Department of Informatics
Technical University of Munich

Reproducible Layer 3-Enabled TSN Experiments

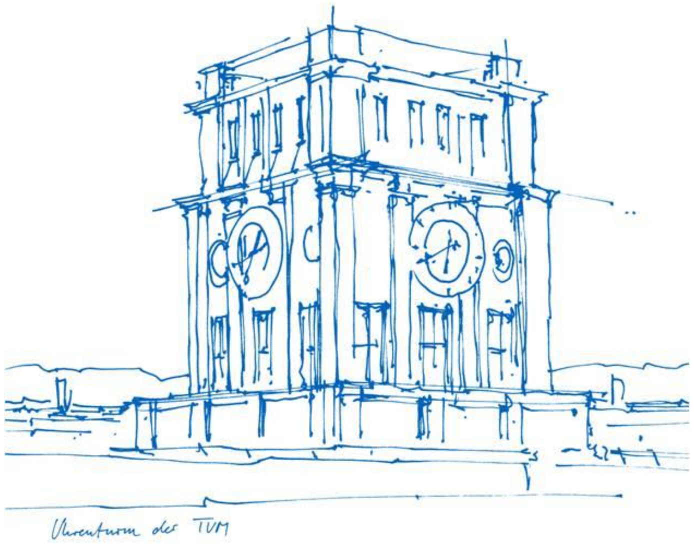
Authors:

Filip Rezabek, Marcin Bosk, Thomas Paul, Kilian Holzinger, Sebastian Gallenmüller, Angela Gonzalez, Abdoul Kane, Francesc Fons, Zhang Haigang, Georg Carle, Jörg Ott

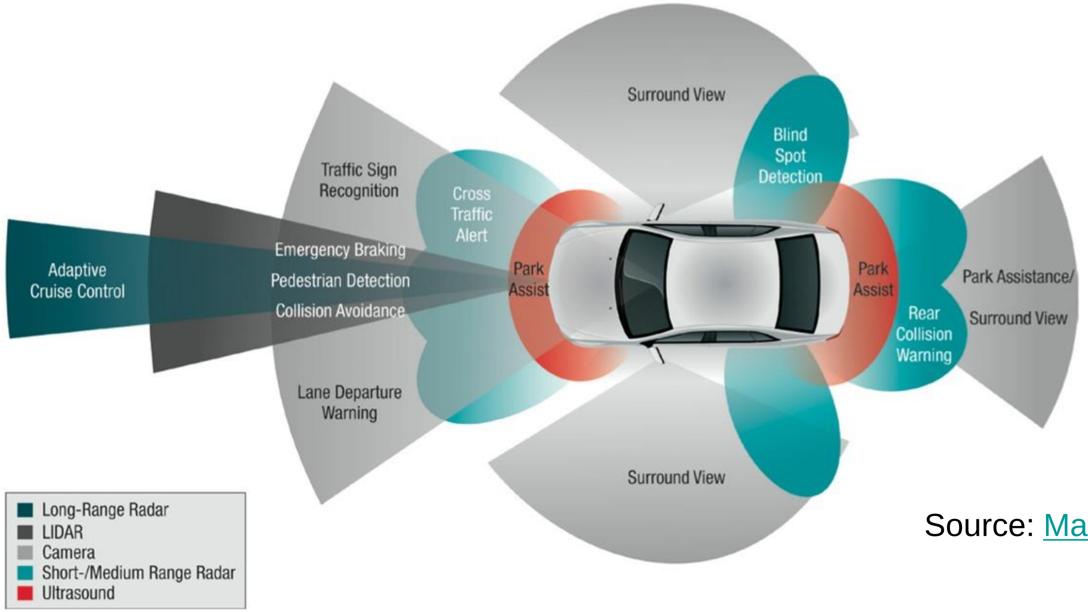
Presenters:

Filip Rezabek, rezabek@net.in.tum.de

Marcin Bosk, bosk@in.tum.de



Introduction
Motivation



Source: Machine Design

→ Usage of Time-sensitive networking (TSN)

Structured approach to assessing the capabilities of IVNs with TSN

- **Early** during the design
- In a **reproducible** manner
- Compare different **architectures** and their implications

→ ***EnGINE*** is a framework for flexible, scalable, and replicable TSN experiments

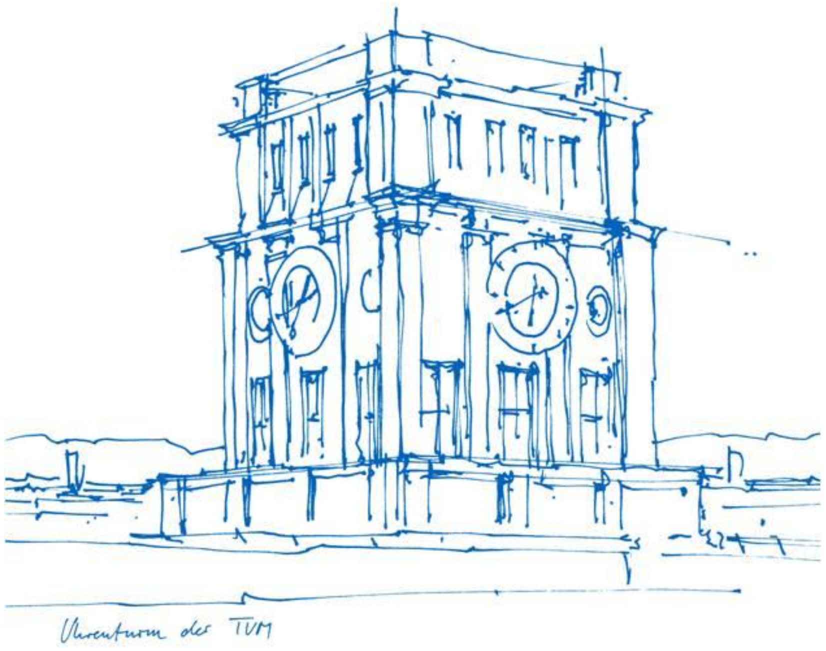
EnGINE – **En**vironment for **G**eneric **I**n-Vehicual **N**etwork **E**xperiments

Focus on TSN
Supported TSN standards

Within the scope of IVNs focus on:
- Bounded latency
- Low packet delay variation
- Low packet loss

IEEE 802.1Qbv – Time Aware Priority (TAPRIO) shaper
IEEE 802.1Qav – Credit-Based Shaper (CBS) algorithm
IEEE 802.1AS – general Precision Time Protocol (gPTP)
Launch time feature – Earliest Time First (ETF)

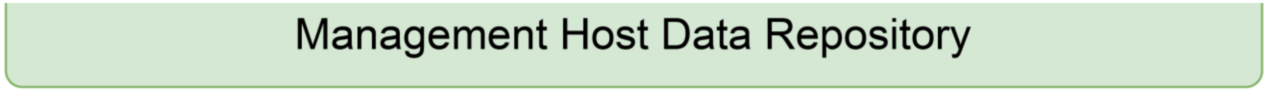
→ Part of IEEE 802.1DG automotive profile standard



