VOLVO

Time-Predictable Communication in Service-Oriented Architecture - What are the challenges?

Hoai Hoang Bengtsson
Volvo Cars Sweden

Nicolas Navet
Uni. Luxembourg & Cognifyer

2023-03-21

Agenda

- Determinism vs Predictability
- Zone-Based Architectures
- Timing Accurate Models
- Takeaways on V&V
- A look forward

Challenges in the design of timing predictability architectures

Deterministic vs. Real-time System

Deterministic System

- is a system which, ***given the same set of inputs and initial state, will always produce the same outputs***.
- In the context of in-vehicle network, deterministic communication is the capability of the network to deliver a message at a specified time, not faster or slower.

Hill Descent Control

Real-time system

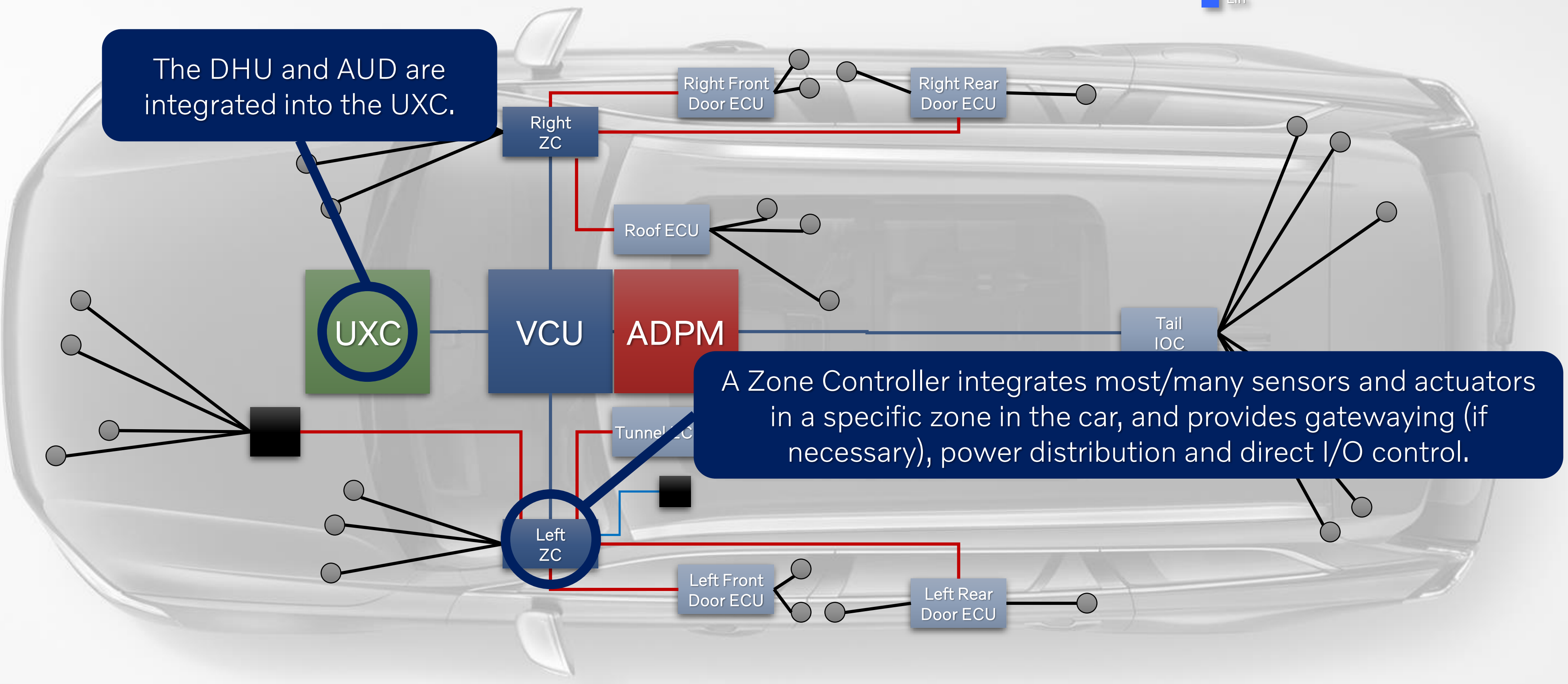
- Is a system in which correctness depends not only on the output value but also *the* ***time at which results are produced***
- In real-time communication, network has capability of deliver a message within a period, referred to as **bounded delay**

Vehicle is a complex real-time system!

