RTSOPS 2026

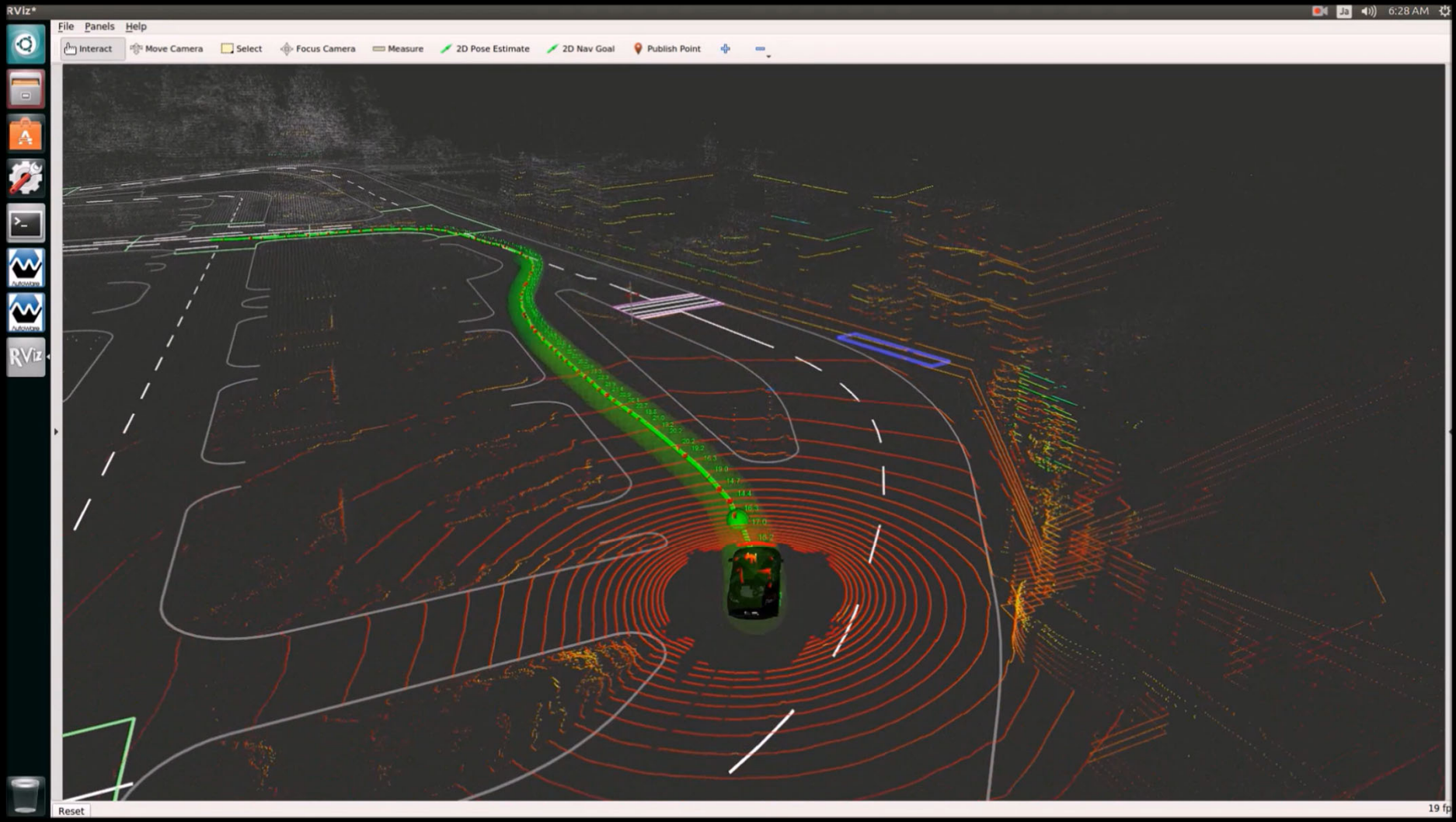
Toward the Right Analytical Model and System Software for Autonomous Driving Systems

