Department of Informatics
Technical University of Munich

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

Authors:

Filip Rezabek rezabek@in.tum.de

Marcin Bosk bosk@in.tum.de

Thomas Paul paulth@in.tum.de

Kilian Holzinger holzingk@in.tum.de

Sebastian Gallenmüller gallenmu@in.tum.de

Angela Gonzalez angela.gonzalez.marino@huawei.com

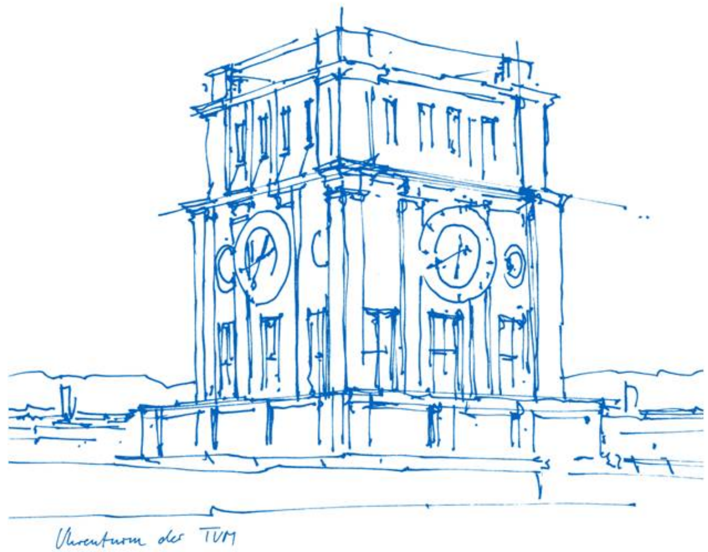
Abdoul Kane abdoul.aziz.kane@huawei.com

Francesc Fons francesc.fons@huawei.com

Zhang Haigang zhanghaigang@huawei.com

Georg Carle carle@in.tum.de

Jörg Ott ott@in.tum.de



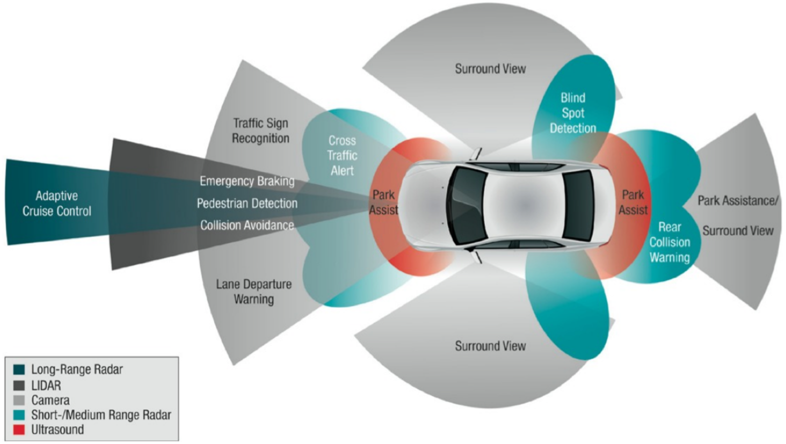
* Accepted to CNSM 2021 HiPNet workshop, 29.10.

Introduction
Motivation

Increased complexity of Intra-Vehicular Networks (IVN)

- Autonomous driving
- Safety mechanism
- Passenger entertainment
- V2X communication
- Maintenance and monitoring
- …

→ Usage of TSN

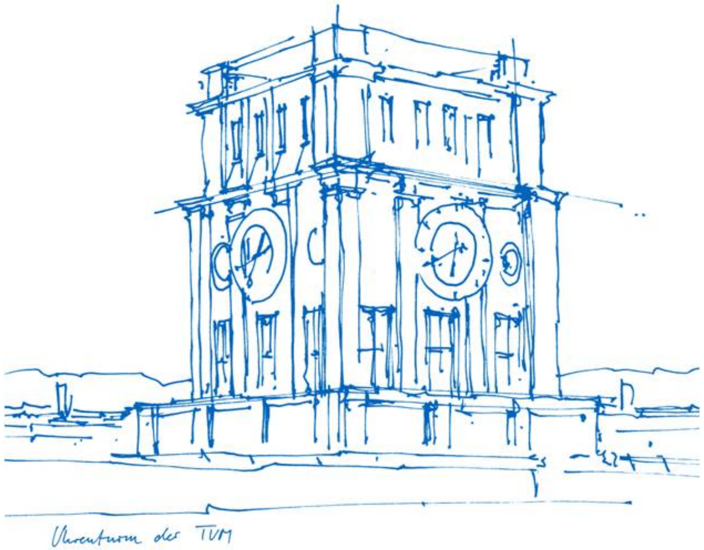


Structured approach to assessing the capabilities of IVNs with Time-sensitive networking