Zonal Architecture

Integrating VIUs, power distribution and mechatronic ECUs into zone controllers

Ethernet
CAN/CAN FD
Lin

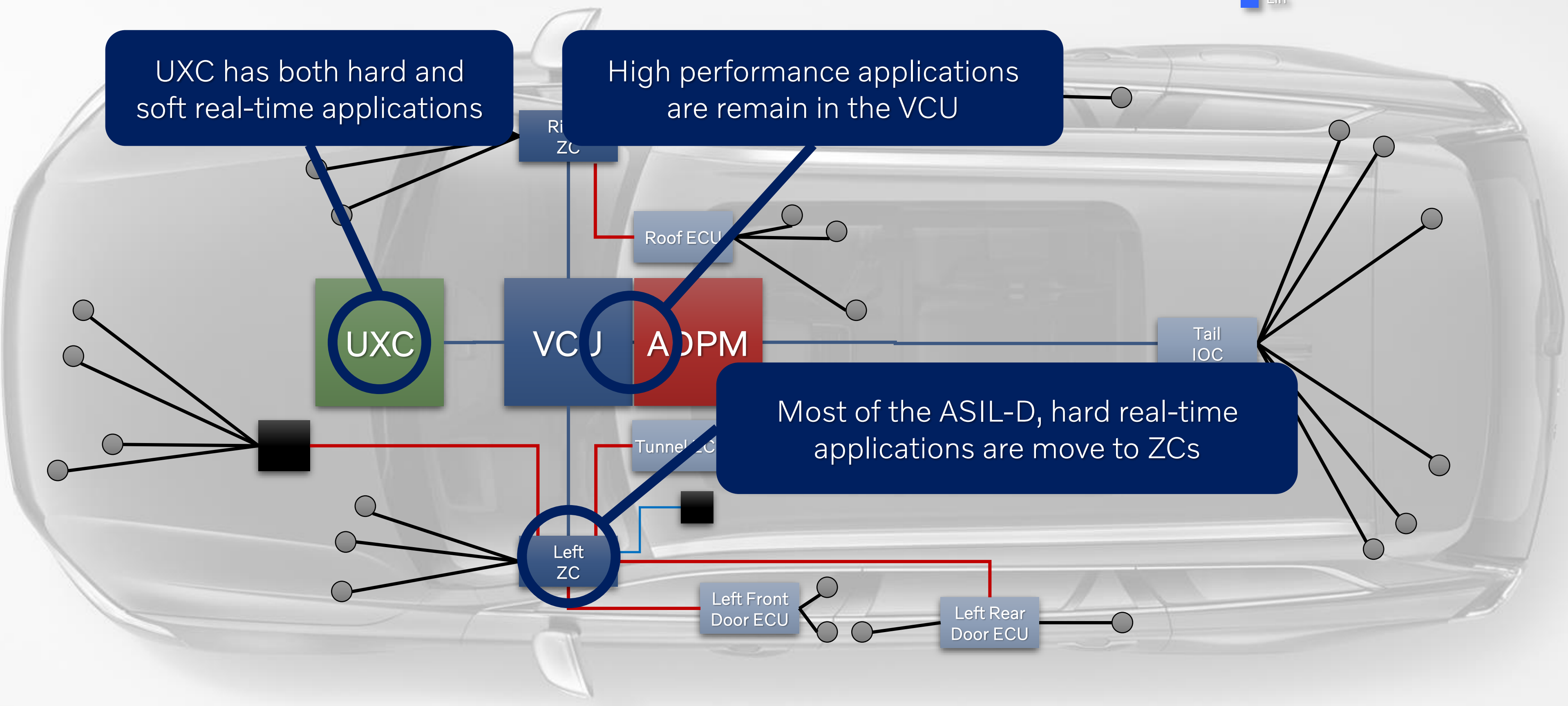


The DHU and AUD are integrated into the UXC.