DESIGN OF ENGINE

EnGINE Design
Overview

Orchestrated from the management host



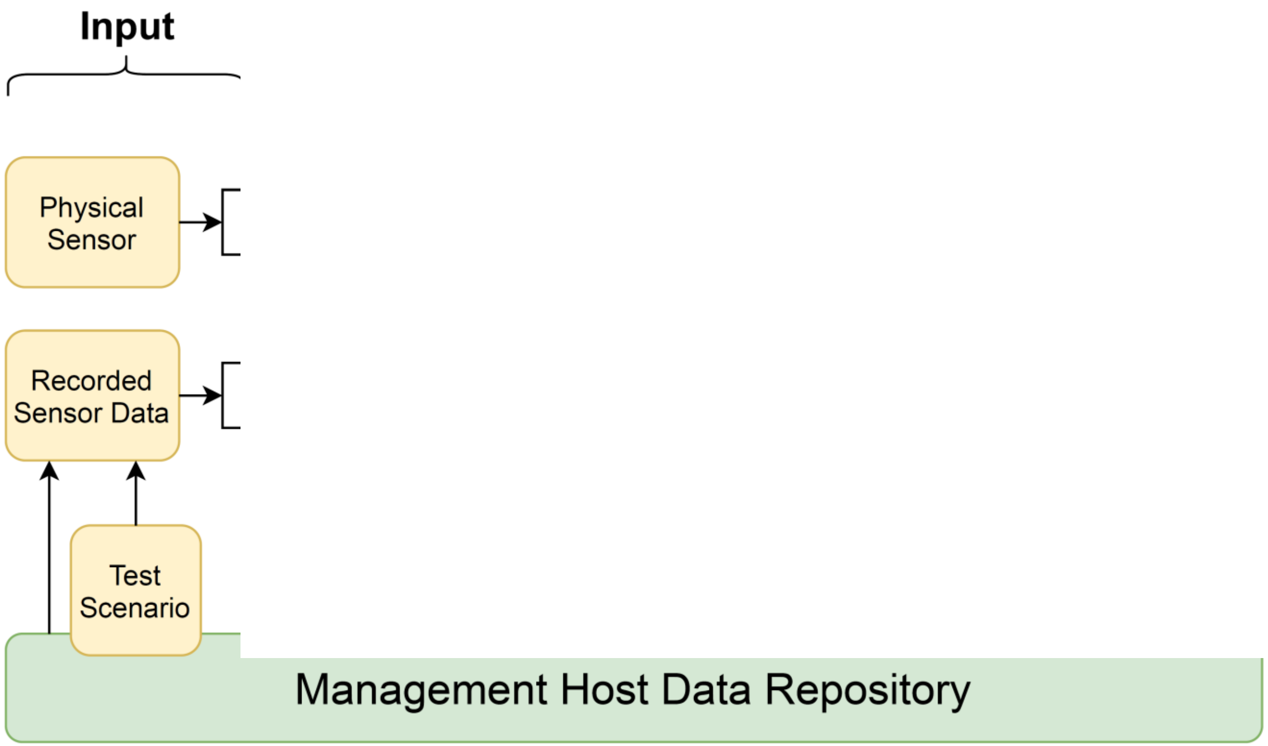
EnGINE Design
Overview

Orchestrated from the management host

Three parts of each experiment

Input

- Defines the experiment
- Specifies data sources and network



EnGINE Design
Overview

Orchestrated from the management host

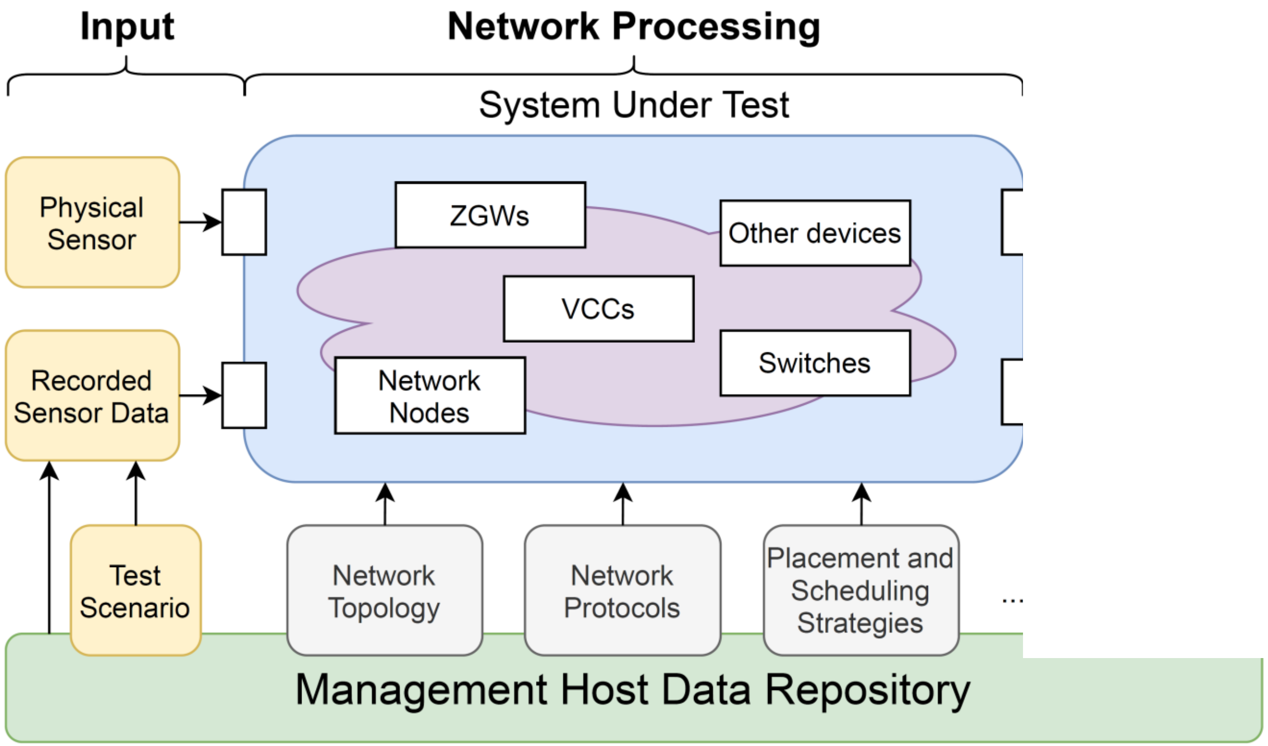
Three parts of each experiment

Input

- Defines the experiment
- Specifies data sources and network

Network Processing

- Encompasses the tested system
- Takes configuration from input
- Supports the experiment
- Uses sensors



ZGWs – Zonal gateways
VCCs – Vehicle control computers

EnGINE Design
Overview

Orchestrated from the management host

Three parts of each experiment

Input

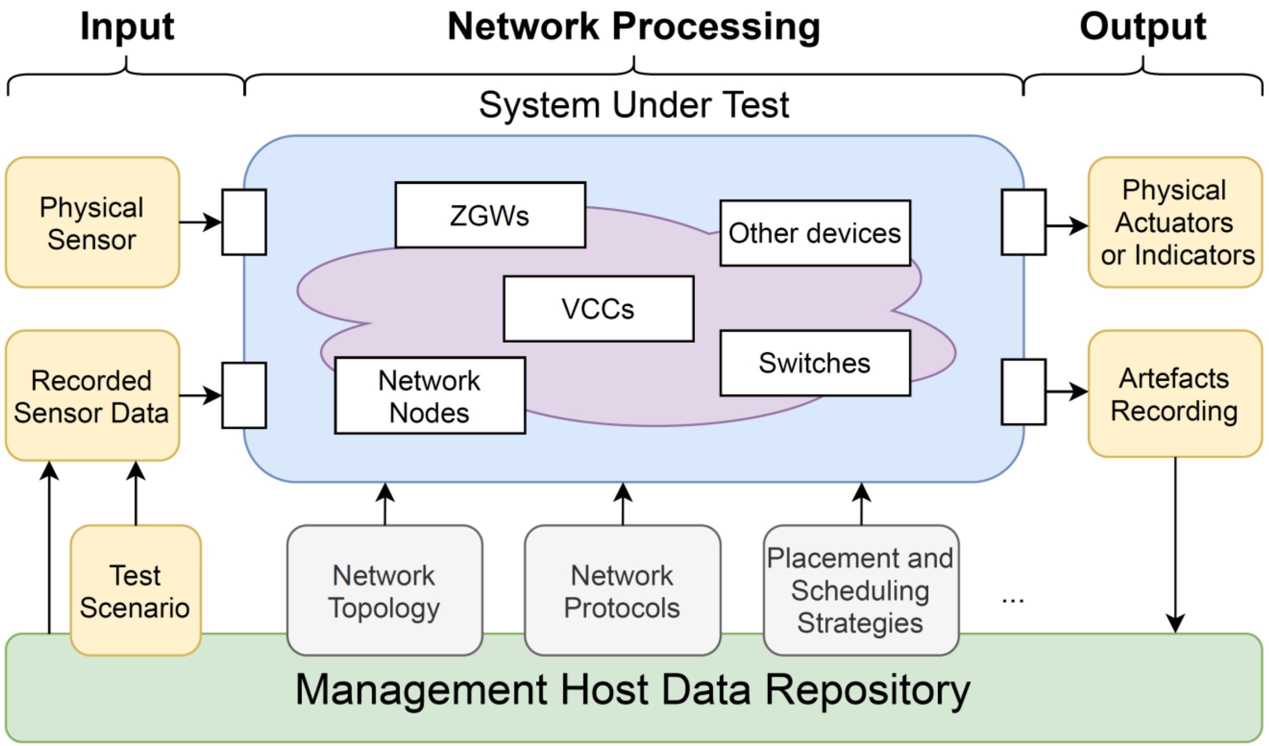
- Defines the experiment
- Specifies data sources and network

Network Processing

- Encompasses the tested system
- Takes configuration from input
- Supports the experiment
- Uses sensors

Output

- Records experiment results
- Can include physical actuation
- Can be shown on monitors



ZGWs – Zonal gateways
VCCs – Vehicle control computers

EnGINE Design
Overview – HW Description

15 Nodes

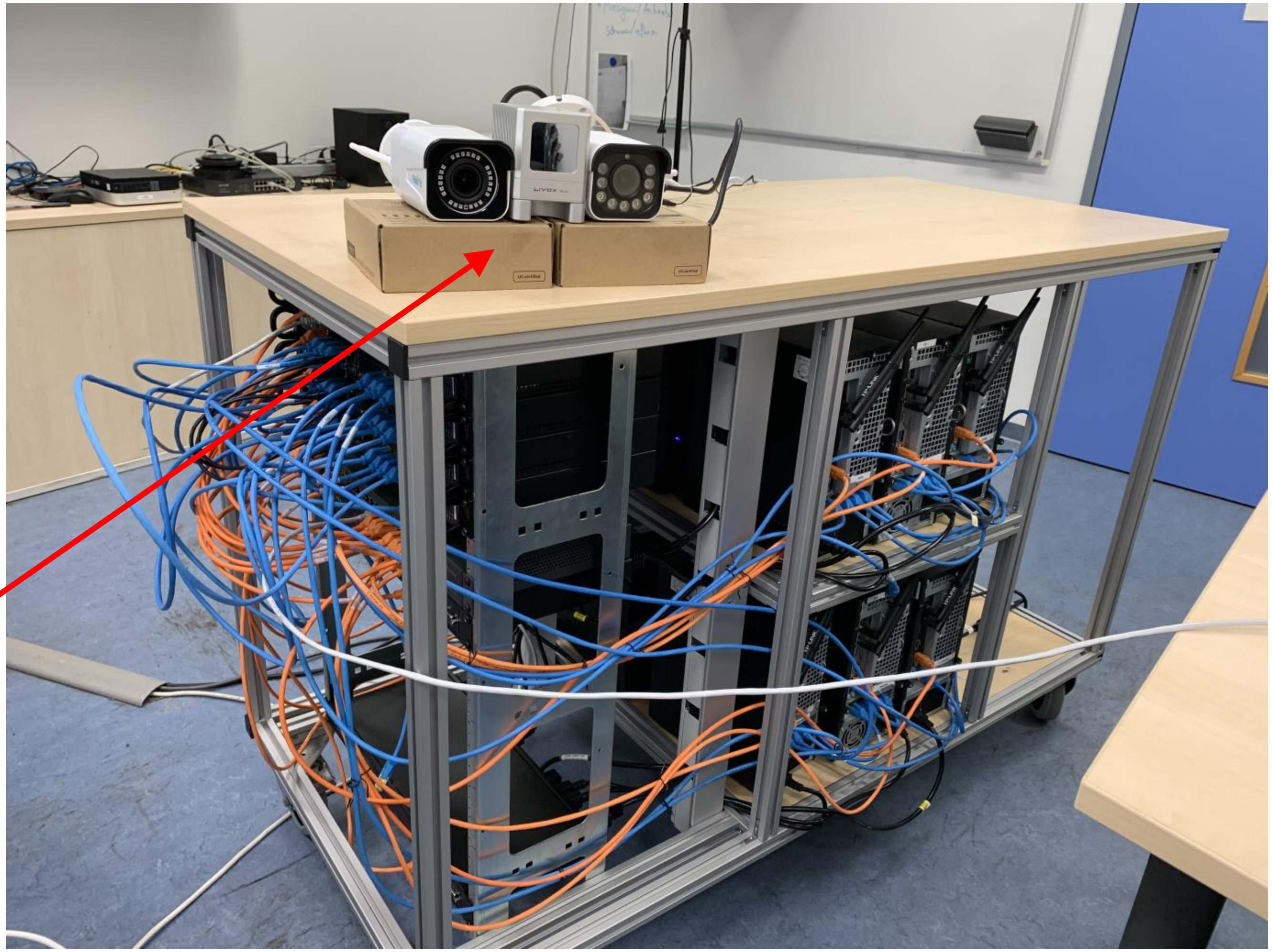
- 12 PCs – ZGWs
- 3 Servers – VCCs

Supported Network Interface Cards (NICs)