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

→ Identified that is hard to achieve **repeatability**, **reproducibility**, and **replicability** of TSN experiments

→ Challenge ***EnGINE*** works on



DESIGN OF ENGINE

EnGINE Design
Overview

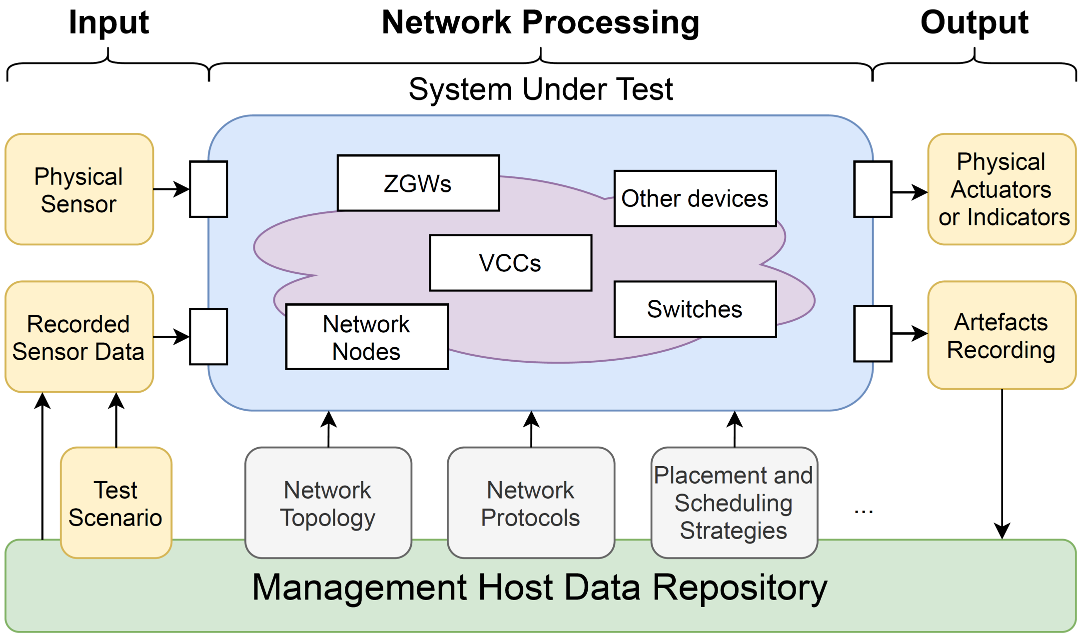
Orchestrated from the management host

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

Output
- Records experiment results
- Can include physical actuation



ZGWs – Zonal gateways
VCCs – Vehicle control computers

Design
Overview

15 Nodes

- 12 PCs – ZGWs
- 3 Servers – VCCs

NICs

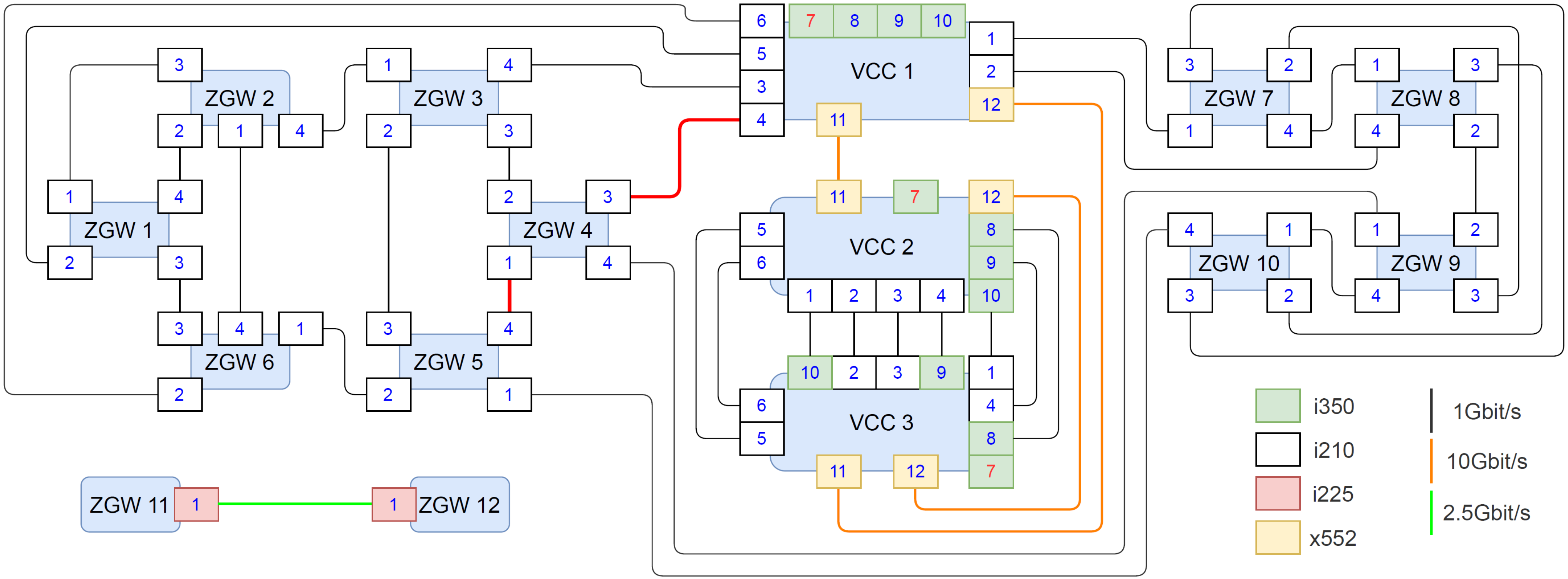
- Intel i210 – 1Gbit/s, 802.1{AS, Qav, Qbv}
- Intel i225 – 2.5Gbit/s, 802.1{AS, Qav, Qbv}
- Intel i350 – 1Gbit/s, 802.1AS
- Intel x552 – 10Gbit/s, None