Open Problems and Research Directions

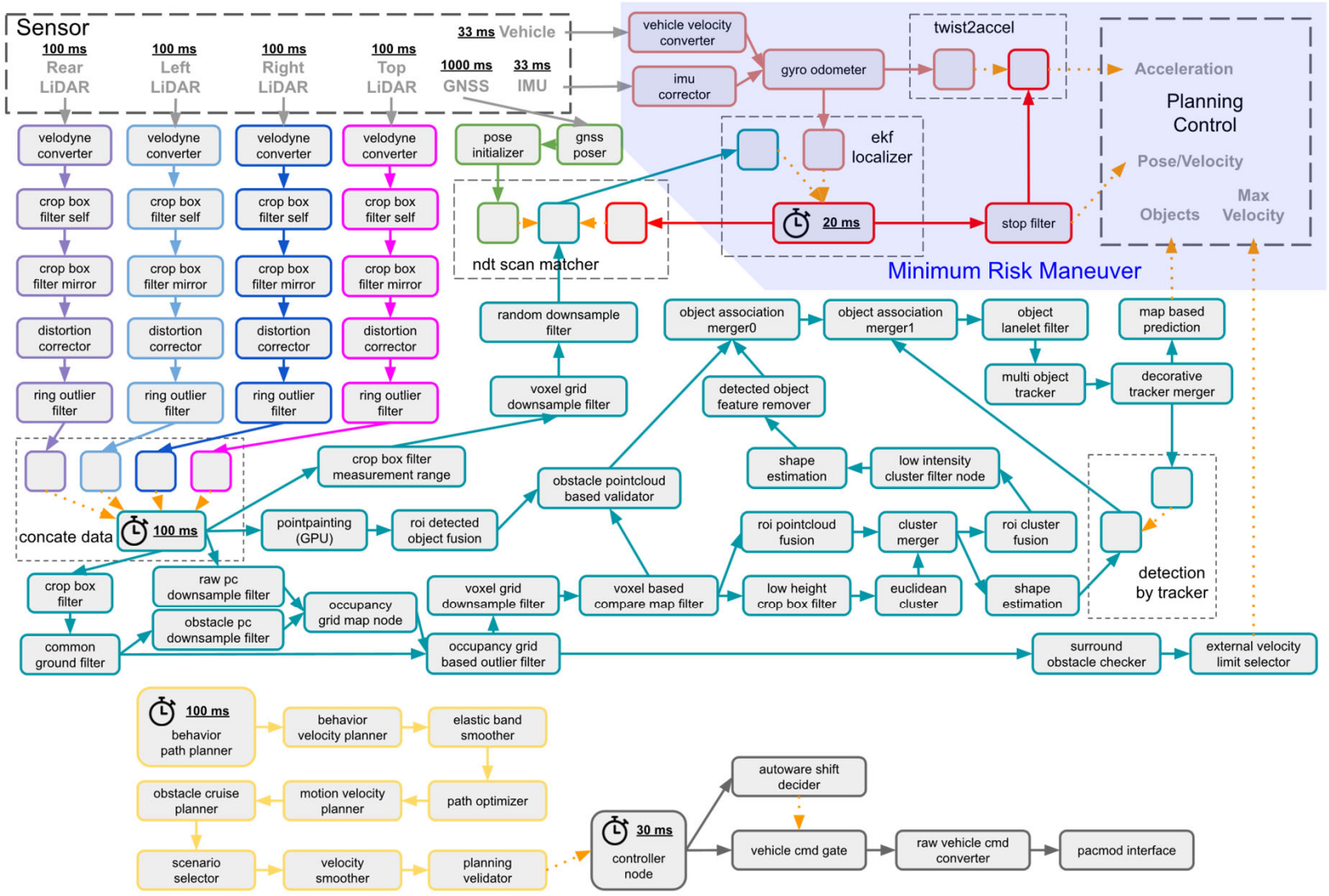
Atsushi Yano and Takuya Azumi

Graduate School of Science and Engineering, Saitama University, Japan
TIER IV Incorporated, Japan

Autoware on DrivePX2

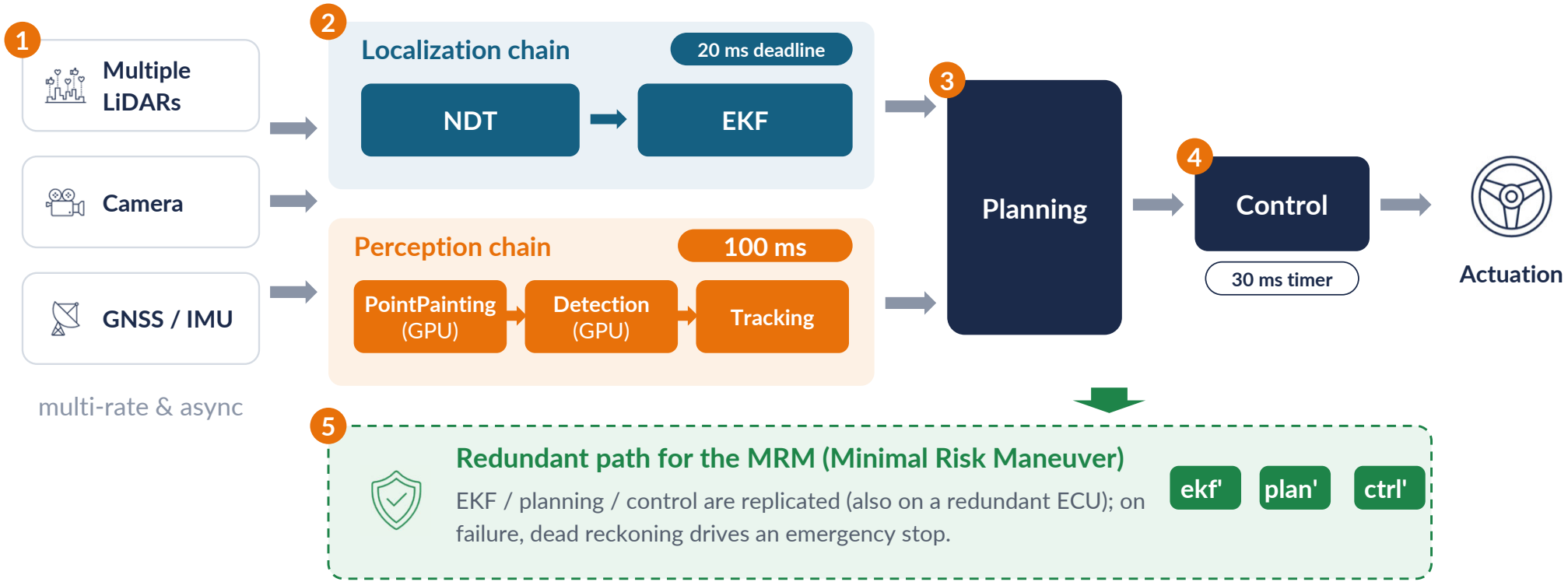


(Presenter: Takuya Azumi) Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and **Takuya Azumi**,``**Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems**,'' *In Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS2018)*, Porto (aka Oporto), Portugal, Apr. 2018.