NIC Type	NIC Speed	Supported IEEE 802.1 Standards
Intel i210	1Gbit/s	AS, Qav, Qbv
Intel i225	2.5Gbit/s	AS, Qav, Qbv
Intel i350	1Gbit/s	AS
Intel x552	10Gbit/s	AS

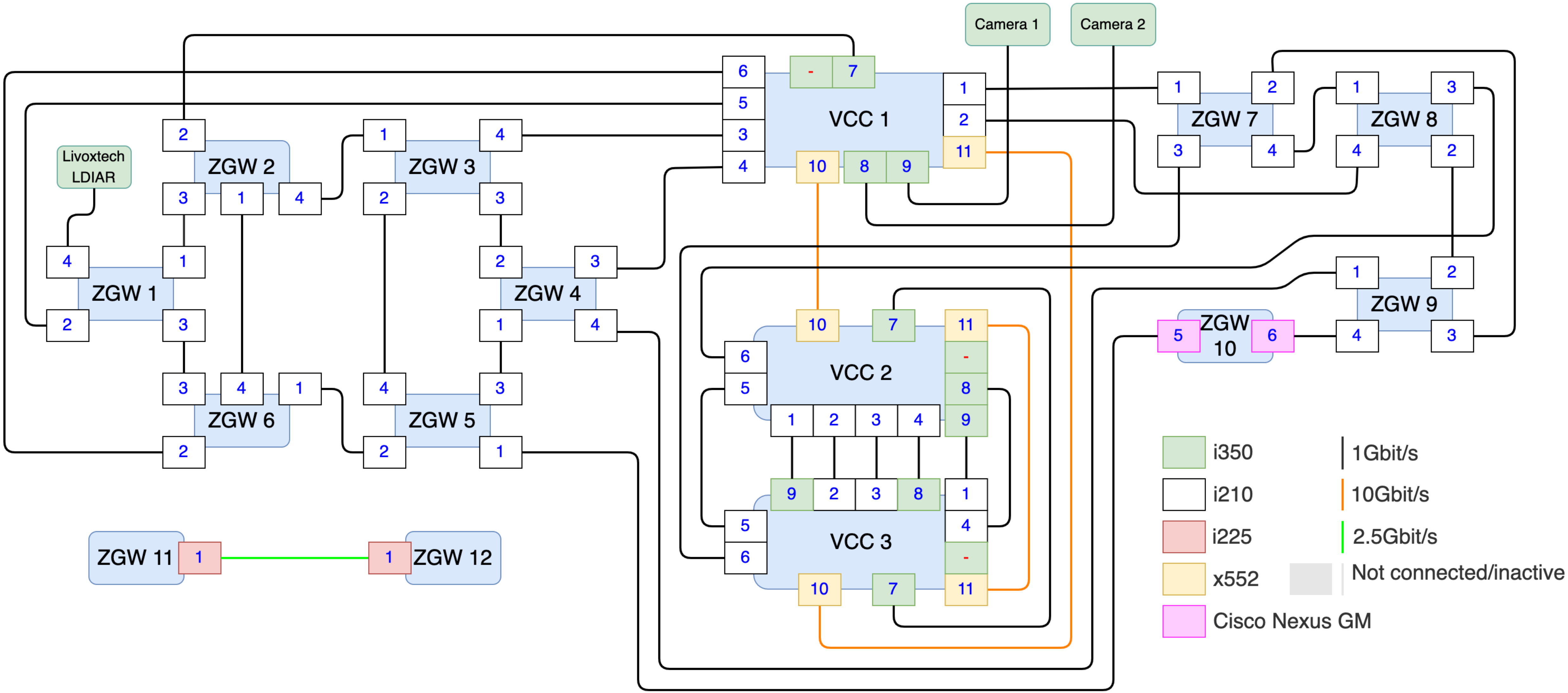
Supported Sensors

- LIDAR Livoxtech Mid 40
- Cameras Reolink Full HD



EnGINE Design

Overview – Physical Deployment

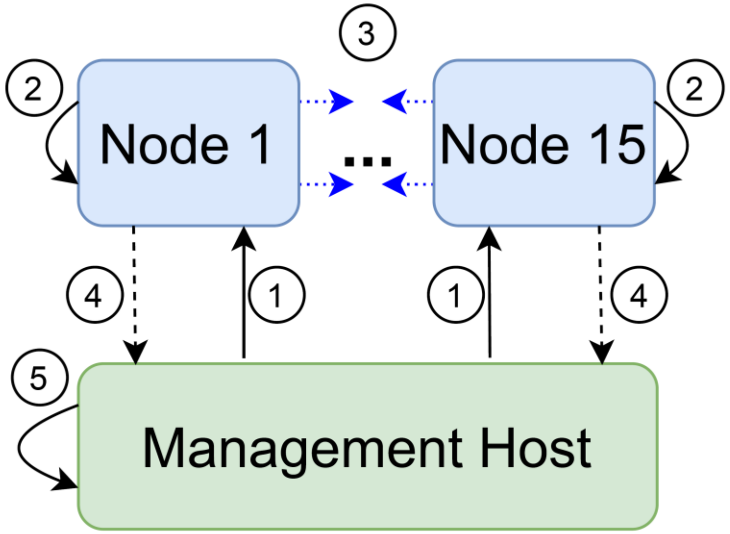


EnGINE Design

Configuration and Management

Written in Ansible

- Automatic experiments set execution
- Orchestration from management host via SSH



1. Management host communicates with nodes
2. Nodes execute the tasks
3. Interact with other nodes
4. Store the collected artifacts
5. Process artifacts

EnGINE Design

Configuration and Management

Experiments within campaign independent of each other

- Defined by an input dataset
- Evaluated output for each individual experiment

Four phases of experiment **campaigns**

1. Install → **2. Setup** → **3. Scenario** → **4. Process**

1. Install: Install OS image on testbed nodes e.g. RT kernel
2. Setup: Install dependencies
3. Scenario: Run experiments within scenario
4. Process: Post-process scenario artefacts

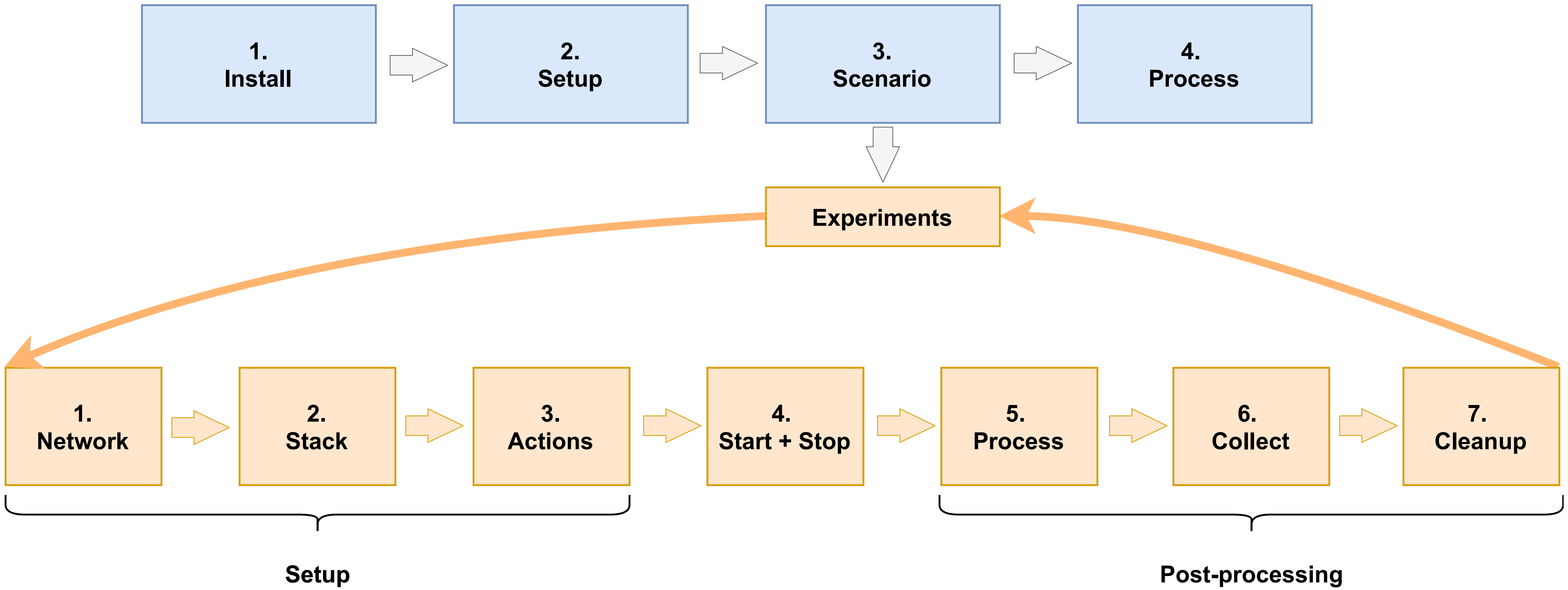
EnGINE Design

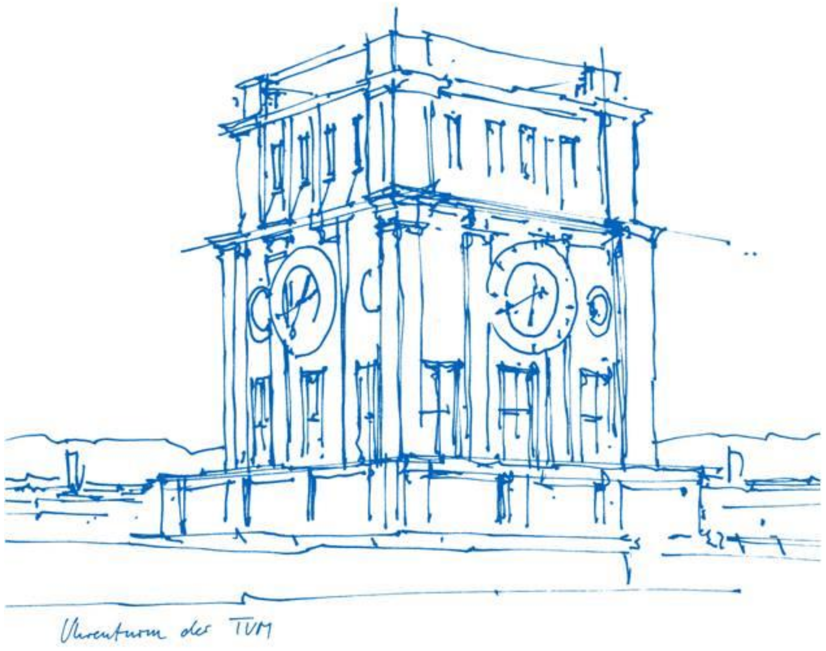
Configuration and Management – Scenario Definition