Sensor

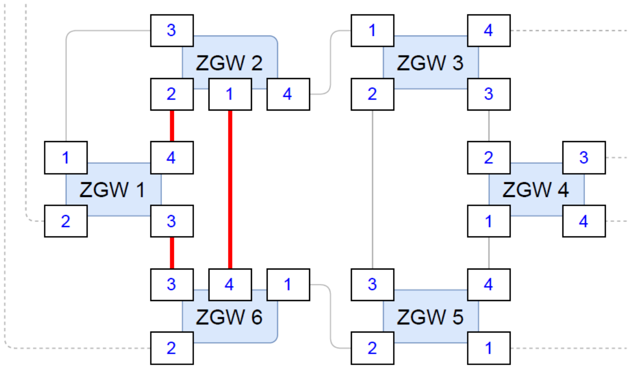
- LIDAR Livoxtech Mid 40

Other HW part of the testbed

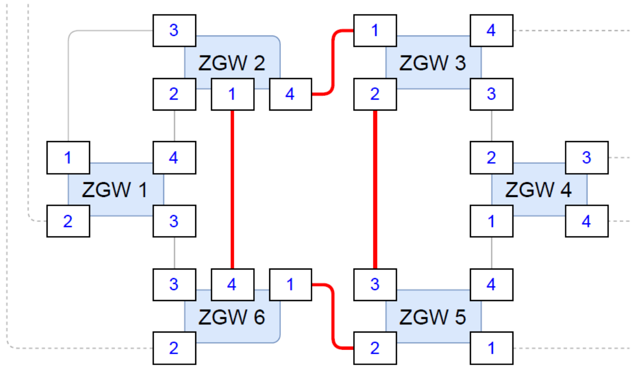
EnGINE Design
Overview - Physical Deployment



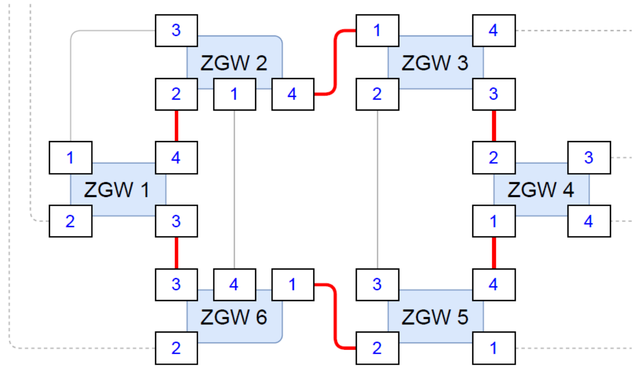
EnGINE Design
Overview – Flexible topology