Atushi Yano and Takuya Azumi, “Work-in-progress: Multi-deadline DAG scheduling model for autonomous driving systems,” in *Proc. of the 45th IEEE Real-Time Systems Symposium (RTSS), Work-in-Progress Session*, 2024, pp. 451–454

AD breaks the task abstraction: callbacks form a graph



① multi-rate sensors ② parallel chains ③ converge at planning ④ control → actuation ⑤ MRM redundant path

Sensors arrive at different rates

A graph, not a pipeline

Safety relies on fallback paths

G1 The schedulable unit is not the safety unit

Classical view

One well-defined task

task + deadline

- the deadline is on the task
- the scheduler dispatches the task
- the proof is about the task

AD reality

Schedulable unit — the callback

cb1 cb2 cb3 cb4 ...

the units do not coincide

Safety unit — the E2E cause-effect chain

Sensor → Perception → Planning → Control

one continuous behavior of a vehicle function

E2E latency, freshness, and the chain deadline attach to the chain / graph — not to any single callback

Essence The callback is schedulable; the chain is safety-relevant — no unit carries both meanings yet → TP1

G2 A met deadline can still use bad timing information

Classical view

Did we meet the deadline?

- callback 1 ✓
- callback 2 ✓
- callback 3 ✓

→ all “in time”, so OK…?

⚠ decisions may rest on stale data and unstable outputs

AD reality — four more metrics that must hold

freshness
age of information

how old is the data behind each output?

F = t_out − t_data

disparity
timestamp gap between sensors

LiDAR and camera do not necessarily see the same instant

jitter
variation of the output period

an unstable period destabilizes control and perception

E2E latency
end-to-end delay

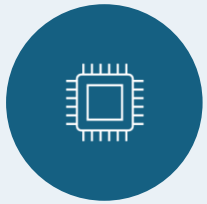
is the response still physically meaningful for a moving car?

Deadlines met ✓ — yet timing quality remains an entirely separate problem

Essence No framework yet treats all these timing metrics jointly, within one analysis → TP2

G3 Timing is shaped by GPU, memory, and communication

Classical view



execution time ≈ CPU time

the CPU is the main resource;
GPU, communication, and memory
are abstracted away

VS

AD reality — five interacting resources

CPU

GPU

NPU

Memory

Comm / IPC

contention across resources

Perception includes GPU / NPU inference

Communication / IPC adds to E2E delay

Memory bandwidth is contended

Essence The timing model must include heterogeneous resources — and their contention