A Zone Controller integrates most/many sensors and actuators in a specific zone in the car, and provides gatewaying (if necessary), power distribution and direct I/O control.

VOLVO

From complete vehicle perspective, we want to guarantee bounded delay for hard real-time application



Example use Case: brake system

ZCL processes data and send to front actuators

ZCR processes data and send to rear actuators

Control function

Ethernet frames

Control function

CAN frames

Actua tor

Brake ECU

Electric Machine

Brake ECU send out CAN frame periodically every 10ms

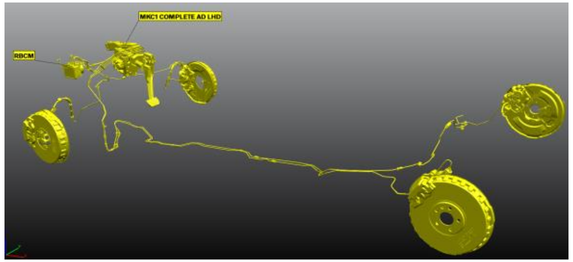
Sens or

Actua tor

ZC gatewaying CAN to ETH

Data from wheel speed sensor pooling every 1ms

End to end requirements is 15ms



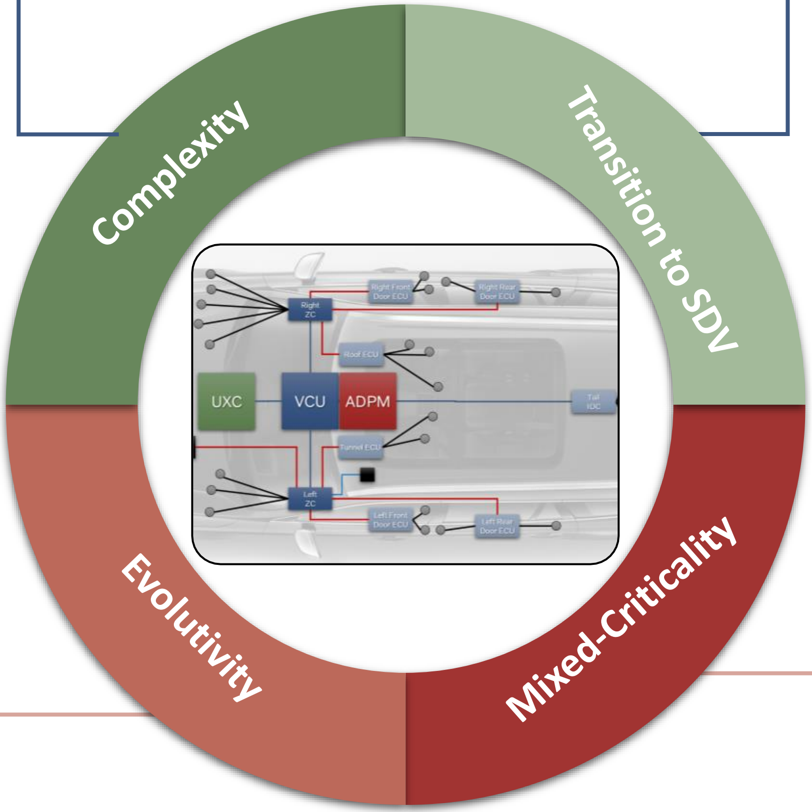
Challenges in the design of timing predictable architectures

Intrinsic complexity of systems and technologies

- # of functions, signals, services, flows
- Technology selection & configuration
- Mixed legacy / next-gen: e.g., signals to services
- Multi-tier dev. process, product lines, ...

New business models based on SW

- Now and in the future, How to future-proof an E/E architecture ?



Besides TSN ?

- Predictability of complex execution platforms (e.g., Autosar Adaptive) and complex SoCs ?
- Which SOA? SOME IP, DDS, Iceoryx, Ecal?

