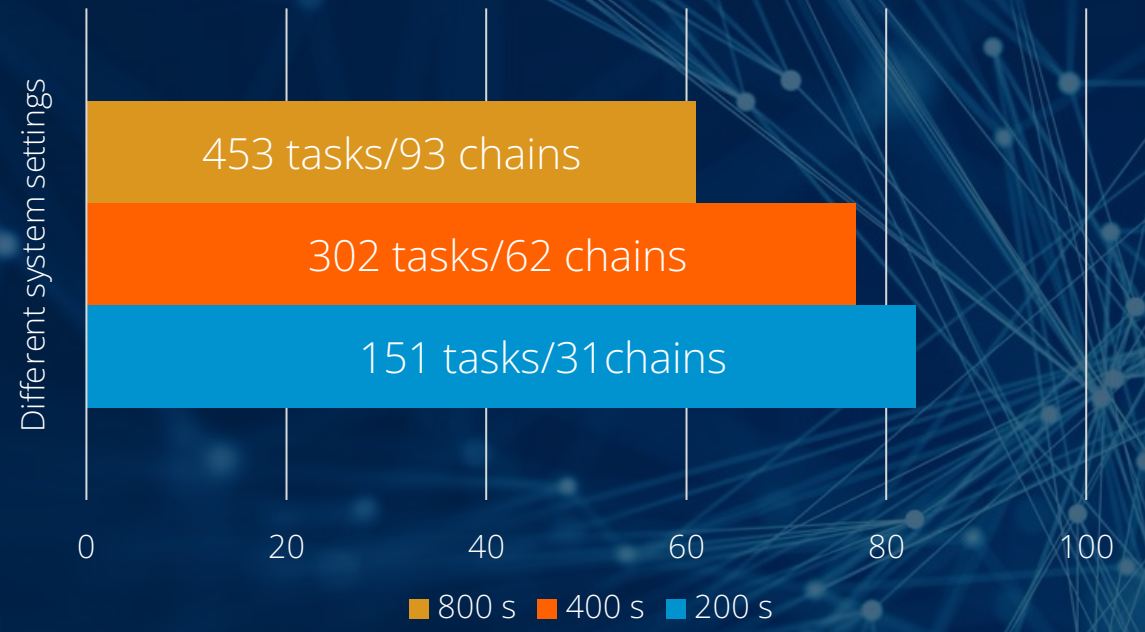
Global scheduler



Based on a rigorous formal model:

- non-interpretable correctness constraints expressed in mathematical language
- required for schedule correctness characterization and validation
- can be independently verified
- we have probably the most complete, rigorous, and comprehensive formal correctness model for task & network scheduling in the ADAS domain (~60 pages)

Smart heuristics



Smart heuristics that approach the solution space faster

S.D. McLean, S.S. Craciunas, E.A. Juul Hansen, and P. Pop - Mapping and Scheduling Automotive Applications on ADAS Platforms using Metaheuristics. In Proc. ETFA, IEEE, 2020.

Best paper award

Beyond time-triggered

There is no free lunch!