→ TP3

G4 Execution time is a distribution, not a fixed number

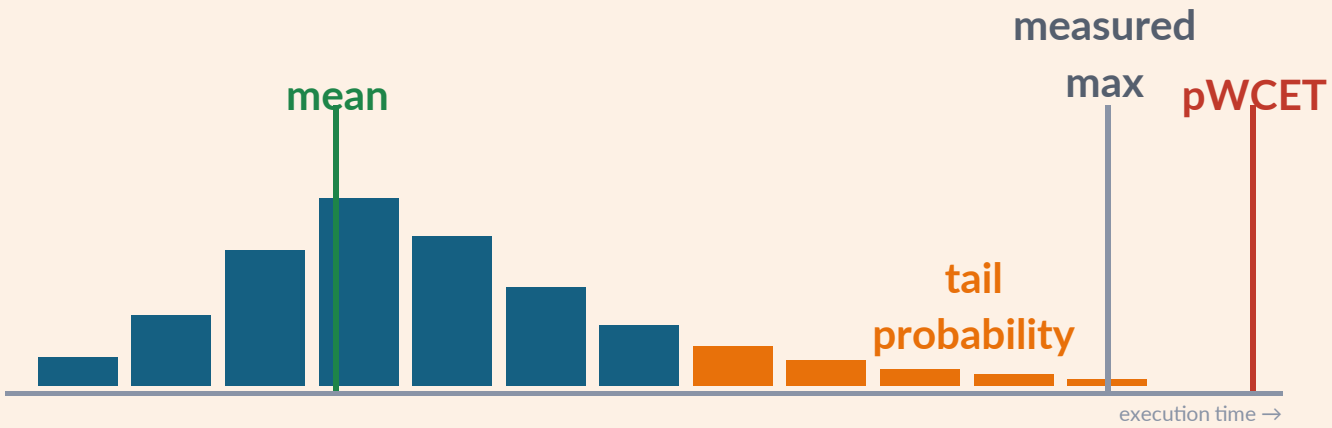
Classical view

One fixed number

C = WCET

- a known WCET
- variation assumed small
- one bound fits every scene

AD reality

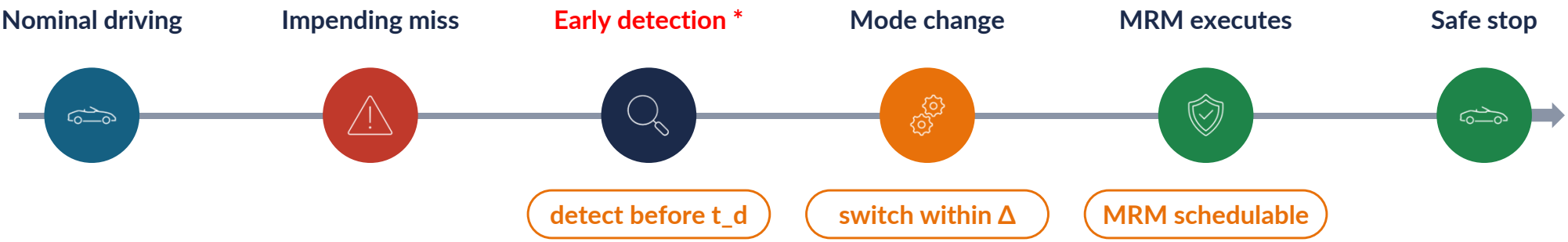


Essence Execution time is a distribution with a long tail — bound the tail probabilistically (pWCET) → TP4

G5 A miss must trigger a guaranteed safety sequence

Classical view a deadline miss is just a violation — safety is handled elsewhere, after the fact

AD reality — the fail-safe sequence

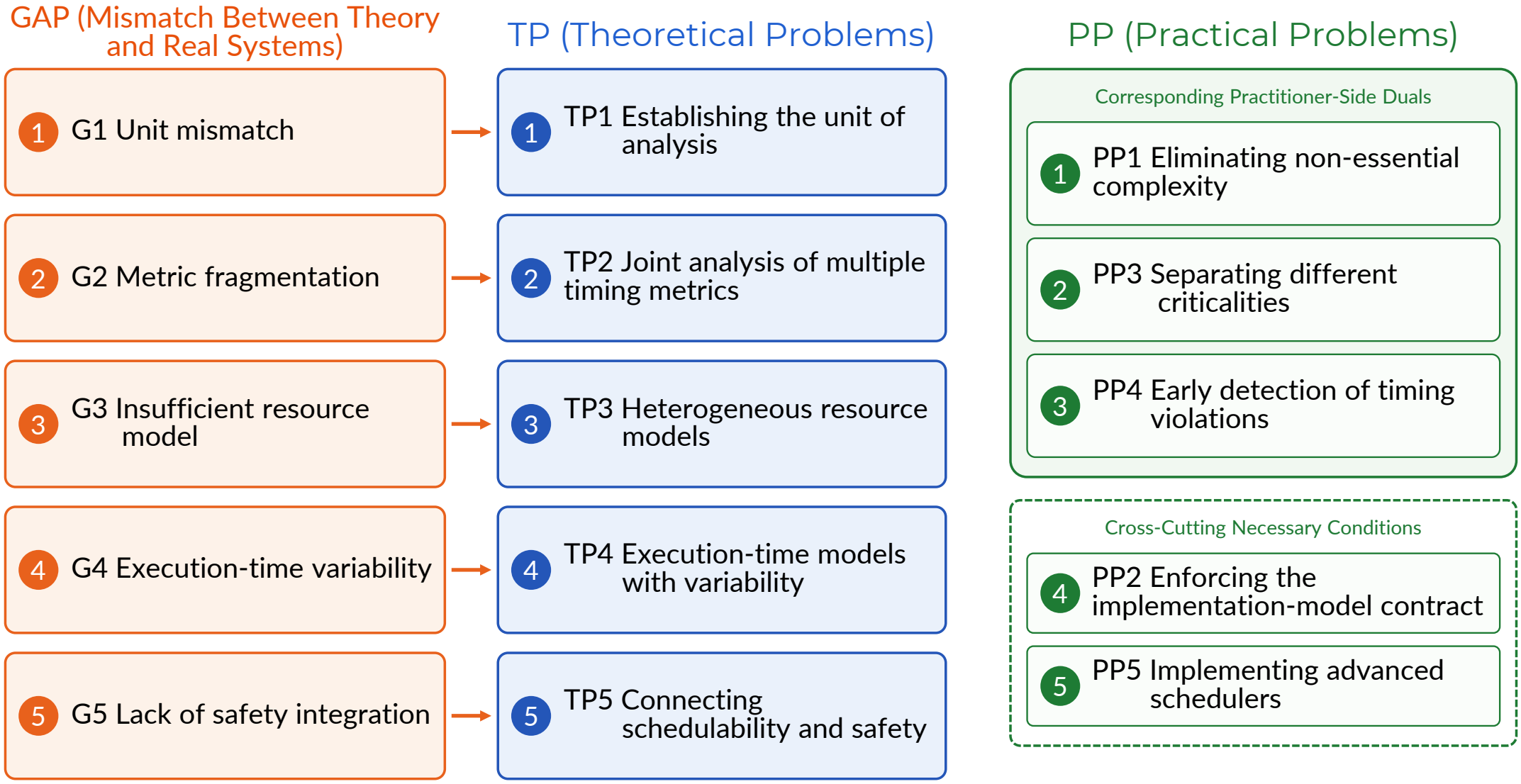


The entire sequence needs timing guarantees

Hayate Toba, Sanjoy Baruah, and **Takuya Azumi**, **Enhancing Deadline Miss Early Detection for DAG Tasks with Variable Probabilistic Thresholds,** ACM Transactions on Embedded Computing Systems, 2026 *(extended version of the paper presented at ECRTS 2024)*