Mixed criticality with TSN

- Network engineering
- Fail-operational requirements
- Verification & Validation

Timing predictability requires controlling interfering activities

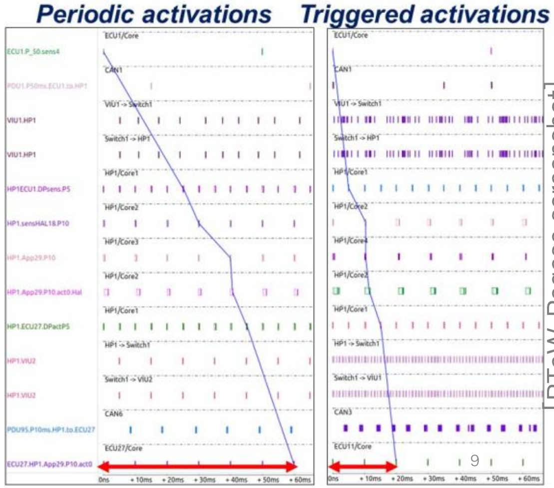
on a single resource, e.g. an ETH link

across segments of a timing chain

- ✓ Interferences from higher / lower / same-priority traffic
- ✓ Interferences from one segment of a timing chain to another: triggered transmissions, timeouts, ..

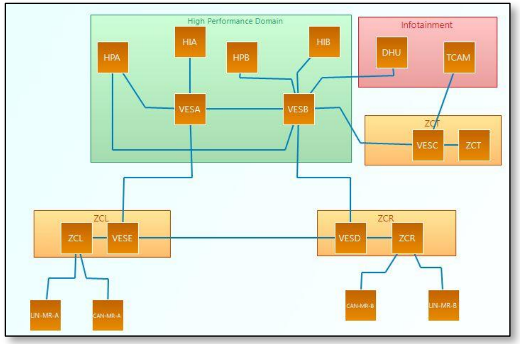
Solution: temporal isolation, total or partial partitioning in the time domain

- ✓ Apply TSN QoS mechanisms
  - Priorities, preemption or TAS to protect from lower-priority traffic
  - TAS to contain uncontrolled traffic in dedicated time slots
  - CBS to limit the interference of a medium-priority class with bursty streams
  - ...
- ✓ Periodic transmissions & executions .. at the expenses of latencies
- ✓ Jitter-aware triggering mechanisms possible



[RTaW-Pegase screenshot]

Guaranteeing timing predictability requires timing accurate models



[RTaW-Pegase screenshot]

The need for timing accurate models of the system

A System must be timing predictable, and we need timing-accurate models of it too

B ≠ kinds of models for ≠ perf. metrics

- ✓ Models of the worst-case or typical-case behavior
- ✓ Latencies, jitters, throughput, memory usage, reliability
- ✓ Formalisms: Network-Calculus, sched. analysis, discrete event simulation

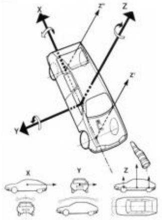
C Models must be a) accurate enough and b) require reasonable computational efforts

a) E.g., no accurate models of the worst-case behavior of TCP streams
b) Simulation models may not be computationally efficient for rare events

Combining timing verification techniques along the dev. process

Simulation

Typical Case Behavior



- ✓ Functional simulation
- ✓ **Timing-accurate simulation of ECU, networks, system level w/wo fault-injection**
- ✓ Model / SW / CPU / HW in-the-loop

Formal Verification

Worst Case Behavior & Rare Events

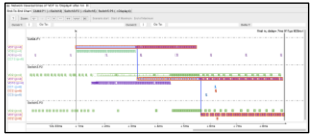
- ✓ Worst-Case Execution Time analysis
- ✓ **Worst-Case Response Time analysis: ECU, buses, system level**

$$K_i^k(t) \stackrel{\text{def}}{=} \underbrace{\left\lfloor \frac{J_i^k + \varphi_i^k(\phi^i)}{T_i^k} \right\rfloor}_{\substack{\text{max. number of instances} \\ \text{that may accumulate at } t_c}} + \underbrace{\left\lfloor \frac{t - \varphi_i^k(\phi^i)}{T_i^k} \right\rfloor + 1}_{\substack{\text{max. number of instances} \\ \text{in } [t_c, t_c + t)}} \quad (7)$$