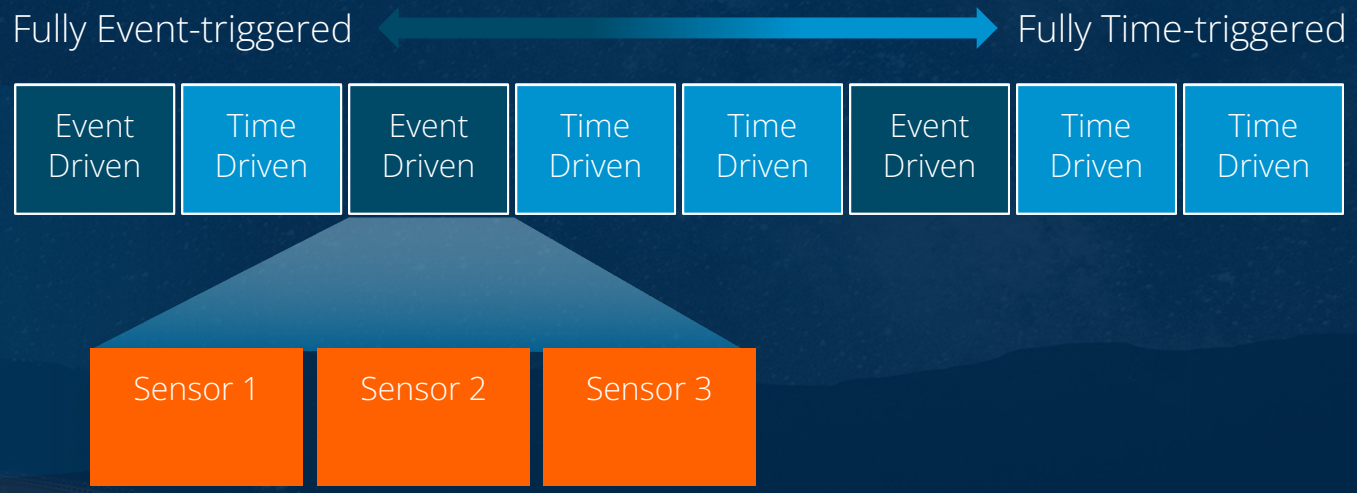
TTTechAuto

- Inflexible: no (big) adaptations at runtime, task execution is fixed
  - event-triggered tasks (sporadic activation)
- Overprovisioning: Slots must be reserved for the worst-case
  - WCET analysis is hard and very pessimistic!
- Precomputed: task properties must be known at design time
  - this is a development process issue!
- System limitations: macrotick, hyperperiod
  - not really a problem in practical systems!

- Event Driven tasks with guaranteed deadlines
- Data-driven (DAG sequencer) execution in reserved slots
- Slot shifting to maximize CPU time and minimize timing budget estimation problems
- Scheduling Modes to switch between different schedules depending on driving modes

As much time-triggered as needed, as much flexibility as possible

Time-triggered and event/data-driven integration



Second-level scheduling strategies based on requirements (e.g., FP)
Provide timing guarantees to critical event-driven tasks and optimize response times

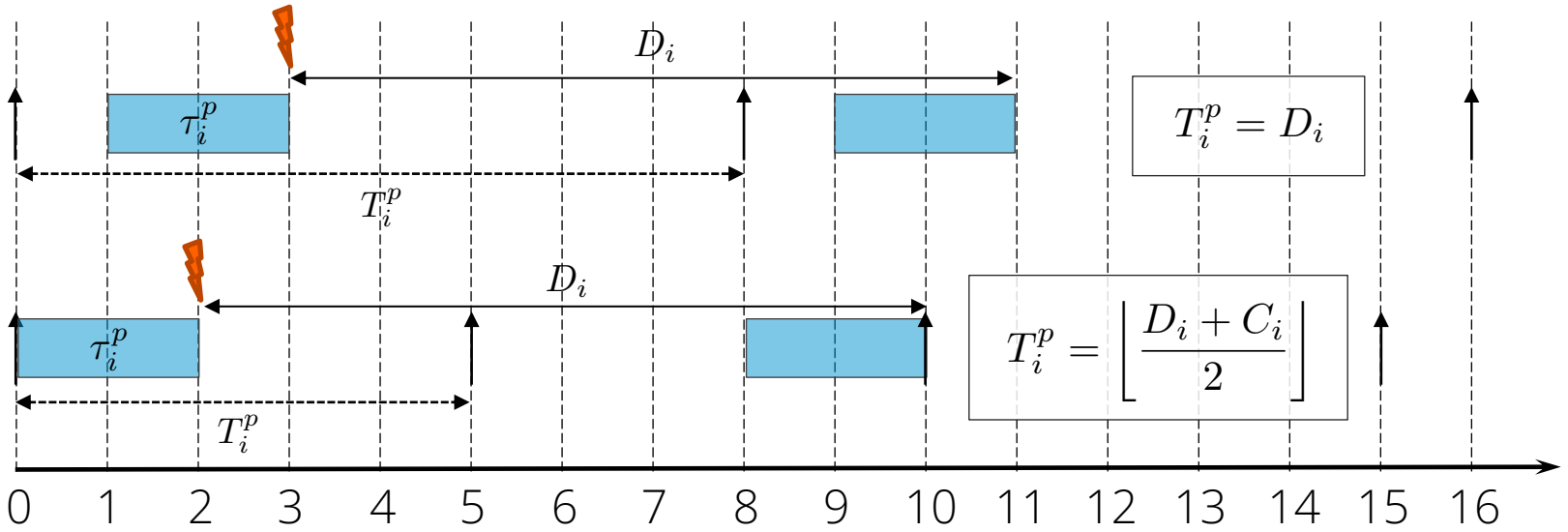
How can we ensure that both TT and ET tasks respect their deadlines?

Oversampling – SPoll

Simple polling: Use one periodic polling task per event – oversampling
How long is the sampling period?

a) Strictly periodic

b) Non-strictly-periodic



⚠ Can be quite inefficient!