A use-case or specific topic; can be divided into multiple experiments
Example: LIDAR with a multi-hop path and VCC as a sink

Contains individual experiments, executed in a loop

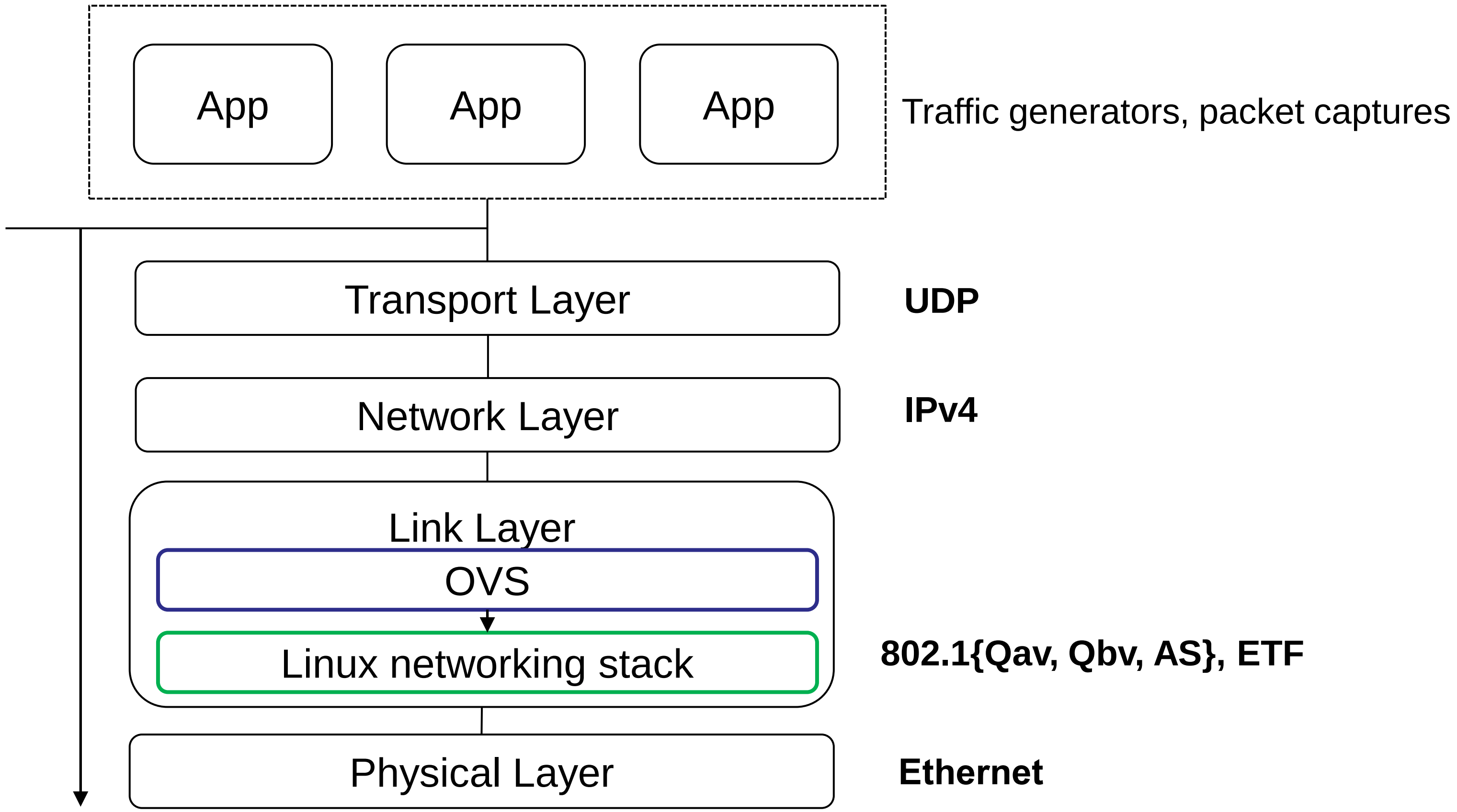
Each experiment = 7 steps





CAPABILITIES

EnGINE capabilities
Supported TSN Standards



EnGINE Capabilities

Defining a Scenario – Sample Use-case

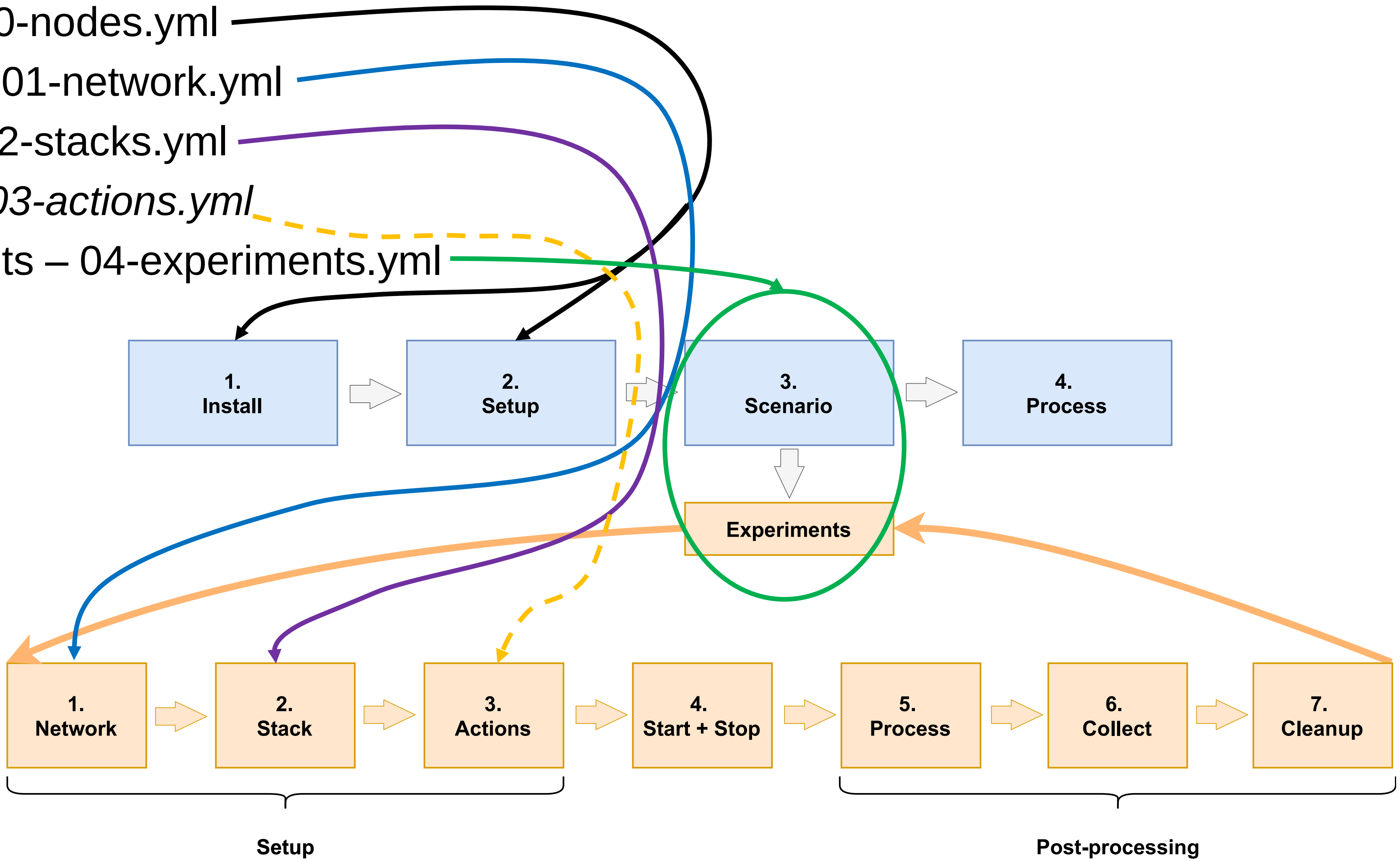
Nodes – 00-nodes.yml

Network – 01-network.yml

Stacks – 02-stacks.yml

Actions – 03-actions.yml

Experiments – 04-experiments.yml



EnGINE Capabilities
Post-processing

Processing of results happens in two phases

First – on a node e.g., ZGW-5

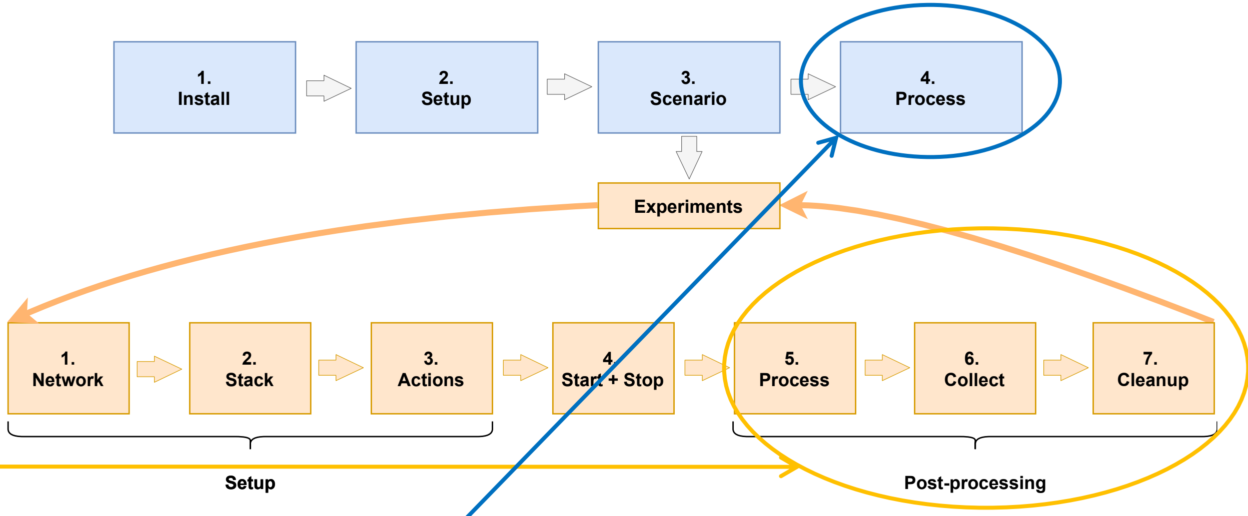
- Happens immediately, e.g. .pcap data to csv
- Then collect and cleanup on the node