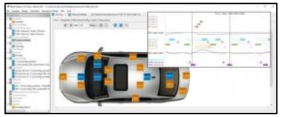
- ✓ Reliability analysis

Testing

Measurements



- ✓ Execution time measurements
- ✓ **Off-line trace analysis**
- ✓ Runtime monitoring
- ✓ Integration tests



"Early stage"

Technological & Architectural choices

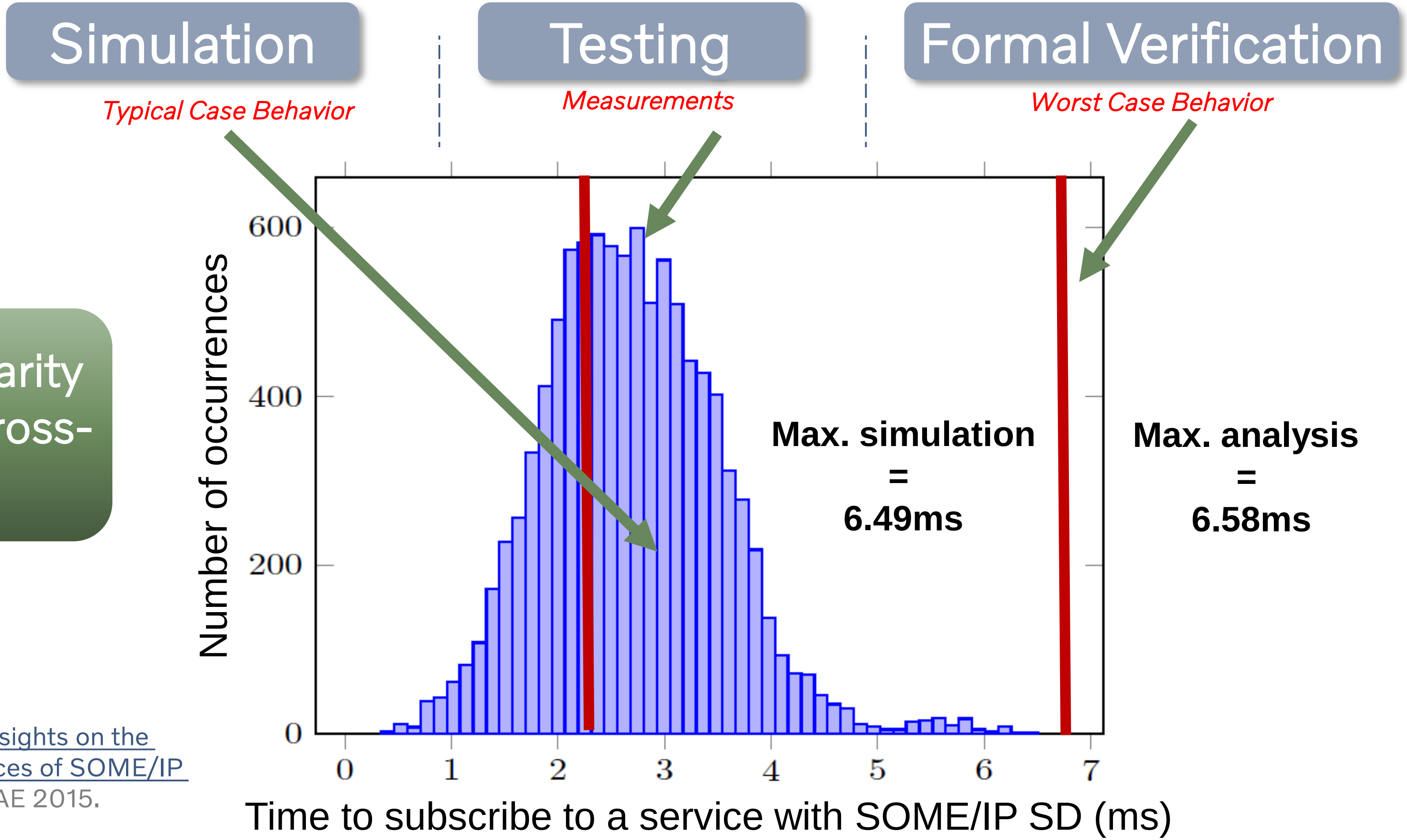
"Project"