Think of an event with C = 2 ms, D = 20 ms, and T = 100ms. (2% utilization)
We have to reserve a slot every 9 ms, consuming 22,2 % of CPU bandwidth.

Solution using response-time analysis

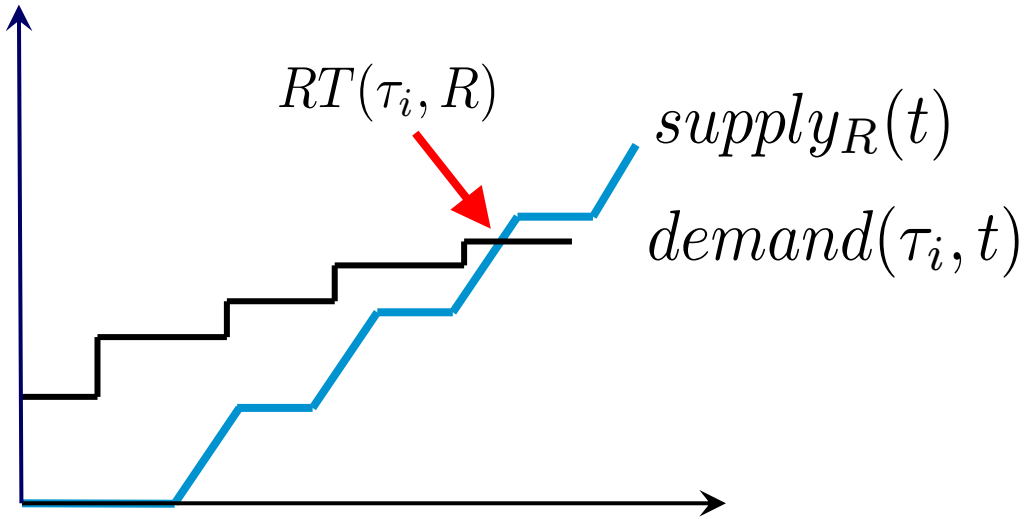
Holistic scheduling [Pop2003]

1. Generate TT schedule such that TT tasks are fulfilled.
2. Check schedulability over resulting idle slots using the response-time method over partitioned resources for every event-based task.
3. Recompute TT schedule if ET tasks not feasible

Slot-shifting [Isovic2009] is a similar method with a different schedulability test that does not assume FP scheduling.

Can be very time-consuming!
No guidance on how to place idle slots.

? Can we do better?



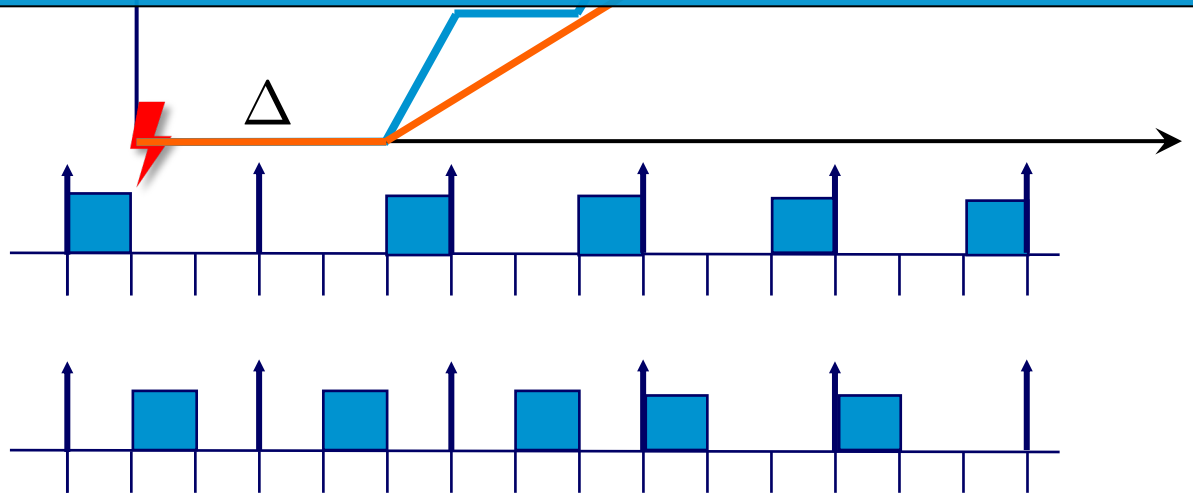
Solution using periodic resource abstraction

Explicit Deadline Periodic: R(C,D,T) [Shin2008]

- we do not care when the supply C is given, as long as it is given in every period T until the deadline D
- we can use the worst-case linear supply bound and schedulability test defined below [Almeida2004]
- New method: AdvPoll [Meroni2023]

- Schedulability analysis for ET is independent of the TT schedule
- The polling task (periodic resource) can be scheduled as a normal TT task

Tradeoff – schedulabiliy for runtime



$$lsbf(t) = max\{0, (t - \Delta) \cdot \alpha\}$$

$$RT(\tau_i, R) = earliest\ t :$$
$$t \geq \Delta + demand(\tau_i, t)/\alpha$$

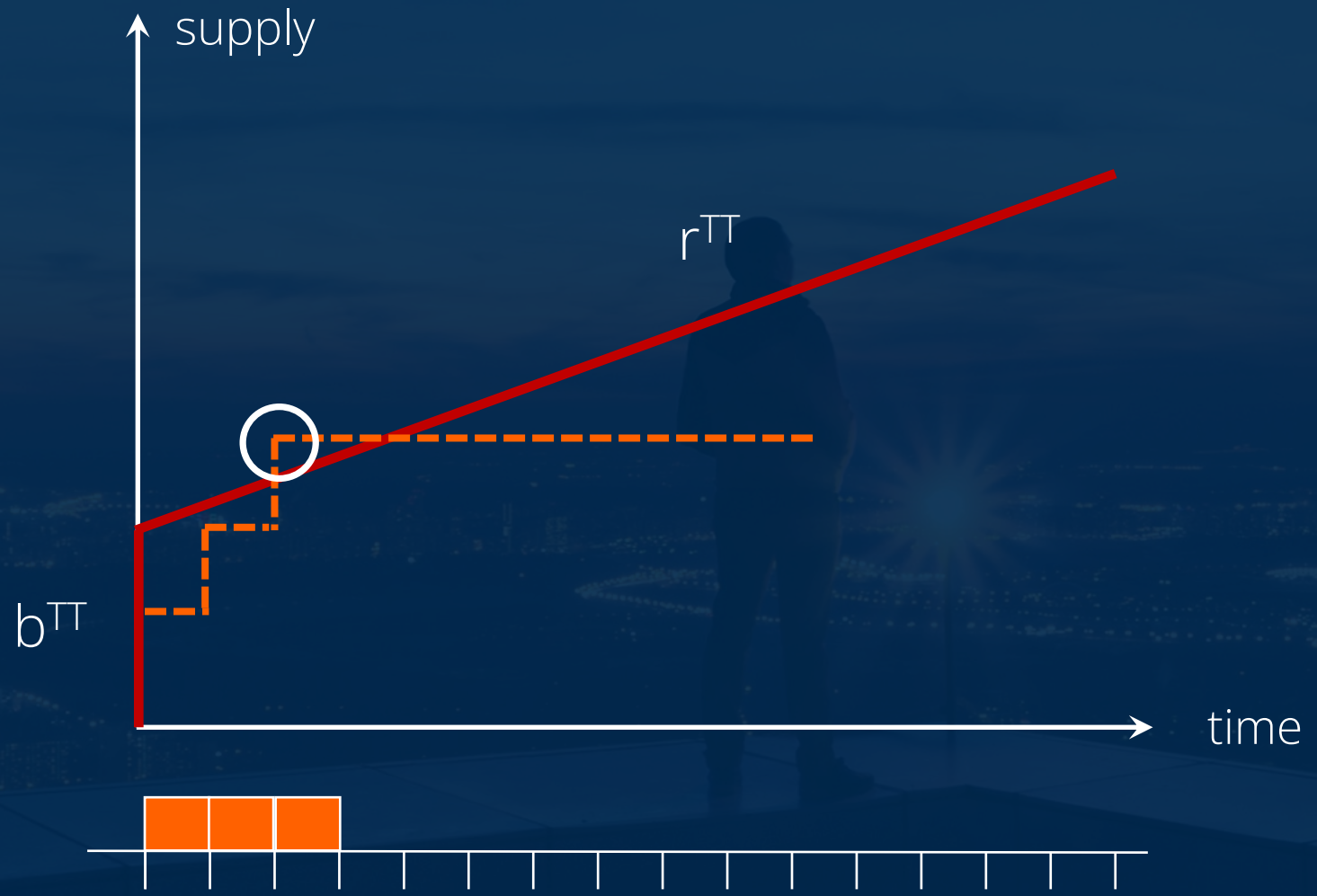
$$RT(\tau_i, R) = earliest\ t : t \geq \Delta +$$
$$\frac{1}{\alpha}\left(C_i + \sum_{\forall \tau_j \in HP(\tau_i)} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j\right)$$

Integration of ET + TT via real-time calculus

Main idea: derive a constraint on the TT schedule that will guarantee ET task schedulability & create a TT schedule that respects that constraint

A. Finzi, S.S. Craciunas, M. Boyer – Integrating Sporadic Events in Time-triggered Systems via Affine Envelope Approximations. In Proc. RTAS 2024

Outstanding paper award

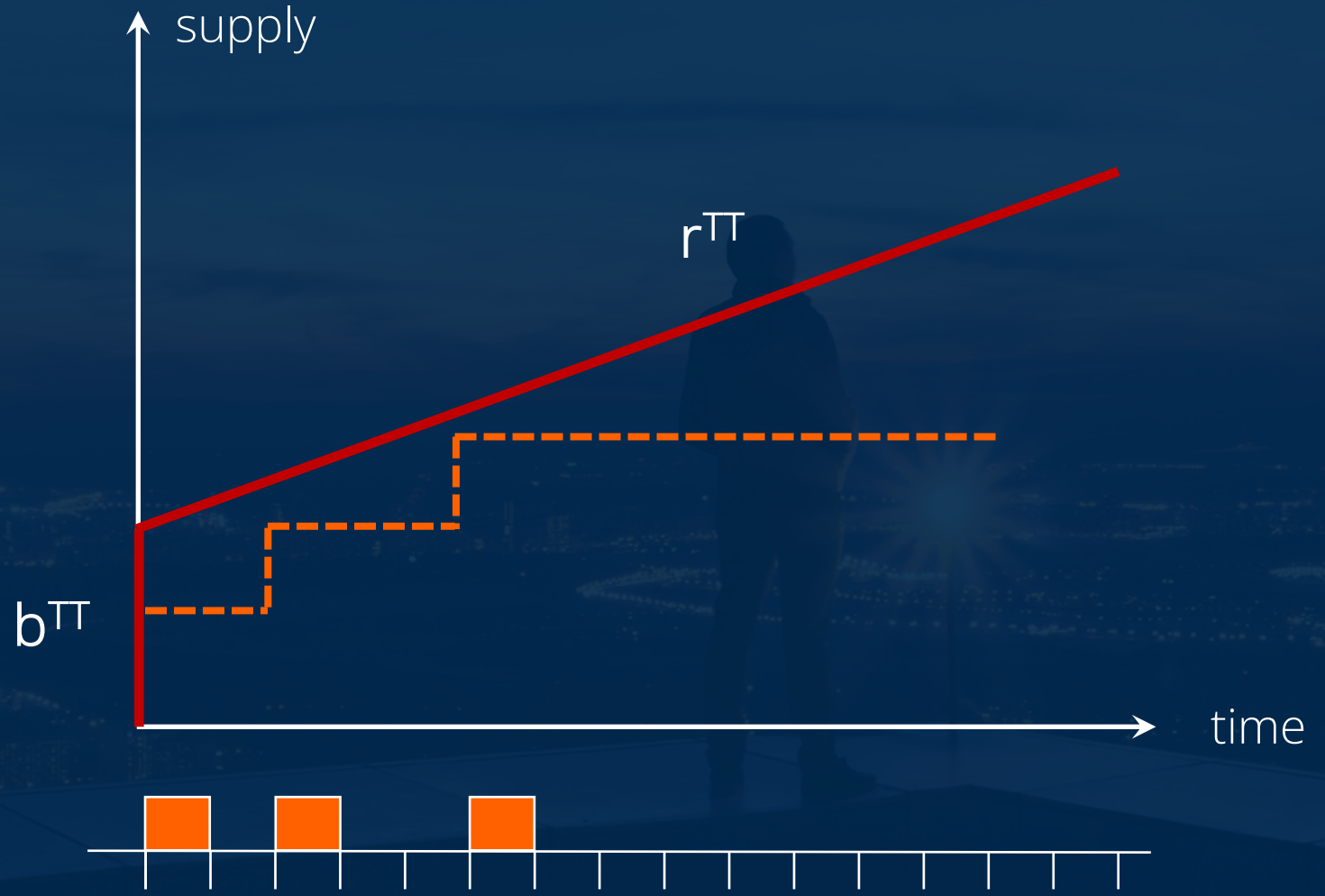


Integration of ET + TT via real-time calculus

- The cumulative computation time of TT must remain under the (b^{TT}, r^{TT}) constraint

Challenges:

- How to calculate the (b^{TT}, r^{TT}) constraint to enforce ET schedulability?
- How to compute an offline schedule remaining under the (b^{TT}, r^{TT}) constraint?