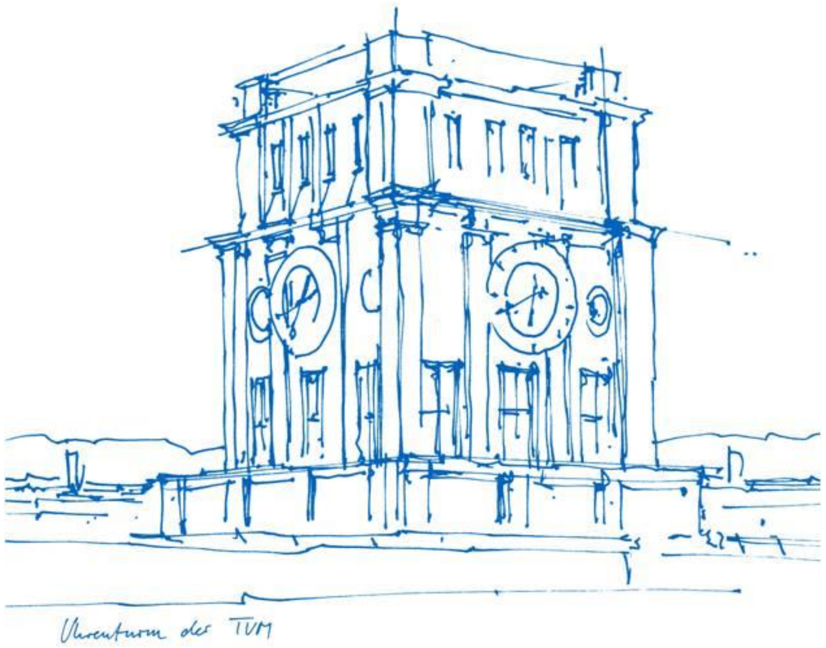
Second – on a management host, after all experiments finished

Option to do additional evaluation on:

→ experiment base

→ among various experiments





SYSTEM OPTIMIZATION

System Optimization

Enable Support for SR Class A and B Requirements

Verification in a simple exemplary scenario

- **Credit-Based Shaper** (CBS) on the interfaces
- Interested in latency and jitter!
- Focus on IVNs → SR class A by Avnu Alliance

Traffic generation using Iperf3 – usually:

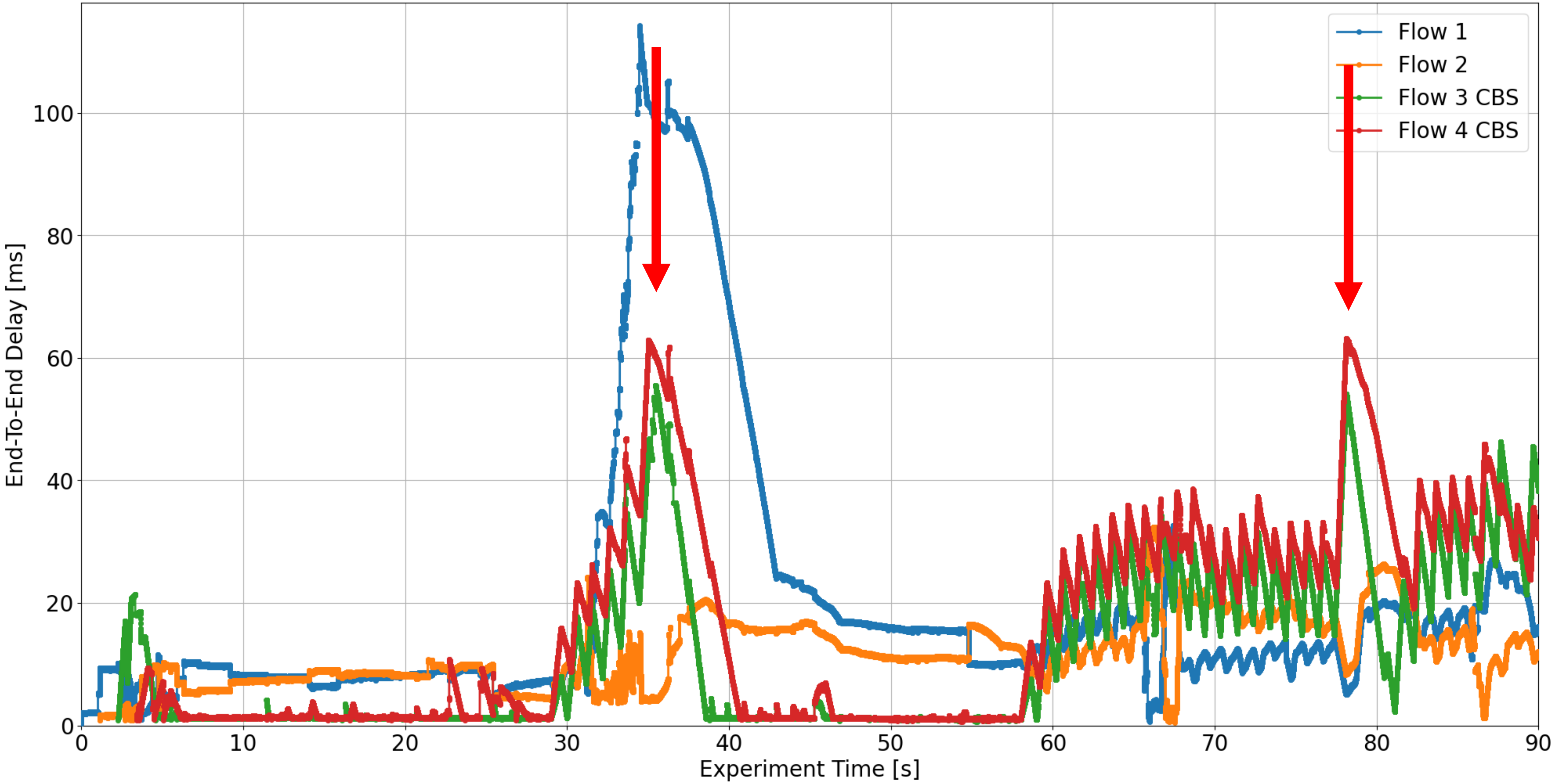
- Two best-effort flows
- Two policed flows
- Policed flows need to fulfil SR class A (and B) requirements

Class	Max Latency over 7 hops	CMI	Max Jitter
A	2 ms	125 us	125 us
B	50 ms	250 us	1000 us