Configuration & Model-Based Verification

"Real"

Refine and validate models & impact of non-conformance

Combining timing verification techniques



Complementarity of results & cross-validation

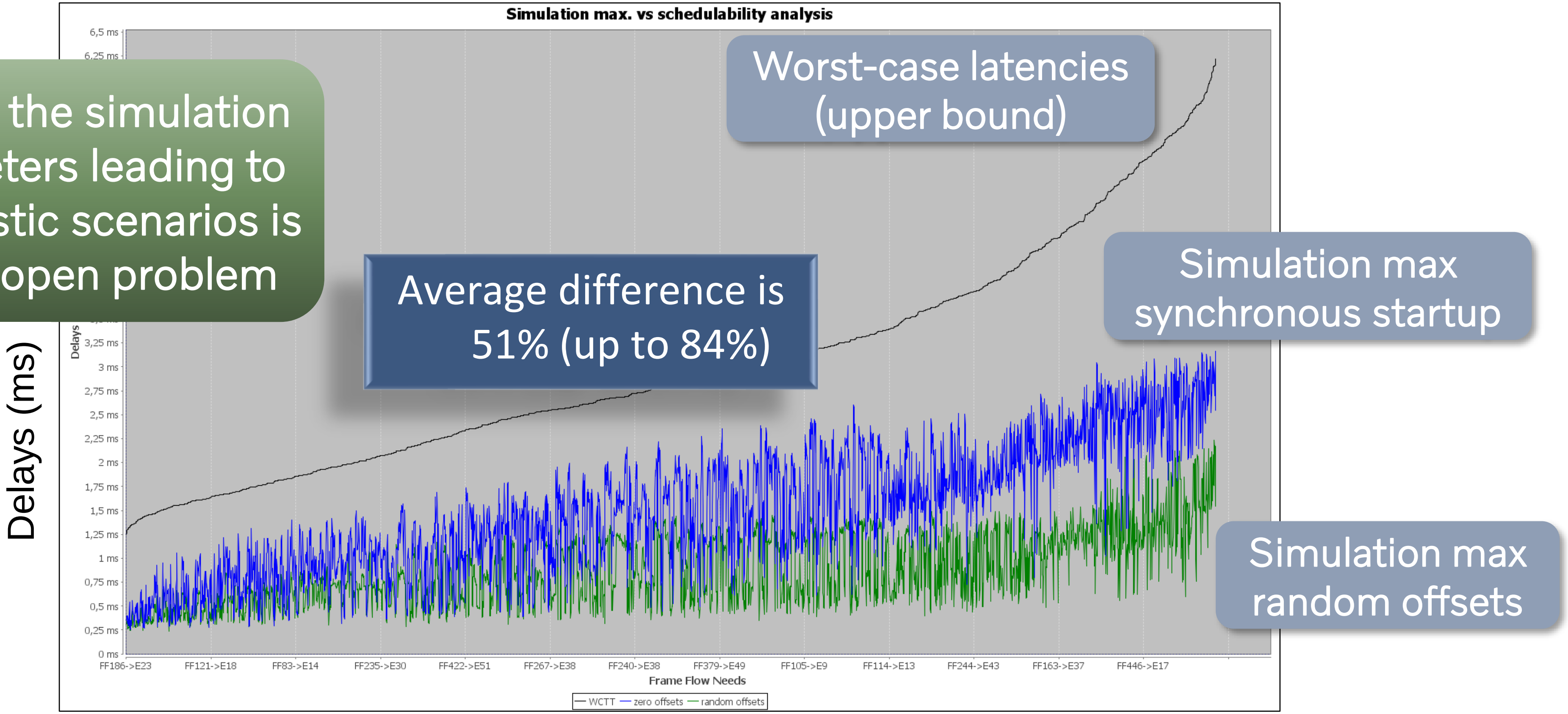
Fig. from N. Navet et al "Insights on the Configuration and Performances of SOME/IP Service Discovery", SAE 2015.