Essence Safety must be inside the schedulability analysis: detection → switch → MRM, guaranteed as one → TP5

Relationships Among GAP, TP, and PP



OPEN PROBLEMS · PART 1

Open Problems: For Theorists

What task / timing **model** is still missing?

IV

Five problems TP1 to TP5,
one per gap G1 to G5

Five gaps, five problems for theory

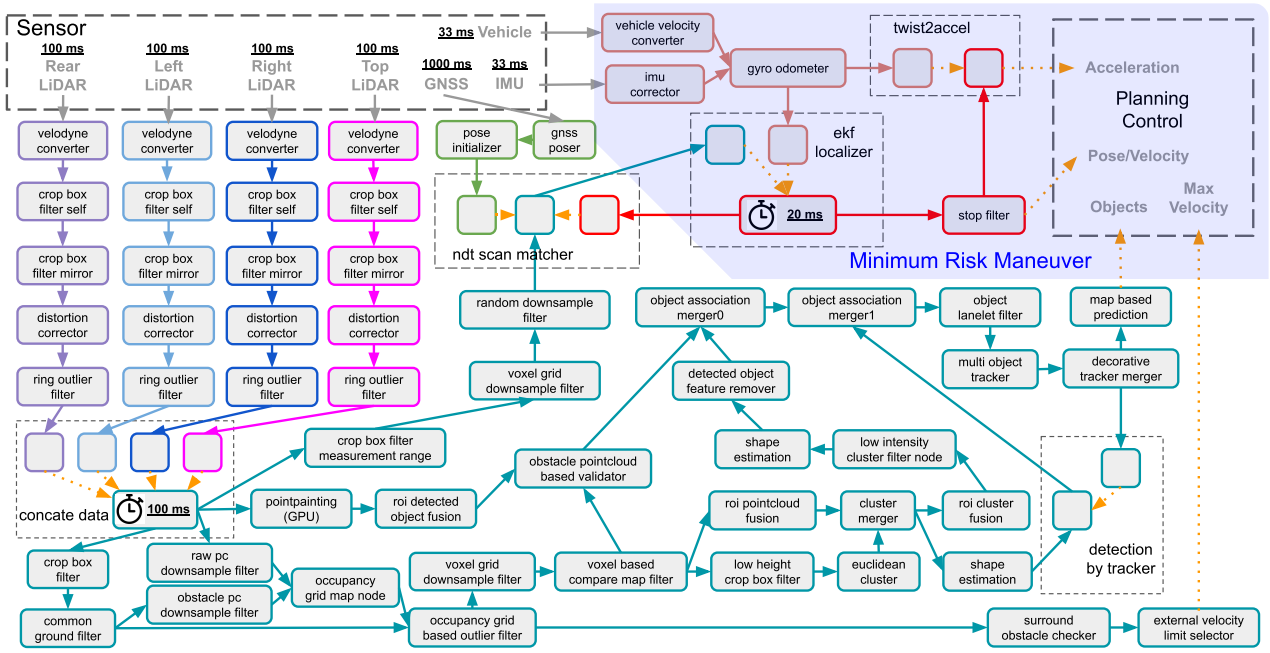
G1 Unit mismatch	→	**TP1** **The right unit of analysis** callback, node, chain, E2E cause-effect chain, or graph?
G2 Metric fragmentation	→	**TP2** **Joint analysis of heterogeneous timing metrics** one analysis over E2E latency, data freshness, timestamp disparity, miss probability
G3 Insufficient resource model	→	**TP3** **Heterogeneous resource models** GPU/NPU, memory bandwidth, interconnect as first-class
G4 Execution-time variability	→	**TP4** **Execution-time models under variability** scene-dependent workload + hardware-induced residual
G5 Lack of safety integration	→	**TP5** **Binding schedulability to safety** across the MRM mode change

TP1 · The right unit of analysis

from G1 · unit mismatch

> **Q.** What should be the unit of analysis: **callback, node, chain, E2E cause-effect chain, or graph**?

- Classical schedulability assumes a clear **unit task** that carries the deadline
- In ROS 2, the **schedulable** unit (the callback) and the **safety-relevant** unit (the cause-effect chain or graph) do not coincide [Choi+, RTAS'21] [Teper+, RTAS'25]



Running example (Autoware): the scheduler dispatches callbacks (boxes); safety constrains the 20 ms / 100 ms chains and the whole graph.

TP1 · What exists, and what it takes

from G1 · unit mismatch

What exists

A variety of models in use for AD systems: **chain**, **DAG**, **E2E cause-effect chain**, …

[Sun+, RTAS'23] [Yano & Azumi, RTSS'24 WiP]

Why not enough

No **shared agreed-upon** model: each work picks its own.