For policed flows – CBS shaper and Iperf3 configured for 100 mbit/s throughput

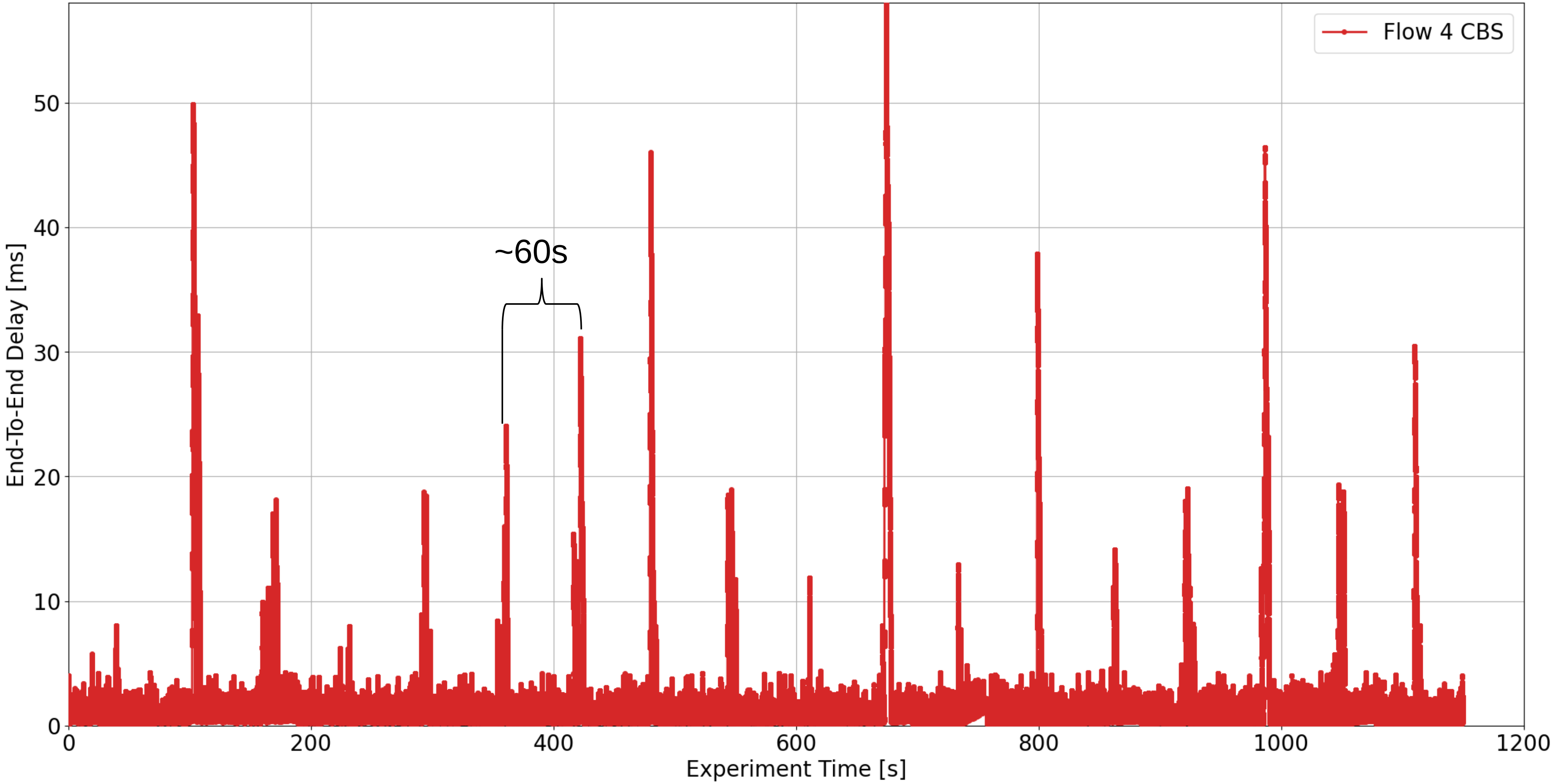
System Optimization

Step 1: CBS and Iperf3 Configuration Verification



Outcome: Configuration seems correct, but other artefacts present

System Optimization
Step 2: Artefact Verification



Outcome: Periodic “spikes” in End-To-End Delay observed

System Optimization
Step 2b: Verify Linux Behavior

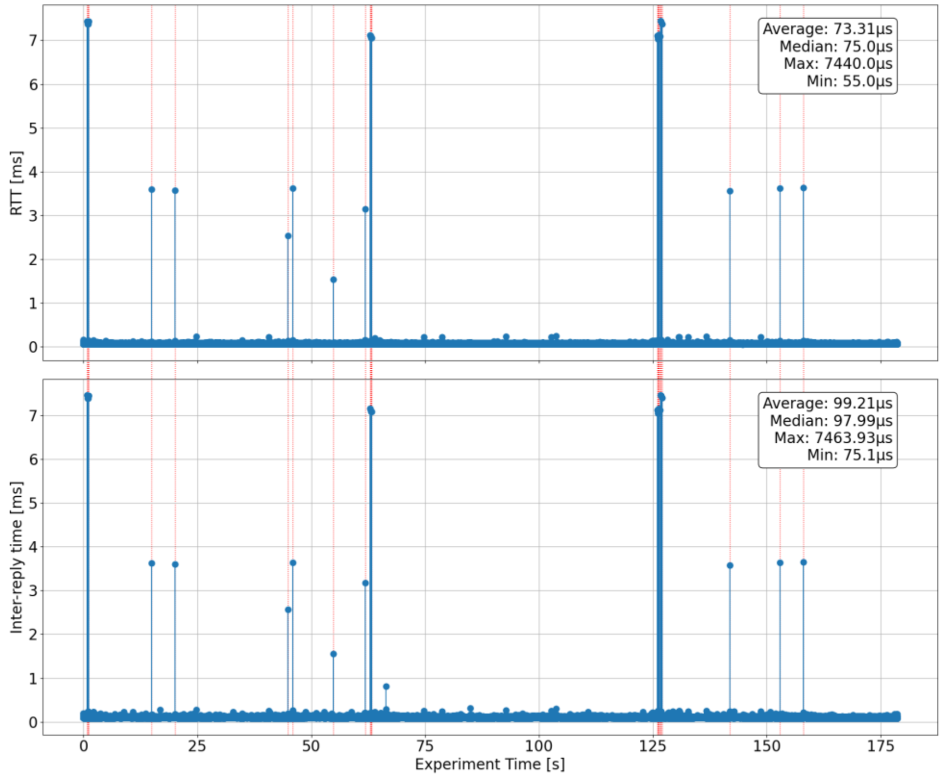
Even a simple ping shows spikes roughly every 60s!

All points to a periodic Linux function, but we can't identify it…

Solution: Use CPU isolation and CPU affinity
- Isolation → Dedicated logical cores to relevant functions
- Affinity → Assign a task/process/IRQ to a certain logical core

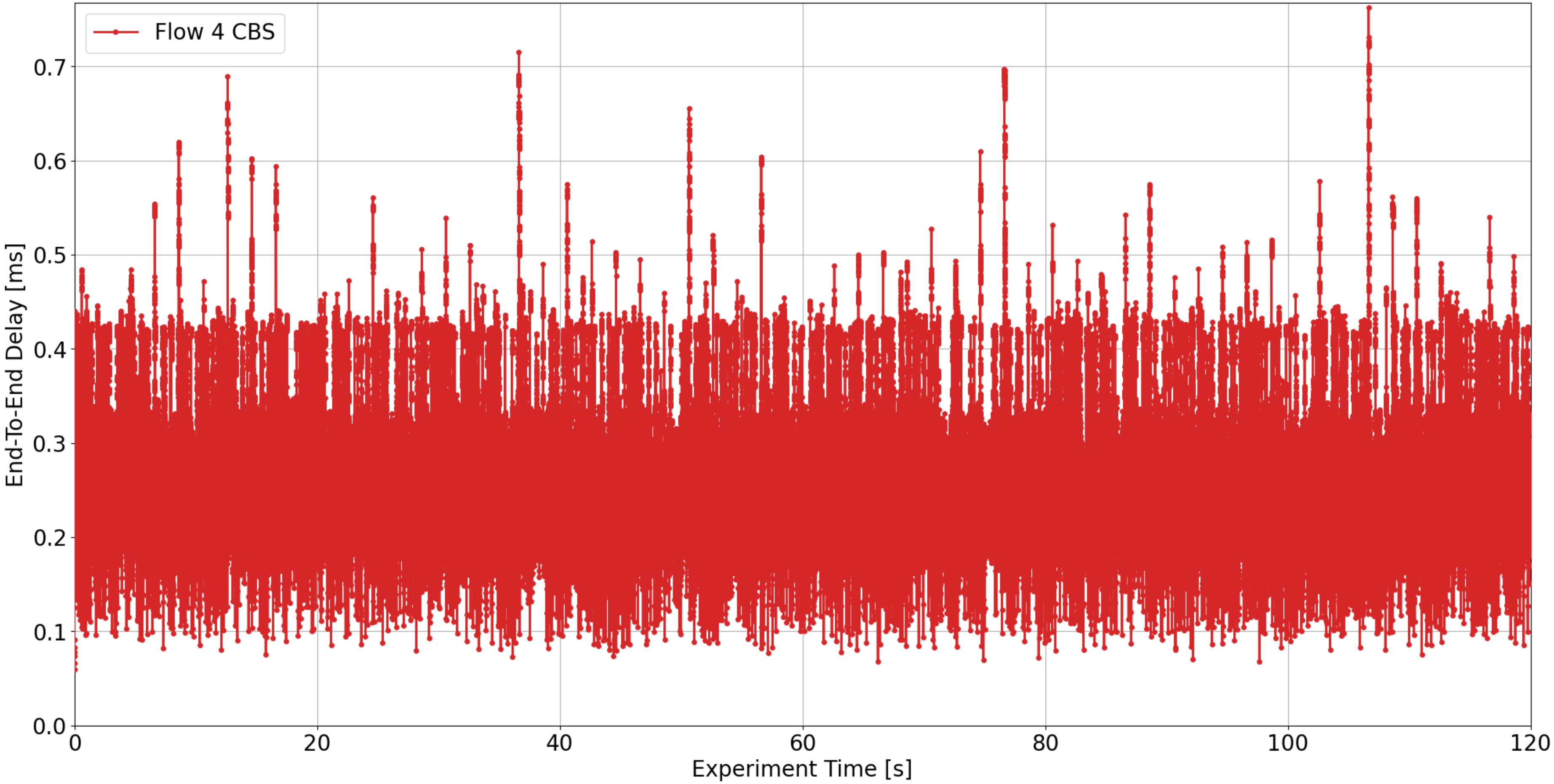
→ Isolate all experiment-relevant functions from the rest of the system!

This also applies to Network Interface Card interrupts
- We dedicate a few (usually two) cores for those
- Requires a low-latency kernel



System Optimization

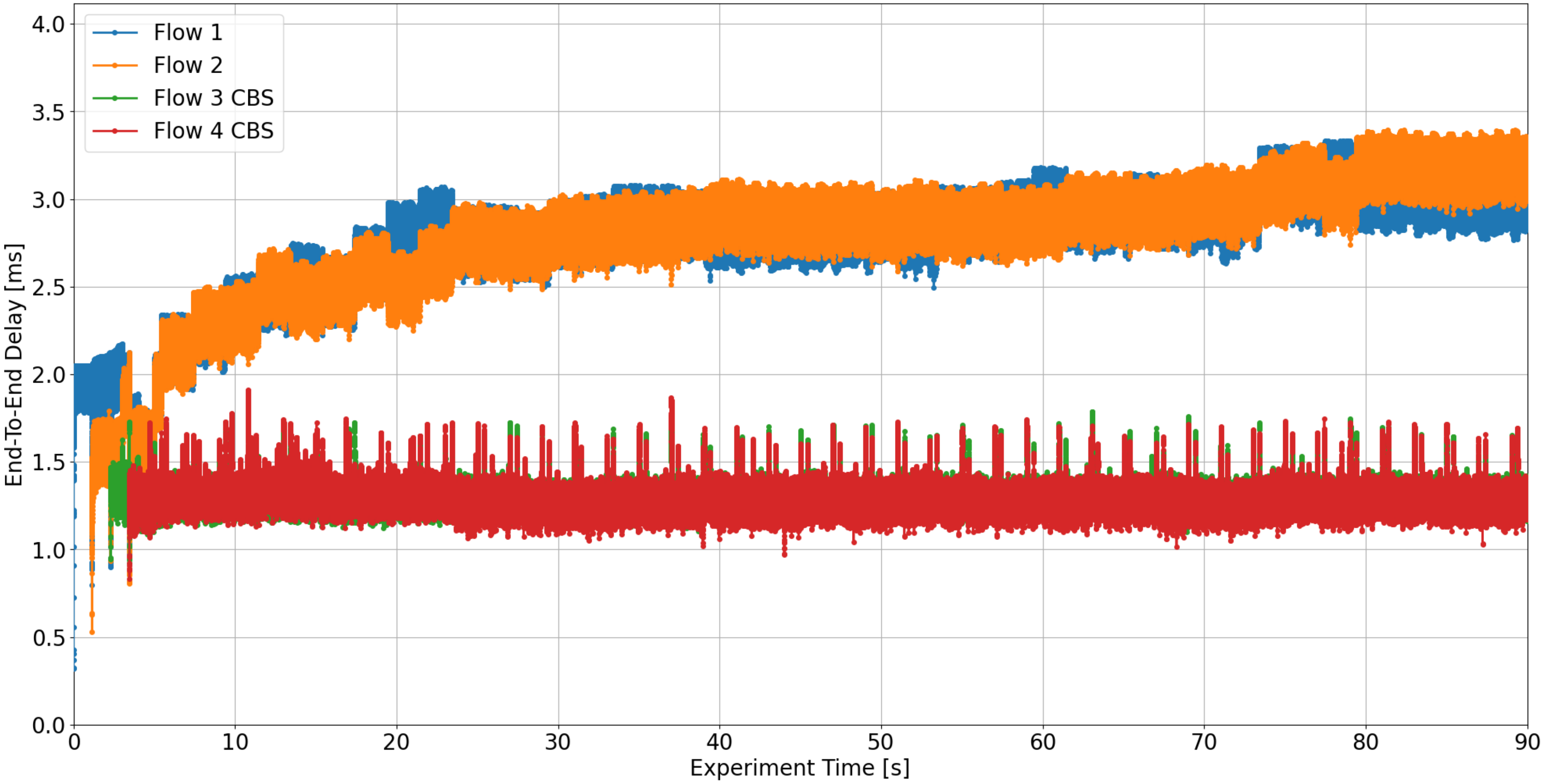
Step 3: Verify the Simple Scenario Results



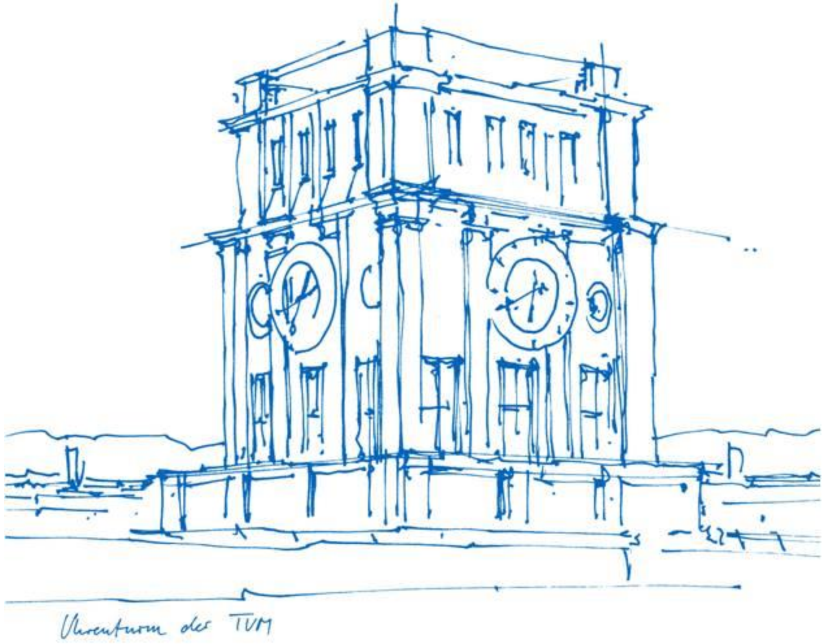
Outcome: Periodic “spikes” mitigated

System Optimization

Step 3: Verify the Complex Scenario Results



Outcome: Periodic “spikes” mitigated; bounded delay achieved



VALIDATION

EnGINE Validation

Can we support the required delay and jitter over 7 network hops?

Exemplary scenario

- Over up-to 7 hops (also fewer hops to demonstrate some of the challenges)
- **Credit-Based Shaper** – CBS (Also tested with **Time-Aware Priority Shaper** – TAPRIO)
- Interested in latency and jitter!
- Focus on IVNs → SR class A by Avnu Alliance

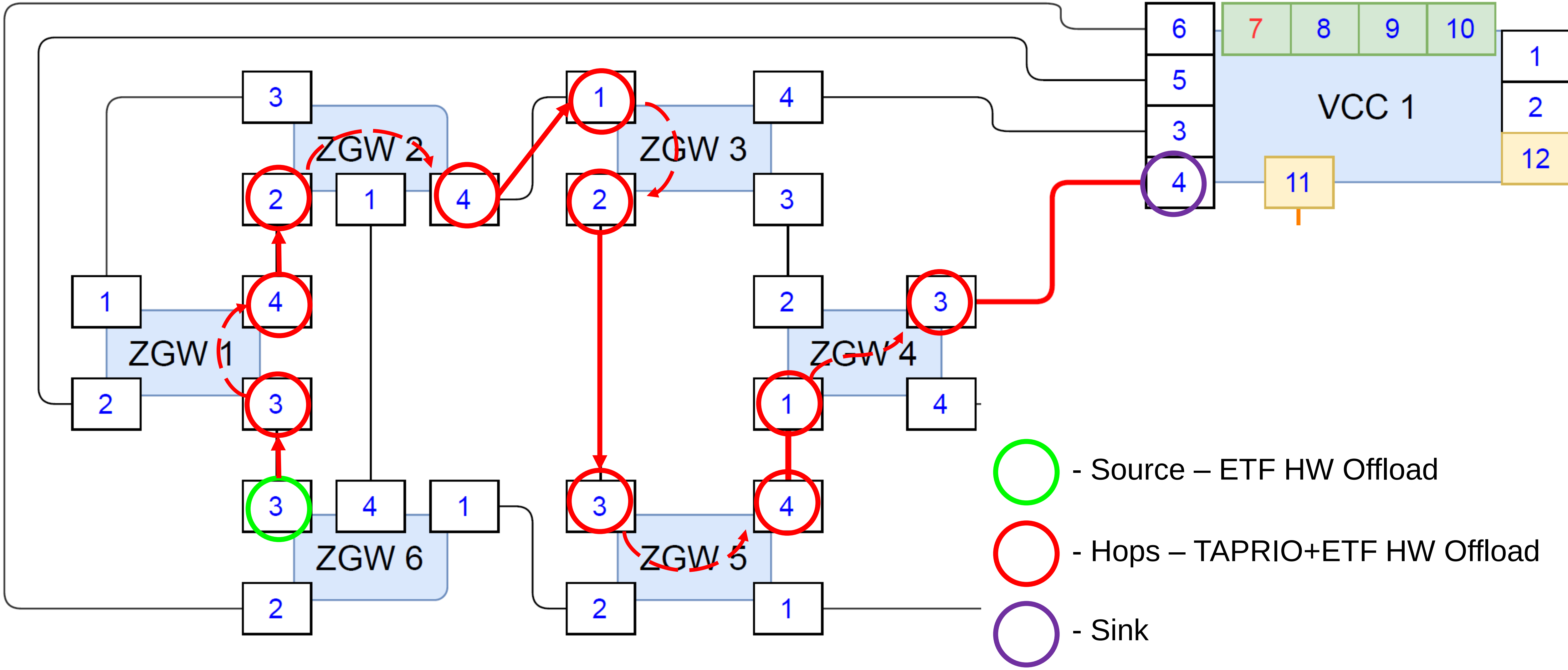
Traffic generation using Iperf3 – usually:

- Two (CBS) or one (TAPRIO) best-effort flows
  → Fill the link with best-effort traffic
- Two policed flows (SR A and SR B equivalent)
- Need to fulfil SR class A (and B) requirements

Class	Max Latency over 7 hops	CMI	Max Jitter
A	2 ms	125 us	125 us
B	50 ms	250 us	1000 us