Takeaways from timing verification in automotive

VOLVO

Observation #1: Worst-case temporal scenario is out of reach of simulation, schedulability analysis is needed

Finding the simulation parameters leading to pessimistic scenarios is still an open problem



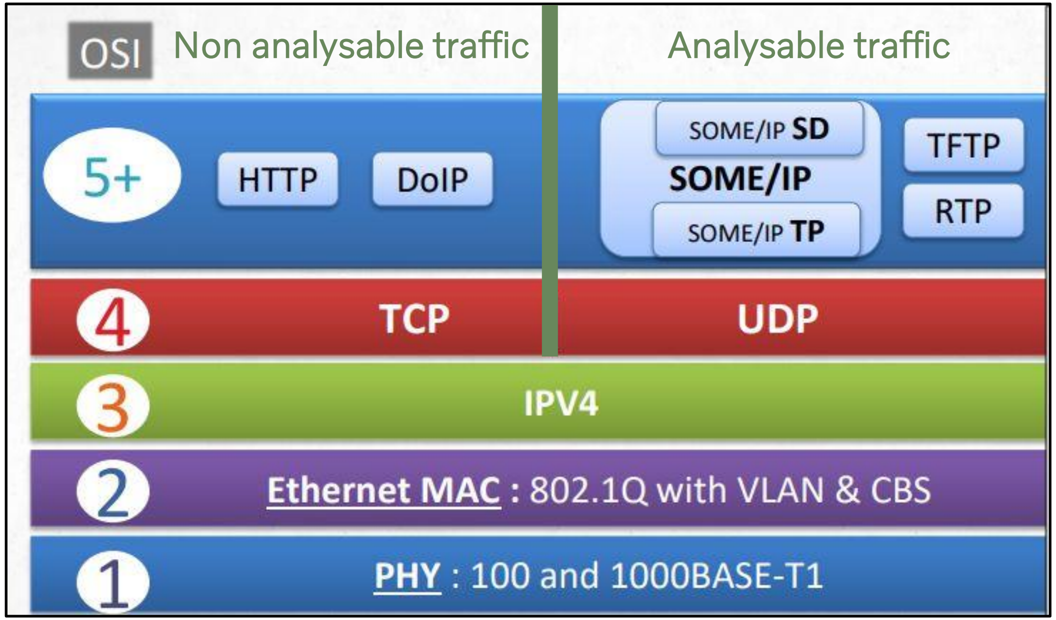
Flows sorted by increasing worst-case latencies

Fig. from "Timing verification of real-time automotive Ethernet networks: what can we expect from simulation?"

Observation #2: worst-case timing analysis will not cover all in-vehicle communication technologies, e.g. TCP

Solution: build "temporal firewalls" to control interference of non-analysable traffic

1. Isolate non-analysable traffic at lowest priority, in dedicated TAS slots or CBS classes & verify it with simulation
2. Worst-case schedulability analysis can be applied on the rest of the traffic