How to calculate the (b^{TT}, r^{TT}) constraint?

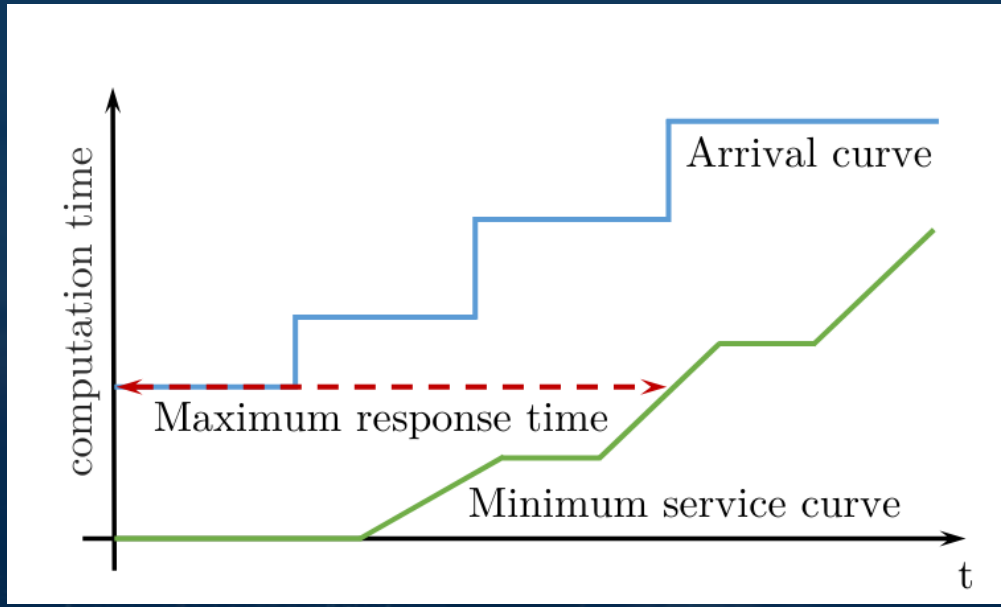
TTTech

Real-Time Calculus:

Framework used to calculate the **maximum response time** of a task.

Arrival curves: describe how a set of tasks arrives in any time interval

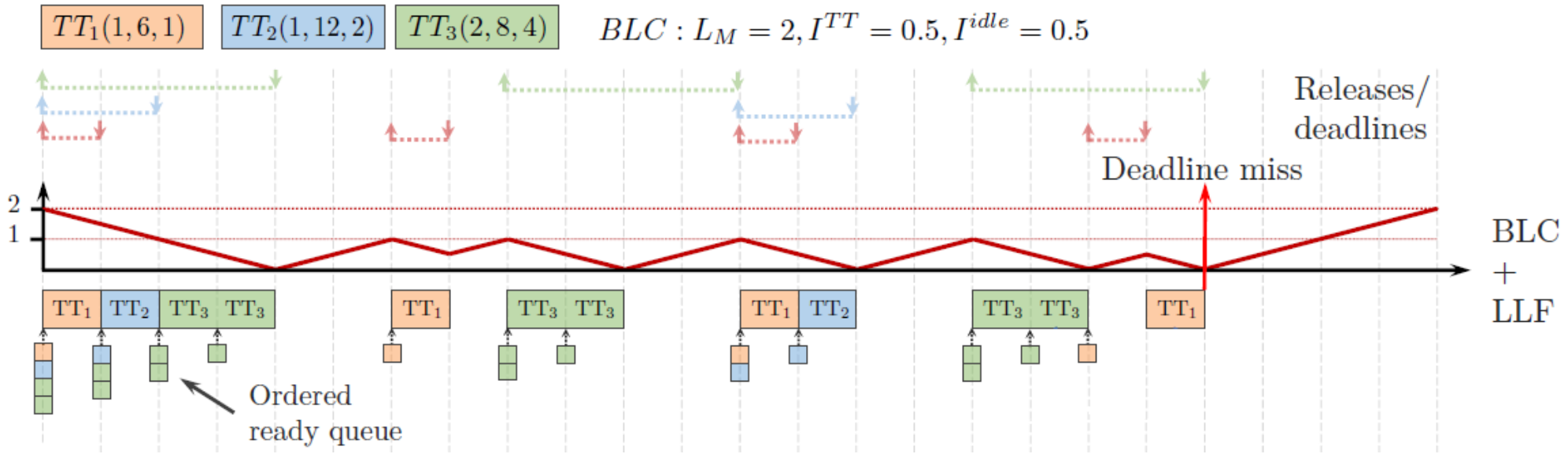
Service curves: describe the service offered by a CPU to a set of tasks in any time interval