(a) Low-End topology with 3 ZGWs



(b) Mid-End topology with 5 ZGWs



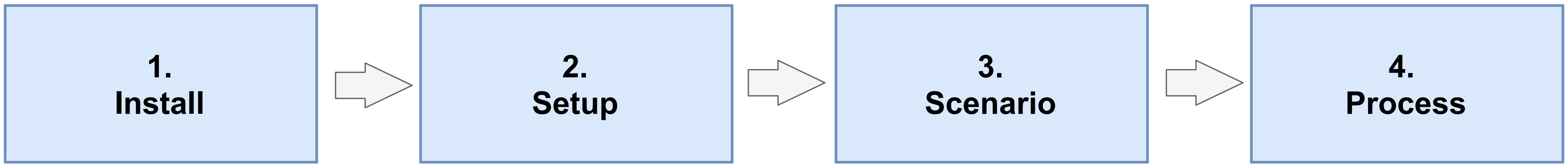
(a) High-End topology with 6 ZGWs

EnGINE Design
Configuration and Management

Four phases of experiment **campaigns**

Experiments within campaign independent of each other

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

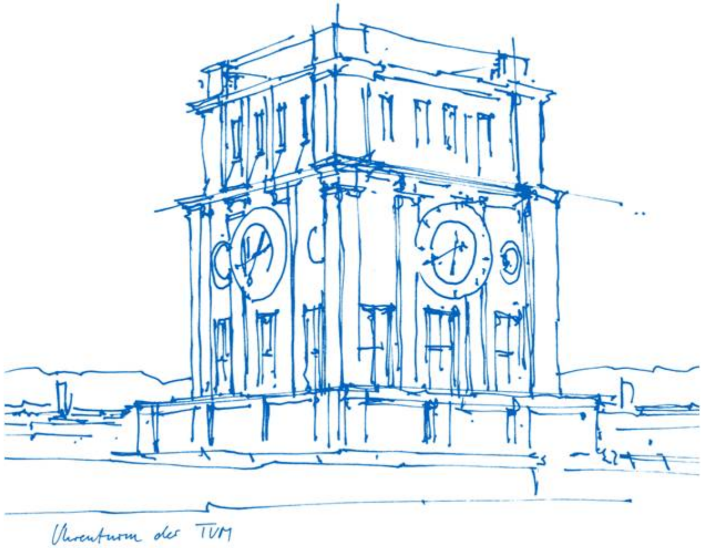


Install testbed nodes
Different images
e.g. RT kernel

Install additional packages, copy script files to every node, etc.

Loop through experiments: setup test environment, execute experiment

Post-process scenario artefacts



CAPABILITIES AND VALIDATION

EnGINE capabilities
TSN standards

802.1Qav – Credit-based shaper (CBS) algorithm – protects allocated BW

802.1AS – general precision time protocol (gPTP) for high precision clock synchronization

802.1Qbv – Traffic Priority (TAPRIO) shaper – separates traffic into individual time windows

Launch time feature – Earliest time first (ETF) – specifies when packets should be dequeued

→ In Linux implemented as a part of queuing disciplines (qdiscs)

→ Supported in HW and SW

EnGINE has granular control on which interface which configuration should be applied

Focus on IVNs

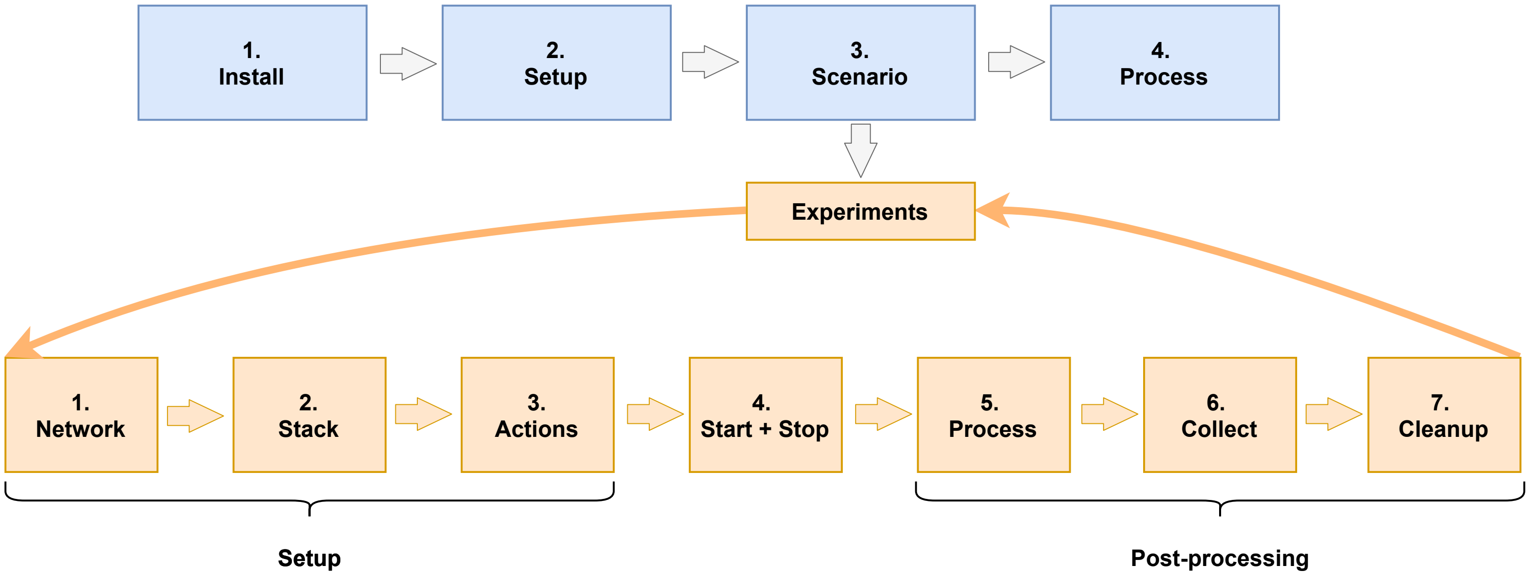
- Metrics categorized into stream reservation (SR) classes by the Avnu Alliance; latency and jitter
- IEEE P802.1DG - TSN Profile for Automotive In-Vehicle Ethernet Communications