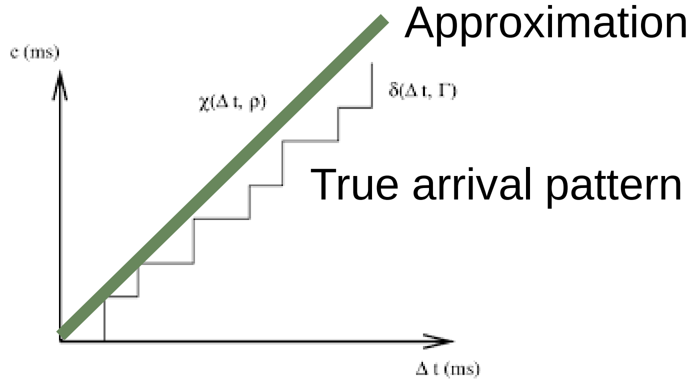


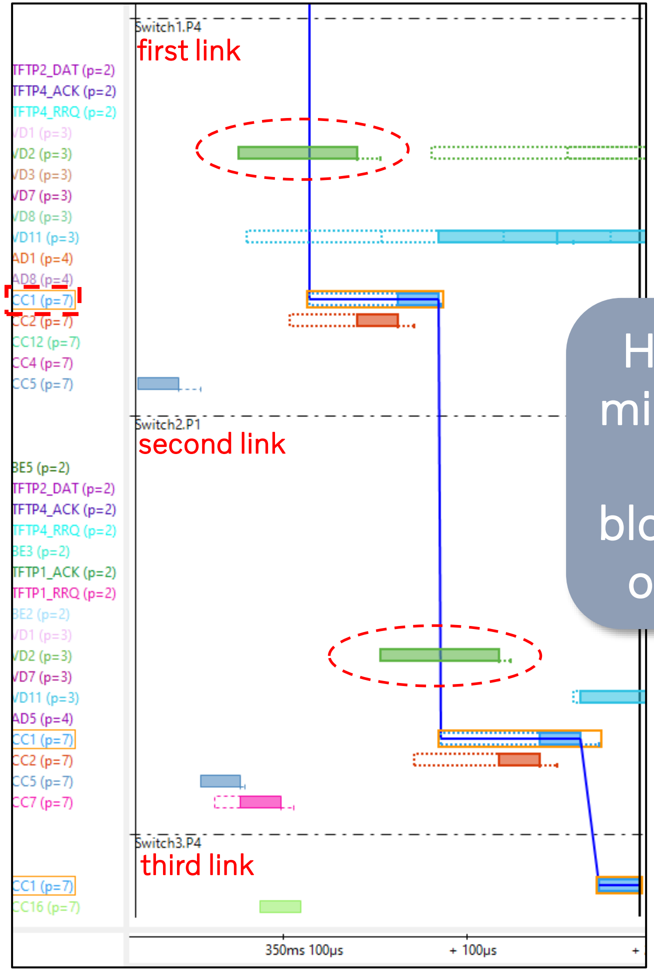
Simplified example automotive communication stack

Simulation, analysis and a "verification-aware" configuration strategy are needed – could the same strategy be applicable for complex execution platforms on SoCs ?

Observation #3: Worst-case analysis gives limited insight about what goes wrong .. analysing worst simulated scenario is helpful

To reduce computing time, worst-case analyses on ETH make approximations corresponding to no actual scenario





[RTaW-Pegase screenshot]

Replaying the worst scenario observed during long simulations

High-priority frame CC1 misses its deadline due to lower priority frames blocking it → preemption or TAS would help here

Observation #4: models (and their parameters) can be either accurate or approximate, and will fulfil ≠ use-cases

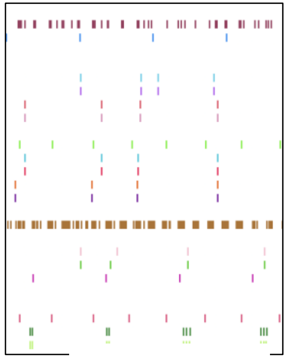
1. <u>Fine-grained vs coarse-grained models</u>: HW, protocols, MW, OS services, Apps, …
2. <u>Precise vs approximate model parameters</u>: traffic patterns, knowledge of all or part of the traffic, …

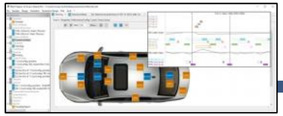
<u>Approximate models</u>

- ✓ Not for verification!
- ✓ Support early-stage design choices
- ✓ Parameter settings corresponding to ≠ scenarios can be considered

<u>Accurate models and parameters</u>

- ✓ Suited for verification
- ✓ But conservative load assumptions not usable in automotive!
- ✓ Possible solution: refine parameters with trace analysis





“Early stage” → **“Project”** → **“Real”**

Configuration & Verification

Refine models & impact of non-conformance

Conclusion

Zone-based architecture explored by Volvo & challenges to timing predictability

Timing predictability through temporal firewalls between transmissions subject to ≠ requirements → TSN offers solutions with priorities, CBS, TAS and preemption

Complexity of next-generation execution platforms is a threat to timing predictability → clear V&V strategy throughout the dev. process

Design decisions should be made with the understanding of the timing guarantees that that can be obtained and how

Thank you for your attention!