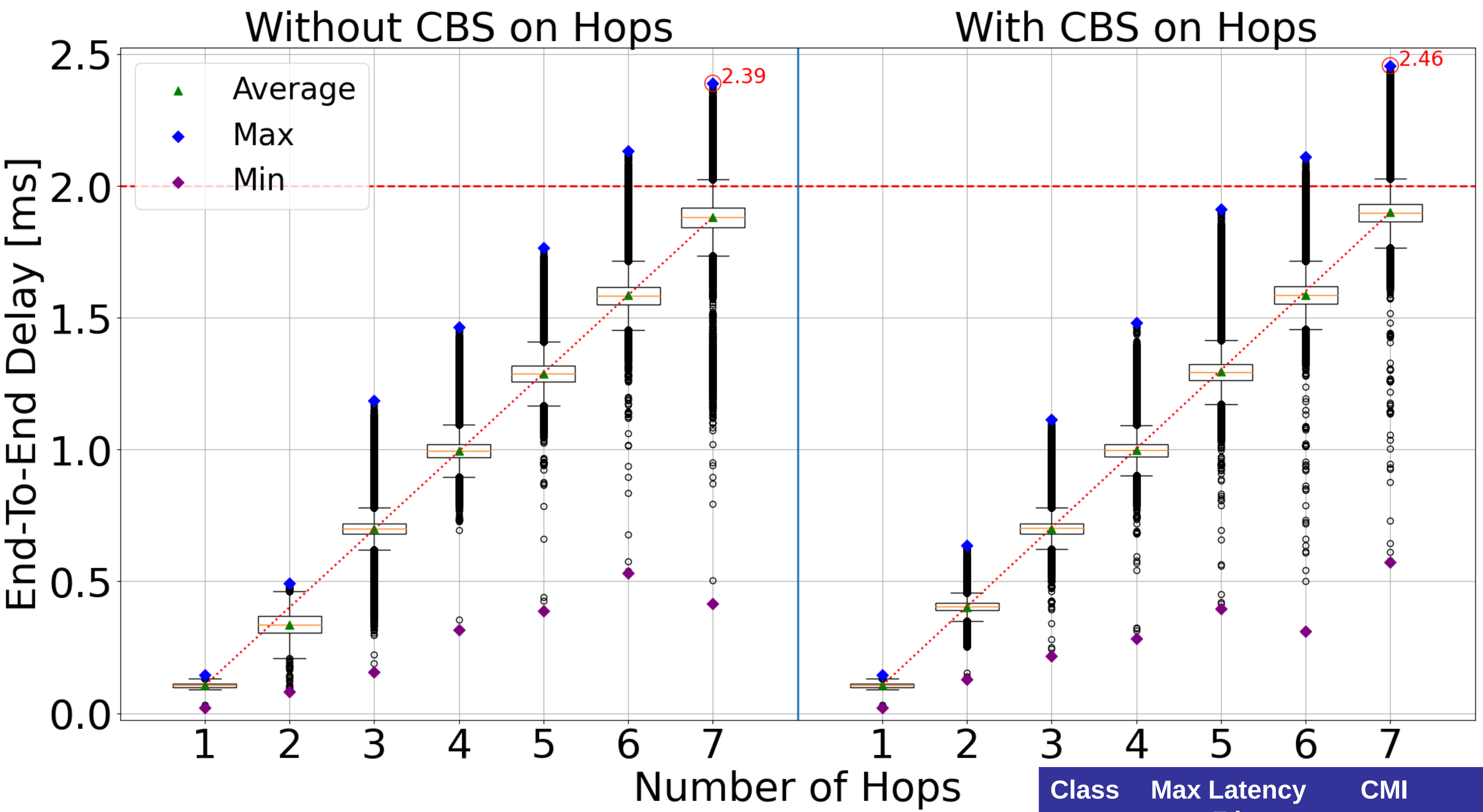
EnGINE Validation

An Example of a 6 Hops Flow



EnGINE Validation

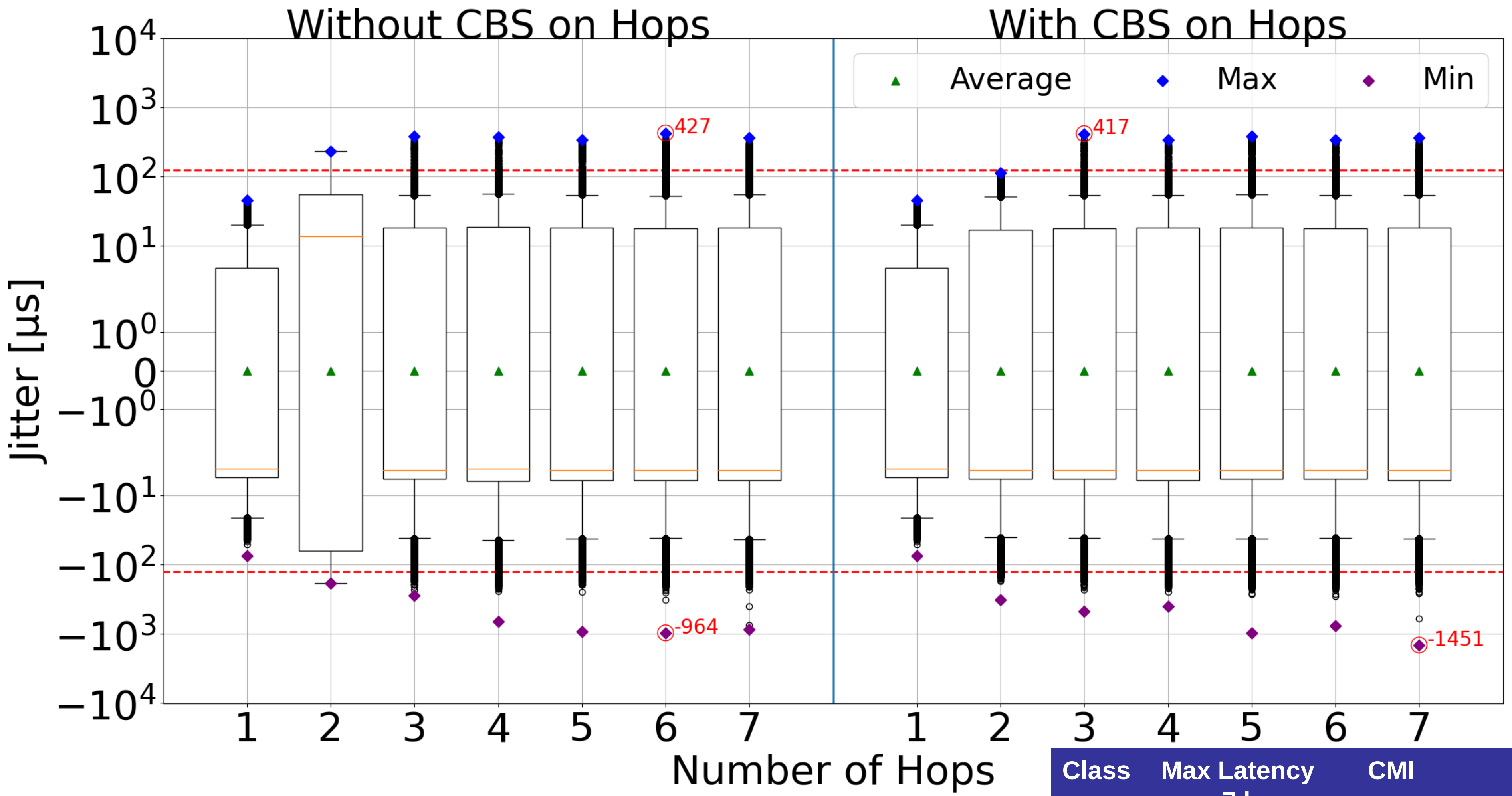
Results with CBS – End-To-End Delay of SR Class A Flow



Class	Max Latency over 7 hops	CMI	Max Jitter
A	2 ms	125 us	125 us
B	50 ms	250 us	1000 us

EnGINE Validation

Results with CBS – End-To-End Delay Jitter of SR Class A Flow



Class	Max Latency over 7 hops	CMI	Max Jitter
A	2 ms	125 us	125 us
B	50 ms	250 us	1000 us

EnGINE Validation
Sample Use-case – Summary

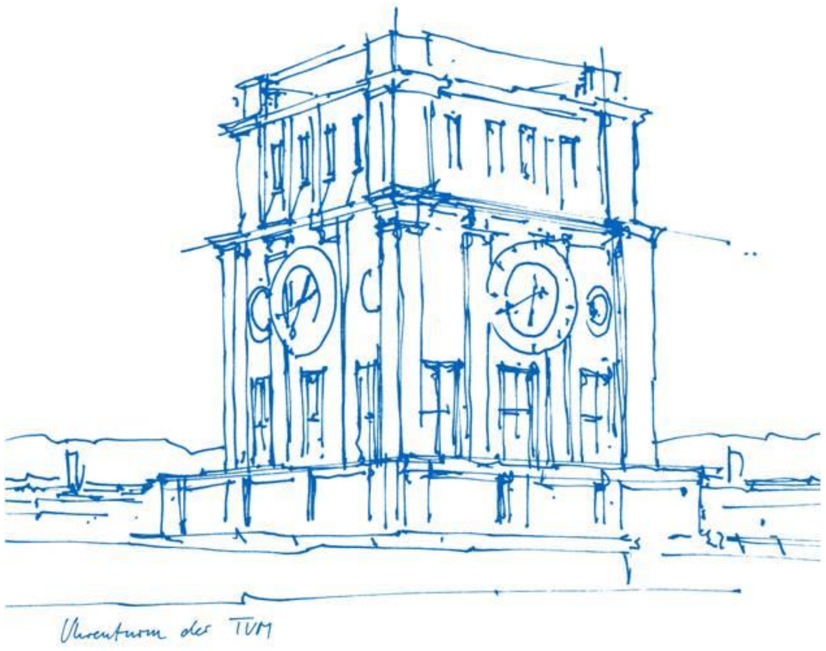
CBS

Class	Max Latency over 7 hops	CMI	Max Jitter
A	2 ms	125 us	125 us
B	50 ms	250 us	1000 us

- End-To-End delay for high priority policed flow mostly within the requirement, however, there are outliers exceeding the 2 ms target
- For all configurations, the maximum jitter exceeded the 125 us target
- Configuration of the qdisc on all hops provides a better bound on the jitter

TAPRIO

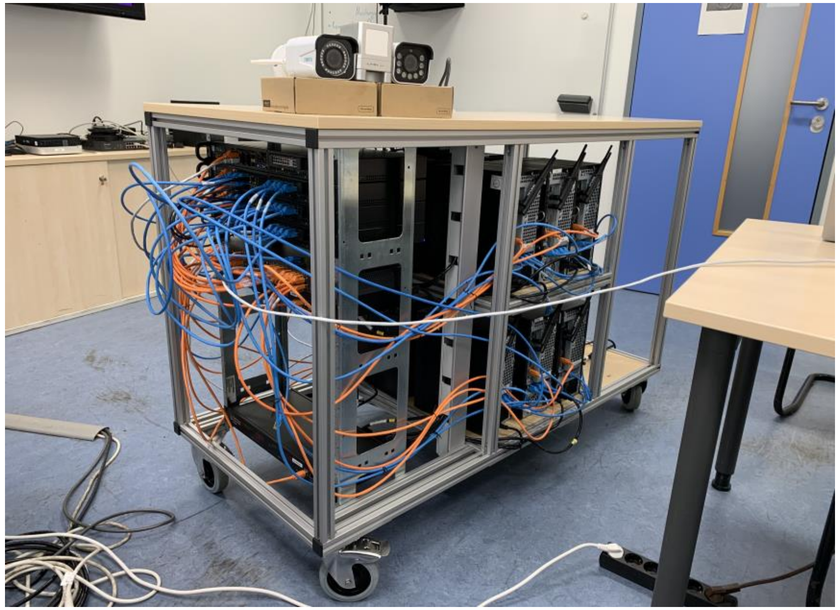
- End-To-End delay for TAPRIO flows mostly within the 2ms target for ETF deadline and strict modes
- Jitter for TAPRIO flows mostly values under 100µs



SUMMARY

Summary
EnGINE Properties

Realism
Diversity
Input & Output properties
Interpretability



Security
Collected insights
Reliability

Accessibility
Replicability
Configurability
Updatability
Testbed properties
Openness
Autonomy
Scalability

Questions?

Feel free to reach out via email to:

Filip Rezabek rezabek@in.tum.de

Marcin Bosk bosk@in.tum.de

References:

[1] Rezabek, F., Bosk, M., Paul, T., Holzinger, K., Gallenmüller, S., Gonzalez, A., ... & Ott, J. (2021). **EnGINE: Developing a Flexible Research Infrastructure for Reliable and Scalable Intra-Vehicular TSN Networks.** In *2021 17th International Conference on Network and Service Management (CNSM)* (pp. 530-536). IEEE.

[2] Bosk, M., Rezabek, F., Holzinger, K., Gonzalez, A., Kane, A., Fons, F., ... & Ott, J. (2021). **Environment for Generic In-vehicular Network Experiments-EnGINE.** In *2021 IEEE Vehicular Networking Conference (VNC)* (pp. 117-118). IEEE.

[3] Rezabek, F., Bosk, M., Paul, T., Holzinger, K., Gallenmüller, S., Gonzalez, A., ... & Ott, J. (2022). **EnGINE: Flexible Research Infrastructure for Reliable and Scalable Time Sensitive Networks.** In *Journal of Network and Systems Management (JNSM) Special Issue on High Precision, Predictable, Low-Latency Networking.*

[4] (Accepted) Bosk, M., Rezabek, F., Holzinger, K., Gonzalez, A., Fons, F., Kane, A., Ott, J., Carle, G. (2022). **Methodology and Infrastructure for TSN-based Reproducible Network Experiments.** In *IEEE Access.* IEEE.