EnGINE capabilities

Defining a scenario – sample use-case

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



EnGINE validation
Sample use-case – Overview

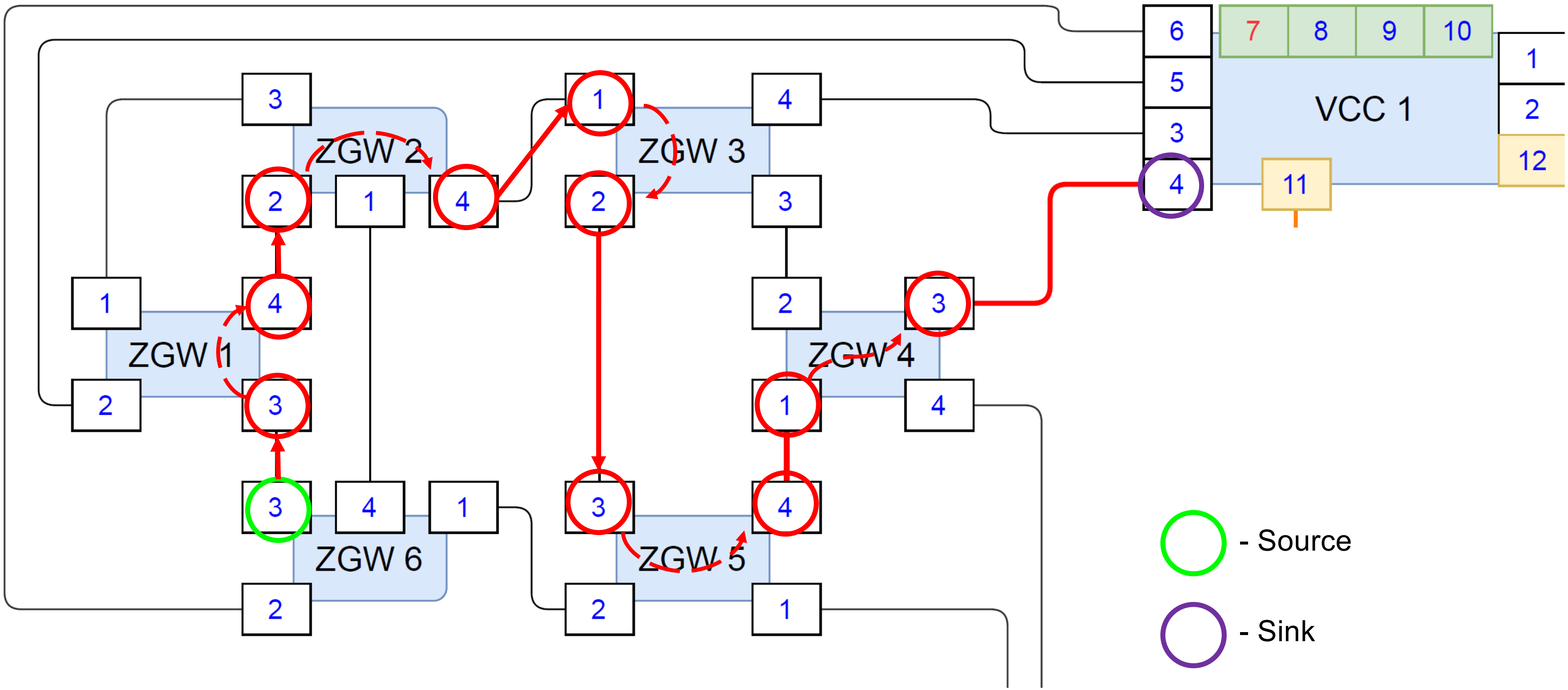
Show an example of a scenario

- Over 6 hops
- **Time-aware priority shaper (TAPRIO)** and Credit based shaper (CBS)
- Interested in latency and jitter

Using CPU isolation and CPU affinity

- Dedicated logical cores to relevant functions
- Assign a task/process/IRQ to a certain logical core