What it takes

Theorist-practitioner **dialogue**: express the implementation structures AD essentially requires, while balancing **fidelity to the real timing constraints** against **tractability of analysis and optimization**.

OPEN Consensus, aligned with practitioners, on one model that captures the essential AD implementation and balances constraint fidelity with analyzability.

TP2 · Joint analysis of heterogeneous metrics

from G2

> **Q.** How can **one analysis** reason jointly about all four of these?

E2E latency

sensor-to-actuator delay along a cause-effect chain

Data freshness

age of the data a consumer reads (max data age)

Timestamp disparity

timestamp gap between fused sensor streams (LiDAR vs camera)

Miss probability

exceedance probability of a deadline

- Each metric carries a **different safety meaning**, and one deployed AD system must satisfy **all of them at once**
- Collapsing them into one number loses exactly what safety needs

TP2 · Today: pairs · Goal: the full set

from G2

ANALYZED TOGETHER TODAY: A FEW PAIRS

timestamp disparity + E2E latency
[Li+, RTSS'22] [Sun+, RTAS'25]

MRT + MDA (max reaction time + max data age)
[Tang+, JSA'24] [Alcon+, TODAES'26]

ONE DIRECTION: PROBABILISTIC E2E LATENCY UNDER DATA FRESHNESS / TIMESTAMP DISPARITY CONSTRAINTS

$$\text{fresh} \le F,\ \text{disp} \le D \Rightarrow \Pr[\ \text{E2E} > L\ ] \le p_{\text{miss}}$$

Treat data freshness and timestamp disparity as **constraints**, and under them derive a **probabilistic** E2E latency.

OPEN An analytical model that treats the full metric set jointly.

TP3 · Heterogeneous resource models

from G3

Q. How can a timing model **explicitly capture GPU/NPU execution, memory bandwidth, and on-chip interconnect**, rather than leaving them unmodeled or abstracted away?

CPU	GPU / NPU	Memory bandwidth	Interconnect / NoC
modeled ✓ (classical response-time analysis)	DNN inference offloading: unmodeled?	contention between cores and accelerators: unmodeled?	on-chip communication: unmodeled?

- Accurate and optimal scheduling **requires modeling these resources explicitly**: ROS on NoC-based and clustered many-core (ROS-lite, ROS-lite2), heterogeneous SoC platforms [Azumi+, IROS'20] [Tajima+, IROS'24] [Chishiro+, ICESS'19]

OPEN An analysis of ROS 2 systems that makes these resources first-class.

TP4 · Execution time under variability

from G4

Q. How should execution time be modeled when it depends on **scene, data volume, and hardware states**?

- A single **worst case** is overly pessimistic, an **average** is unsafe: neither is actionable on its own

The variability has two sources deserving different treatment:

Scene-/data-dependent workload

Largely fixed once the scene is given: it scales with the number of detected objects or input points.

Hardware-/runtime-induced residual

Shared-resource contention on multicore and heterogeneous SoCs: caches, memory bandwidth, co-running work.

[Nowotsch+, ECRTS'14] [Reghenzani+, TECS'20]

Folding the scene-dependent part into a distribution **mischaracterizes it** — once the scene is given, it is not random: **separate the two**.

TP4 · Scene-explicit + probabilistic residual

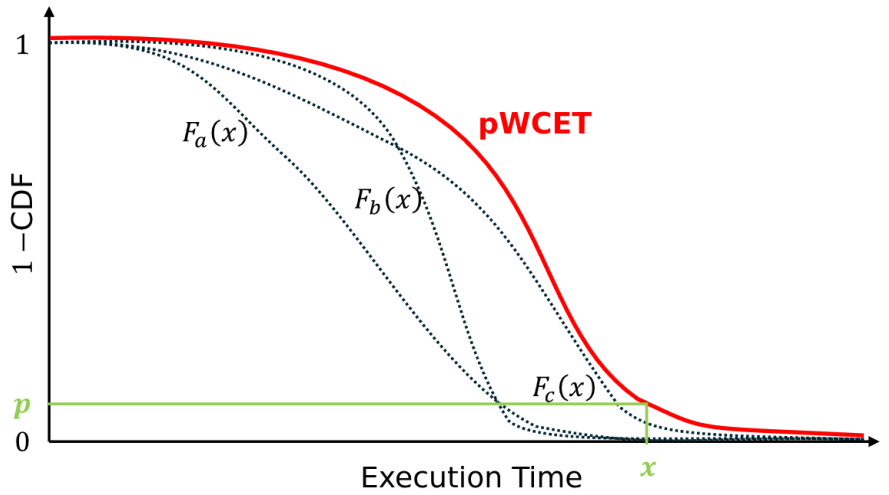
from G4

- Starting point: **pWCET** — treats every source of variability as a probabilistic event [Gil+, ESL'17] [Bozhko+, RTSS'23]
- Extends to probabilistic response time or E2E-latency analyses [Günzel+, TECS'23] [Han & Kim, RTAS'23]

What is missing: a model that treats the two sources of variability separately — one example of such a split:

$$\text{ExecTime}(\text{scene}) = g(\text{scene}) + \varepsilon$$

g: workload as an **explicit function of the scene** (objects, points) · **ε**: only the residual variation, modeled probabilistically



pWCET: an upper bound on the exceedance probability of the execution time.