- Maximum horizontal distance is the maximum response time (MRT)
- ET schedulability: MRT ≤ deadline

- (r^{TT} ,b^{TT}) are parameters of the ET service curves
- r^{TT} is the sum of the rates of the set of TT tasks
- The maximum b^{TT} such as the ET tasks are schedulable is computed in a binary search

How to compute a schedule remaining under the (b^{TT}, r^{TT}) constraint?



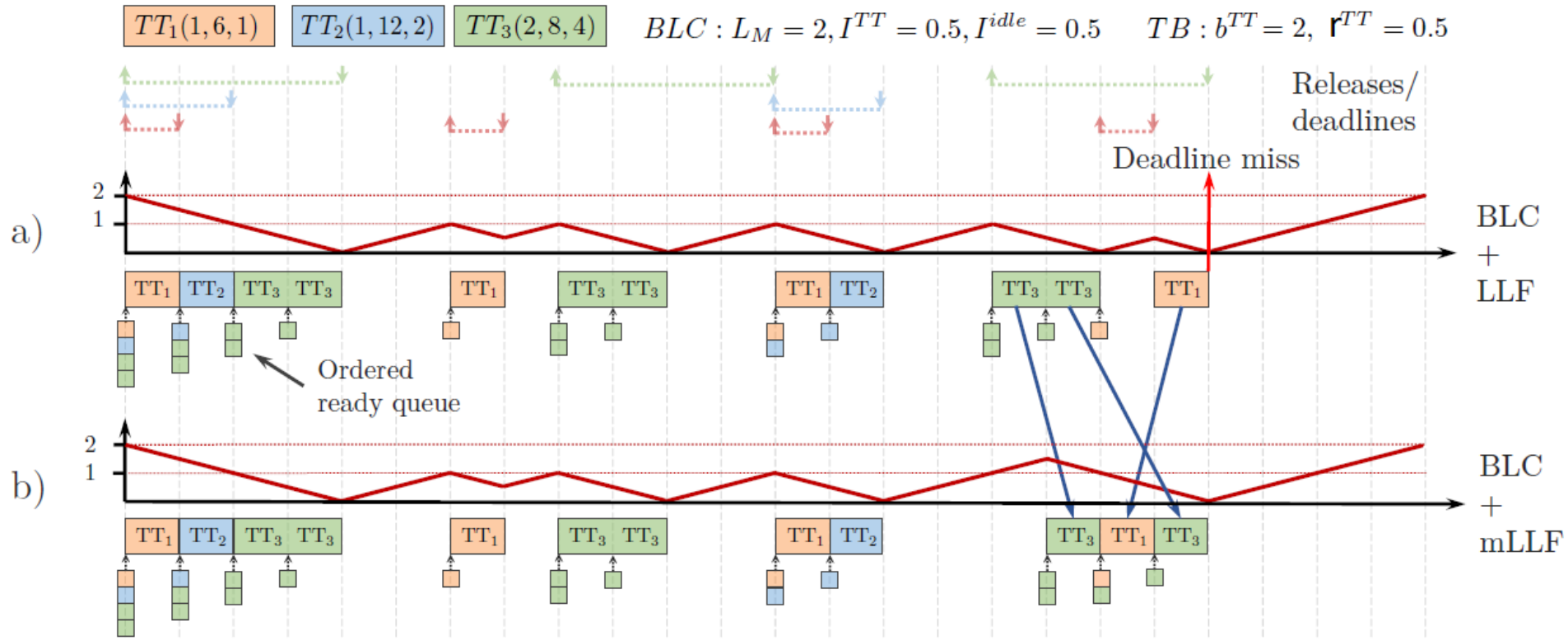
ET schedulability is enforced, but to the detriment of TT schedulability

- We need to balance ET and TT schedulability
  - We add "idle" tasks in the LLF, with Laxity:

$$L_{IDLE}(t) = \begin{cases} \left\lfloor \frac{bdg_\sigma(t)}{I^{TT}} \right\rfloor & \text{if } bdg_\sigma(t) < L_M - I^{idle} \\ \infty & \text{otherwise} \end{cases}$$