EnGINE validation
Sample use-case – 6 hops



EnGINE validation
Sample use-case – TARPIO setup

Time-aware priority (TAPRIO) shaper

Configured ETF on the source and TAPRIO on hop

Using ETF offload (NIC does the decision)

One full window cycle is always 1ms, 50us guard windows

Periodic traffic - 100us, 256B payload size

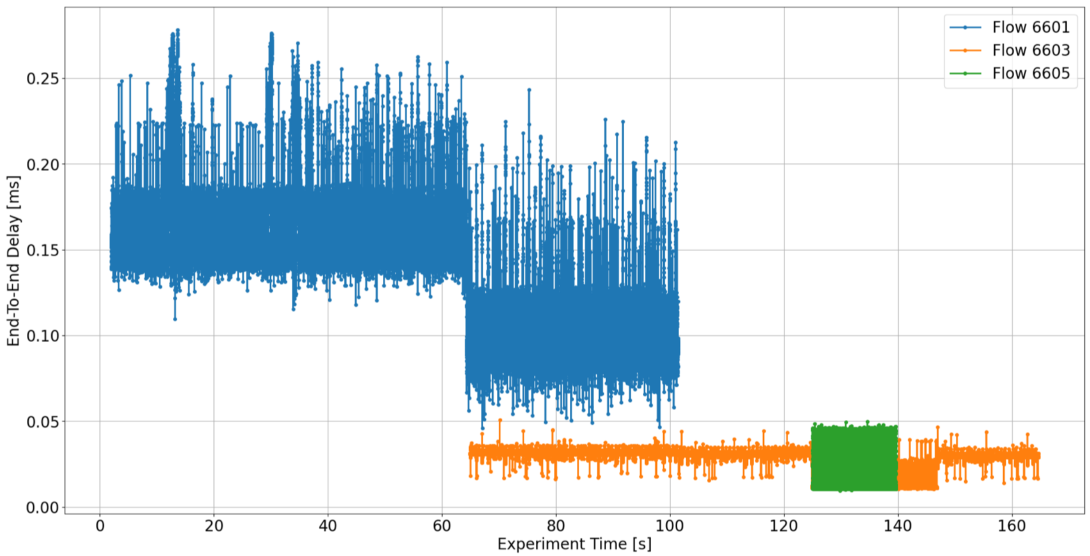
```
# Example Ascii windows:

# 2 |_______x___|........D___|bbbbbbbb|
# 1 |_______x___|........D___|bbbbbbbb|
#  0us       250us          550us     950us
```