OPEN A model that captures the scene-dependent workload explicitly, and only the residual variation probabilistically — and the analysis built on it.

TP5 · Binding schedulability to safety

from G5 · the MRM mode change

Q. What does a timing violation **mean** for safety, and what must the analysis guarantee **when a miss is about to happen**?

- A deadline miss or stale data acquires meaning only once tied to **fallback, degraded modes, and the MRM** (minimal-risk maneuver: an emergency stop based on dead reckoning)

STEP 1 · during execution

Detect the impending miss

at the optimal instant: too late leaves no room for a safe reaction, too early causes spurious mode changes

→

STEP 2 · on the violation

Trigger the mode change

hand over from nominal driving to the MRM

→

STEP 3 · after the change

Guarantee MRM schedulability

the fail-safe itself must meet its deadlines, at the highest criticality

The analysis should cover the **full fail-safe sequence**, not a single stage.

TP5 · Three stages, three separate existing works

from G5

Runtime miss detection

Early detection of an impending miss during execution → closest to **STEP 1**.

[Toba & Azumi, ECRTS'24] [Toba+, TECS'26]

Mixed-criticality scheduling

Criticality mode changes, guarantees for the highest level → closest to **STEP 2**.

[Burns & Davis, review 2022]

Schedulability analysis

Guaranteed timing correctness of a given mode → closest to **STEP 3**.

[Sun+, RTAS'23]

- Each line addresses one stage in isolation; the stages interact (the detection instant decides how much time the MRM has)

OPEN A model that co-derives all three: the detection instant, the mode-change trigger, and post-change MRM schedulability.

OPEN PROBLEMS · PART 2

Open Problems: For Practitioners

How to **engineer** a real system that is at once analyzable, performant, and safe

V

Five problems PP1 to PP5: duals and necessary conditions

How the practitioner problems pair with the theory

DUALS OF THEORY PROBLEMS — THE SAME QUESTIONS, ASKED FROM THE ENGINEERING SIDE

PP1 **Eliminating non-essential complexity** · middleware transparency

PP3 **Separating different criticalities** · resource isolation

PP4 **Detecting timing violations early** · online violation detection

CROSS-CUTTING NECESSARY CONDITIONS

PP2 **Enforcing the implementation-model contract**
analysis applies only if the implementation realizes the task model

PP5 **Implementing state-of-the-art schedulers**
analysis applies only if the platform enforces the assumed scheduler

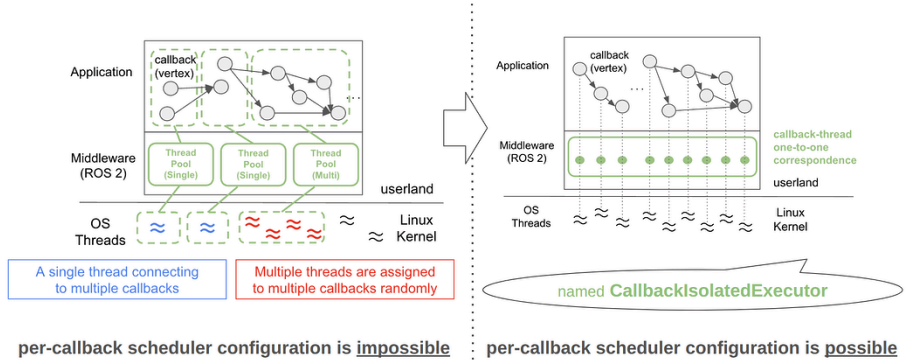
> **Progress on one side is useful only if matched on the other.**

PP1 · Eliminating non-essential complexity

theory dual

Q. How can we remove the **non-essential complexity** the middleware imposes on scheduling?

- **Nested scheduling** (executor atop the OS scheduler) is an implementation artifact: a persistent **1:1 callback-to-thread** mapping lets OS parameters apply per callback, and analysis can ignore the executor [Ishikawa-Aso+, RTAS'25 WiP]
- **DDS internal threads** are invisible to the model yet contend for the same cores: true zero-copy IPC (**Agnocast**) eliminates the dedicated DDS communication threads [Ishikawa-Aso & Kato, ISORC'25]
- **Residues remain**: a mutually exclusive `CallbackGroup` serializes outside the OS view · a non-reentrant callback overrun re-introduces middleware queueing · discovery, services, inter-host transport still occupy DDS threads



Per-callback scheduler configuration becomes possible with a 1:1 callback-thread correspondence.

OPEN Eliminate the residues, and characterize exactly **when the middleware layer can be ignored**.

PP2 · The implementation–model contract

necessary condition

> **Q.** How can the implementation **correctly realize** the theoretical task model?

TODAY: SEND TIMING IS UNCONSTRAINED

```
void callback(const In& in) {
  pub->publish(partial); // fires mid-execution
  heavy_work();          // successor already runs
}
```

Task models presume outputs become visible **on completion**; a mid-body `publish` breaks precedence constraints. The model holds only by programmer discipline.