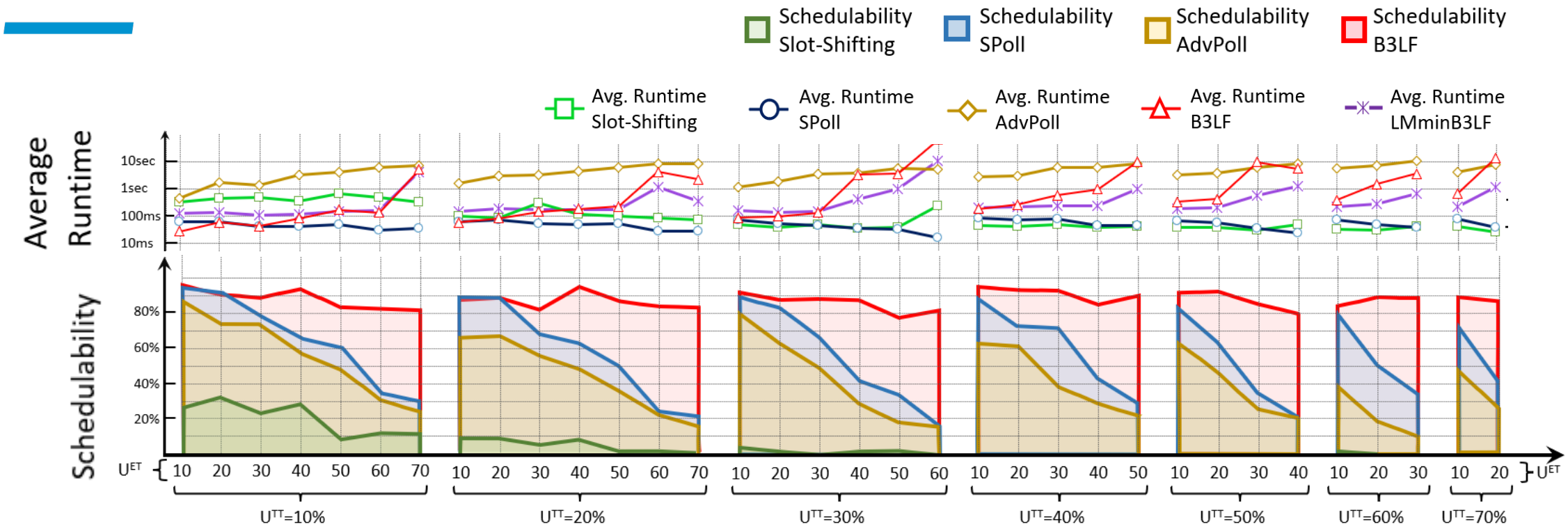
How to compute a schedule remaining under the (b^{TT}, r^{TT}) constraint?

With mLLF + BLC, we can find a valid TT schedule enforcing the ET schedulability



Experiment results:

With single core CPU (or a core in a fully partitioned multicore system)
U^{TT} and U^{ET} are the respective utilizations of the TT and ET task sets



Schedulability and average runtime with $100\mu s$ macrotick, ET and TT task periods $T_i \in \{1, 2, 5, 10, 20, 50, 100, 200, 1000\}ms$ and arbitrary ET deadlines in $[C_i, 5 \cdot T_i]$.

Schedulability of ET and TT tasks with B3LF is better than the other solutions

Thank you!

[Xu2000] Xu, Parnas - Priority Scheduling Versus Pre-Run-Time Scheduling. Real-Time Syst, 2000

[Pop2003] T. Pop, P. Eles, Z. Peng - Schedulability analysis for distributed heterogeneous time/event triggered real-time systems, ECRTS 2003.

[Isovic2009] D. Isovic, G. Fohler - Handling mixed sets of tasks in combined offline and online scheduled real-time systems, Real-Time Syst, 2009.

[Shin2008] I. Shin et al.- Hierarchical Scheduling Framework for Virtual Clustering of Multiprocessors. ECRTS, 2008

[Almeida2004] L. Almeida, P. Pedreiras - Scheduling within temporal partitions: Response-time analysis and server design. EMSOFT, 2004

[Meroni2023] C. Meroni, S.S. Craciunas, A. Finzi, and P. Pop - Mapping and Integration of Event- and Time-triggered Real-time Tasks on Partitioned Multi-core Systems. In Proc. ETFA, 2023.

[Finzi2024] A. Finzi, S.S. Craciunas, M. Boyer – Integrating Sporadic Events in Time-triggered Systems via Affine Envelope Approximations. In Proc. RTAS, 2024

TTTech

We are hiring!

https://www.tttech.com/jobs-career/