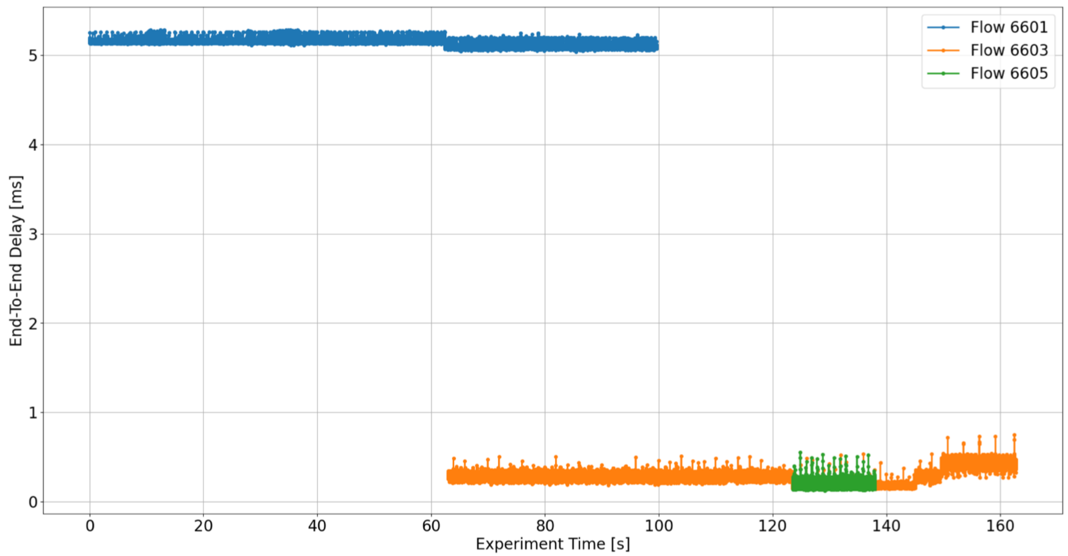
EnGINE validation

Sample use-case - TARPIO, Strict, Deadline, and Best effort - latency

1



6



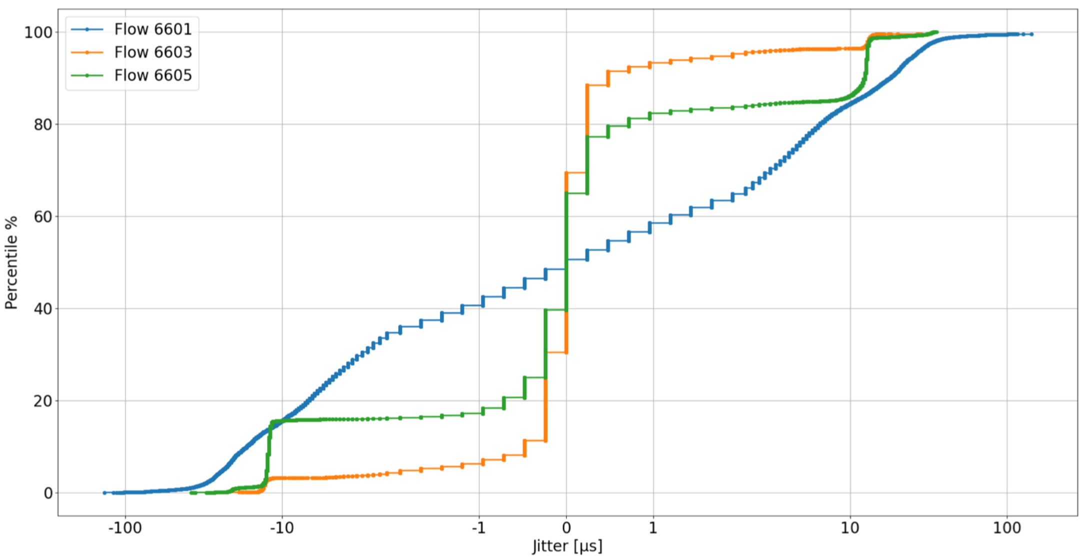
ETF Strict – Flow 6601
ETF Deadline – Flow 6603
Best effort – Flow 6605

HW offload → approx. 1ms increase per hop

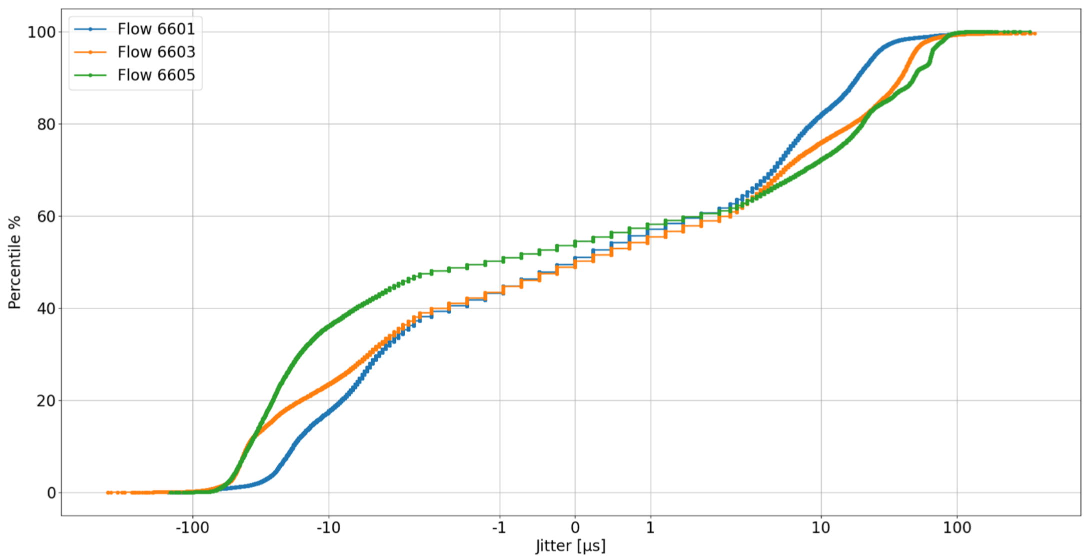
EnGINE validation

Sample use-case - TARPIO, Strict, Deadline, and Best effort - jitter

1



6



EnGINE validation
Sample uce-case - summary

TAPRIO

- End-To-End delay for TAPRIO flows mostly within the 2ms target for ETF deadline mode
- ETF strict increases delay as expected – ETF offload seems to result in enforced waiting time
- Jitter for TAPRIO flows with most values under 100µs

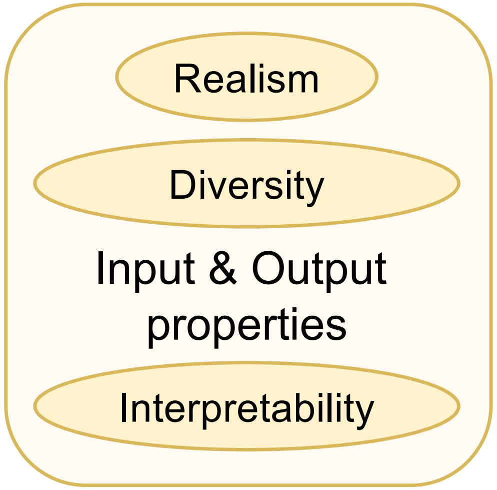
EnGINE
Properties overview

Input & Output properties
- Realism
- Diversity
- Interpretability

Collected insights
- Security
- Reliability

EnGINE/Testbed properties
- Accessibility
- Replicability
- Configurability
- Updatability
- Openness
- Autonomy
- Scalability

EnGINE
Properties overview

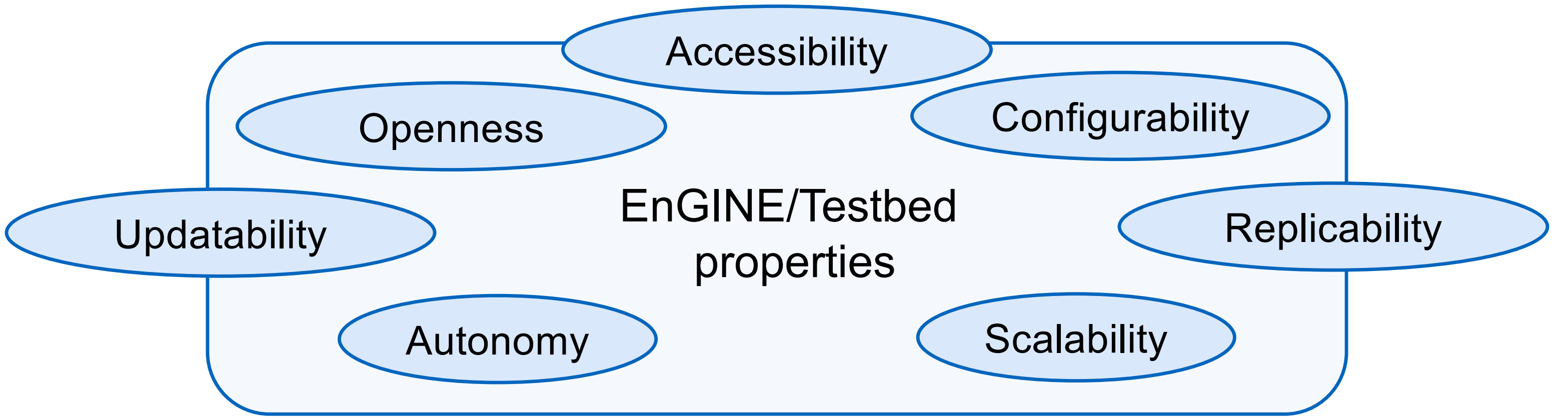


Realistic & **diverse** data source
→ LiDAR, RADAR, cameras, C&C

Known formats to **interpret** results
→ Packet captures, json, point clouds, csv

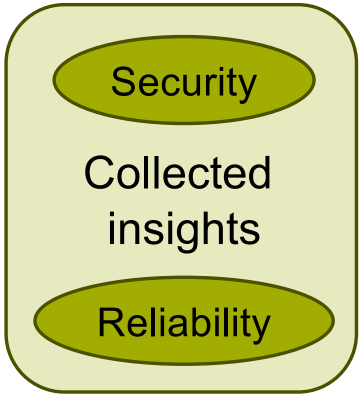
EnGINE
Properties overview

Easy to **extend/update**, using **COTS** and **open-source**
Configurable, **scalable**, and easily **replicable** experiments
Experiments are executed **autonomously**



EnGINE
Properties overview

Insights into timing guaranties
Identify crucial elements
→ ensure **reliability** and **security**



Summary & Future work

EnGINE - Flexible Research Infrastructure for Reliable and Scalable Intra-Vehicular TSN Networks

Introduced **EnGINE** with all its properties → research infrastructure for replicable TSN experiments

Utilizes open-source solutions coupled with commercial off-the-shelf hardware

Covered the experiments execution flow

Introduced few experiments covering a simple scenario

Future Work

Evaluate various traffic patterns and TSN configuration using ***EnGINE***

Verify that they fulfill IVN metrics (Avnu Alliance)

Compare results to related work

Focus on reliability aspects

Department of Informatics
Technical University of Munich

Questions?

Feel free to reach out via email to:

Filip Rezabek rezabek@in.tum.de

Marcin Bosk bosk@in.tum.de

References:

[1] M. Bosk et al. "Demo: Environment for Generic In-Vehicular Network Experiments - EnGINE". In: 13th *IEEE Vehicular Networking Conference (VNC).* 2021.