DIRECTION: FUNCTION-AS-SUBTASK (FASS)

```
Out subtask(const In& a, const In& b) {
  Out out = compute(a, b);
  return out;            // send == completion
}
```

Arguments = incoming edges, return values = outgoing edges: sending is bound to subtask completion **at the API**, not by convention. [Ishikawa-Aso+, RTSS'25 WiP]

OPEN Extend to message synchronization, shared-variable, and blocking flows; deploy on production middleware.

PP3 · Separating different criticalities

theory dual

> **Q.** How can functions of different criticality **share one platform** without the lower disturbing the higher?

Highest	MRM: the safety-critical fail-safe
Intermediate	nominal AD functions: perception, planning, control
Lowest	logging, human-machine interface

- The tension: partitioning for assurance vs sharing for efficiency; full physical separation wastes the compute, power, and weight budget [Burns & Davis, 2022]

> **OPEN** How much CPU to allocate to **redundant functions idle in nominal driving** (e.g. the MRM)? Is **dynamically changing priorities / allocations** at fallback acceptable? — on commodity stacks.

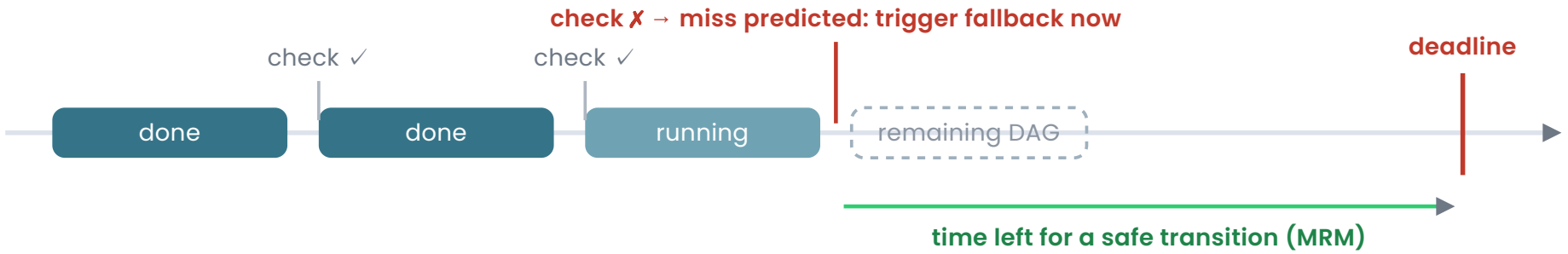
PP4 · Detecting timing violations early

theory dual

Q. How can we detect, **at runtime and early enough to act**, that a constraint will be missed?

- **Post-hoc detection is too late** for a safe transition: the detection instant must be right
- Pointers: DAG early detection with **variable / probabilistic thresholds**; runtime miss and data freshness tracking (**TILDE**)

[Toba & Azumi, ECRTS'24] [Toba+, TECS'26] [He+, DS-RT'23]



OPEN A low-overhead, chain- and graph-level online detector, wired to fallback, in production.

PP5 · Implementing state-of-the-art schedulers

necessary condition

Q. How can the latest scheduling algorithms actually be **enforced** on commodity stacks?

The obstacle: no system software treats a timing-constrained unit of execution (e.g. a DAG) as a **first-class citizen**: maintained natively by the runtime, precedence and deadline semantics enforced directly by the scheduler. Today the unit is reconstructed by convention atop threads and callbacks that have no notion of it.

- Consequence: published DAG schedulers cannot be enforced as specified, so they are evaluated largely by **analysis or simulation**
- Emerging: per-callback OS parameters · chain-aware priorities (PiCAS) · flexible OS-level scheduling (ROSRT) · reconciliation with classical periodic tasks [Ishikawa-Aso+, RTAS'25 WiP] [Choi+, RTAS'21] [Liu+, RTSS'25] [Teper+, RTAS'25]
- None of them makes the timing-constrained unit first-class

OPEN A repeatable path from published algorithm to enforced, low-overhead implementation on production stacks.

Takeaway

Closing the theory-reality gap is a two-way convergence.

TP **Theoretical models must evolve toward AD reality: unit, metrics, resources, variability, safety.**

PP **Implementations must be reshaped to realize analyzable models: transparency, contracts, isolation, detection, enforcement.**

Progress on one side is useful only if matched on the other.

All ten problems on one map

GAPS	FOR THEORISTS → THE MODEL	FOR PRACTITIONERS → THE SYSTEM
G1 Unit mismatch	TP1 The unit of analysis	PP1 Non-essential complexity
G2 Metric fragmentation	TP2 Joint timing metrics	PP2 Implementation-model contract
G3 Insufficient resource model	TP3 Heterogeneous resources	PP3 Criticality separation
G4 Execution-time variability	TP4 Variability-aware timing	PP4 Early violation detection
G5 Lack of safety integration	TP5 Schedulability and safety	PP5 Enforceable schedulers

Takuya Azumi (Saitama University) · Atsushi Yano (Saitama University / TIER IV) · RTSOPS 2026

Two asks before we finish

These ten problems are open — as far as we know

1 Prove us wrong

If any of these problems are already solved, please let us know.

2 Join us

If you want to tackle any of them, let's collaborate.

Thank you — let's close these gaps together.