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

→ 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 – Environment for Generic In-Vehicular Network Experiments
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)

<table>
<thead>
<tr>
<th>NIC Type</th>
<th>NIC Speed</th>
<th>Supported IEEE 802.1 Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel i210</td>
<td>1Gbit/s</td>
<td>AS, Qav, Qbv</td>
</tr>
<tr>
<td>Intel i225</td>
<td>2.5Gbit/s</td>
<td>AS, Qav, Qbv</td>
</tr>
<tr>
<td>Intel i350</td>
<td>1Gbit/s</td>
<td>AS</td>
</tr>
<tr>
<td>Intel x552</td>
<td>10Gbit/s</td>
<td>AS</td>
</tr>
</tbody>
</table>

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
   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

Reproducible Layer 3-enabled TSN Experiments | Academic Salon on Low-Latency Communication, Programmable Network Components and In-Network Computation
CAPABILITIES
EnGINE capabilities
Supported TSN Standards

Traffic generators, packet captures

App → App → App

Transport Layer

Network Layer

Link Layer

OVS

Linux networking stack

Physical Layer

UDP
IPv4
802.1{Qav, Qbv, AS}, ETF
Ethernet

Supported TSN Standards

EnGINE capabilities
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Max Latency over 7 hops</th>
<th>CMI</th>
<th>Max Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 ms</td>
<td>125 us</td>
<td>125 us</td>
</tr>
<tr>
<td>B</td>
<td>50 ms</td>
<td>250 us</td>
<td>1000 us</td>
</tr>
</tbody>
</table>

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
Step 2: Artefact Verification

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

~60s
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

<table>
<thead>
<tr>
<th>Class</th>
<th>Max Latency over 7 hops</th>
<th>CMI</th>
<th>Max Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 ms</td>
<td>125 us</td>
<td>125 us</td>
</tr>
<tr>
<td>B</td>
<td>50 ms</td>
<td>250 us</td>
<td>1000 us</td>
</tr>
</tbody>
</table>
EnGINE Validation
An Example of a 6 Hops Flow

- Source – ETF HW Offload
- Hops – TAPRIO+ETF HW Offload
- Sink

Reproducible Layer 3-enabled TSN Experiments | Academic Salon on Low-Latency Communication, Programmable Network Components and In-Network Computation
EnGINE Validation
Results with CBS – End-To-End Delay of SR Class A Flow

Without CBS on Hops

<table>
<thead>
<tr>
<th>Class</th>
<th>Max Latency over 7 hops</th>
<th>CMI</th>
<th>Max Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 ms</td>
<td>125 us</td>
<td>125 us</td>
</tr>
<tr>
<td>B</td>
<td>50 ms</td>
<td>250 us</td>
<td>1000 us</td>
</tr>
</tbody>
</table>

With CBS on Hops

Average
Max
Min
EnGINE Validation
Results with CBS – End-To-End Delay Jitter of SR Class A Flow

Without CBS on Hops

With CBS on Hops

Number of Hops

<table>
<thead>
<tr>
<th>Class</th>
<th>Max Latency over 7 hops</th>
<th>CMI</th>
<th>Max Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 ms</td>
<td>125 us</td>
<td>125 us</td>
</tr>
<tr>
<td>B</td>
<td>50 ms</td>
<td>250 us</td>
<td>1000 us</td>
</tr>
</tbody>
</table>
EnGINE Validation
Sample Use-case – Summary

**CBS**
- 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

<table>
<thead>
<tr>
<th>Class</th>
<th>Max Latency over 7 hops</th>
<th>CMI</th>
<th>Max Jitter</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2 ms</td>
<td>125 us</td>
<td>125 us</td>
</tr>
<tr>
<td>B</td>
<td>50 ms</td>
<td>250 us</td>
<td>1000 us</td>
</tr>
</tbody>
</table>

**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
EnGiNE Properties

- Realism
- Diversity
- Interpretability

Input & Output properties

- Replicability
- Accessibility
- Configurability

Testbed properties

- Autonomy
- Openness
- Scalability

Security
Collected insights
Reliability

Collected insights
Questions?

Feel free to reach out via email to:
Filip Rezabek rezabek@in.tum.de
Marcin Bosk bosk@in.tum.de

References:

