DETERMINISTIC6G

Validation Results for DETERMINISTIC6G Concepts

D4.5

The DETERMINISTIC6G project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement no 1010965604.

Validation Results for DETERMINISTIC6G Concepts

Grant agreement number:	101096504
Project title:	Deterministic E2E communication with 6G
Project acronym:	DETERMINISTIC6G
Project website:	Deterministic6g.eu
Programme:	EU JU SNS Phase 1
Deliverable type:	Public Report
Deliverable reference number:	D4.5
Contributing workpackages:	WP4
Dissemination level:	PUBLIC
Due date:	30-06-2025
Actual submission date:	30-06-2025
Responsible organization:	USTUTT
Editor(s):	Frank Dürr (USTUTT) Simon Egger (USTUTT) Lucas Haug (USTUTT)
Version number:	V1.0
Status:	Final version
Short abstract:	Deliverable D4.5 describes the final validation results of the DETERMINISTIC6G project. Various novel concepts and mechanisms to enable future dependable 6G systems have been developed in the DETERMINISTIC6G project. One major goal of this document is to present validation results of these enablers, to evaluate their performance, and show their effectiveness and efficiency. Besides these enablers, also different application use cases depending on dependable 6G communication have been defined in the project. Another goal of this report is to present results evaluating the performance of these applications and to validate the enablers in the context of these use cases.
Keywords:	5G, 6G, TSN, validation, evaluation, simulator, emulator, scheduling, packet delay correction, PDC, exoskeleton, security, gPTP, wireless-aware, time-aware, time synchronization, hot standby, edge cloud, adaptive manufacturing, use case, networked control system

Contributor(s):	Joachim Sachs (EDD) Edgardo Montes de Oca (MI) Huu Nghia Nguyen (MI) Mahin Ahmed (SAL) Jose Costa-Requena (CMC) James Gross (KTH) Gourav Prateek Sharma (KTH) Samie Mostafavi (KTH) Niloofar Mehrnia (KTH)

	Frank Dürr (USTUTT) Simon Egger (USTUTT) Lucas Haug (USTUTT) Elena Mostovaya (USTUTT) János Harmatos (ETH) Marilet De Andrade Jardim (EAB) Oliver Höftberger (B&R) Filippo Dell' Agnello (IUVO) Francesco Giovacchini (IUVO) Emilio Trigili (SSSA) Lorenzo Amato (SSSA) Ines Alvarez Vadillo (ABB)

Reviewers:	Joachim Sachs (EDD) Ines Alvarez Vadillo (ABB) Oliver Höftberger (B&R)

Revision History

v0.1	Initial Draft.
v0.2	First version for internal review.
v0.3	Revised version for PMT review.
v1.0	Final version.

Disclaimer

Executive Summary

Deliverable D4.5 describes the final validation results of the DETERMINISTIC6G project. Various novel concepts and mechanisms to enable future dependable 6G systems have been developed in Work Package 2 and Work Package 3, whose effectiveness and performance are evaluated in this document. To this end, we use the validation framework developed as part of Work Package 4, in particular, the simulation framework, emulation frameworks, and delay distributions of 5G networks captured with the latency measurement framework, which are used by the simulation and emulation tools to realistically simulate and emulate the characteristic packet delay of 5G/6G networks. Besides presenting results for individual concepts and mechanisms, we also validate these concepts and mechanisms in the context of application scenarios, derived from use cases of Work Package 1, which also show the influence onto the performance of these applications.

In detail, validation results are presented for the following DETERMINSTIC6G concepts, mechanisms, and use cases:

- Wireless-Friendly Scheduling and Packet Delay Correction Validation
- Measurements and Characterization of RAN Latencies
- Time Synchronization Validation
- Security Validation of Time Synchronization Process
- Edge Cloud Validation including the validation of IEEE 802.1Qbv-aware traffic handling in Edge computing domains and eBPF-based time-aware traffic handling
- Exoskeleton Use Case Validation
- Adaptive Manufacturing Use Case Validation

Contents

1 Introduction

1.1 Objective of the Document

The DETERMINISTIC6G project has developed various concepts and mechanisms as enablers for future dependable 6G networks. One major goal of this document is to present validation results of these enablers, to evaluate their performance and show their effectiveness and efficiency.

Besides enablers, also different application use cases depending on dependable 6G communication have been defined in the project. Another goal of this report is to present results evaluating the performance of these applications and to validate the enablers in the context of these use cases.

In detail, validation results are presented for the following DETERMINSTIC6G concepts, mechanisms, and use cases:

- Wireless-Friendly Scheduling and Packet Delay Correction Validation
- Measurements and Characterization of RAN Latencies
- Time Synchronization Validation
- Security Validation of Time Synchronization Process
- Edge Cloud Validation including the validation of IEEE 802.1Qbv-aware traffic handling in Edge computing domains and eBPF-based time-aware traffic handling
- Exoskeleton Use Case Validation
- Adaptive Manufacturing Use Case Validation

Before we present the validation results, we start this report with a brief overview of the relation of this report to other work packages and an outline of the structure of the rest of the document.

1.2 Relation to Other Work Packages

This document has the following relations to other work packages as depicted in Figure 1-1.

The use cases and application scenarios validated in this document (exoskeleton, adaptive manufacturing), have been defined in WP1 as described in the deliverables D1.1 “DETERMINISTIC6G Use Cases and Architecture Principles“ [D6G-D1.1] and D1.3 “Report on Dependable Service Design“ [D6G-D1.3]. Moreover, D1.4 “Final Report on DETERMINISTIC6G Architecture – A Dependable Network Architecture for 6G” [D6G-D1.4] describes the final DETERMINISTIC6G architecture concepts, which complement the quantitative concept evaluation of this report.

WP2 has developed 6G-centric enablers for dependable communication, such as the characterization and prediction of RAN latencies, Packet Delay Correction mechanisms, and security mechanisms, which are validated in this document. In this report, we will briefly summarize the concepts and mechanisms to be validated; further technical details about these mechanisms can be found in the following deliverables: D2.1 “First report on 6G centric enabler” [D6G-D2.1], D2.2 “First Report on the time synchronization for E2E time awareness” [D6G-D2.2], D2.3 “Second report on 6G centric enabler” [D6G-D2.3], and D2.4 “Second report on the time synchronization for E2E time awareness” [D6G-D2.4].

WP3 has developed enablers of 6G convergence for dependable communication, including wireless-friendly end-to-end scheduling mechanisms and edge cloud mechanisms, which are validated in this document. Again, technical details on these concepts and mechanism to be validated can be found in various other deliverables: D3.1 “Report on 6G Convergence Enablers Towards Deterministic

Communication Standards" [D6G-D3.1], D3.2 "Report on the Security Solutions" [D6G-D3.2], D3.3 "Report on Deterministic Edge Computing and Situational Awareness via Digital Twining Security Solution" [D6G-D3.3], D3.4 "Optimized Deterministic End-to-End Schedules for Dynamic Systems" [D6G-D3.4], and D3.6 "Report on Deterministic Edge Computing and Situational Awareness via Digital Twining" [D6G-D3.6].

Finally, WP4 has contributed the validation frameworks for simulating and emulating the concepts and mechanisms mentioned above as described in deliverables D4.1 "DETERMINISTIC6G DetCom Simulator Framework Release 1" [D6G-D4.1] and D4.4 "DETERMINISTIC6G DetCom Simulator Framework Release 2" [D6G-D4.4], as well as the characteristic latency data for 5G/6G networks as described in the deliverables D4.2 "Latency Measurement Framework" [D6G-D4.2] and D4.3 "Latency Measurement Data and Characterization of RAN Latency from Experimental Trials" [D6G-D4.3].

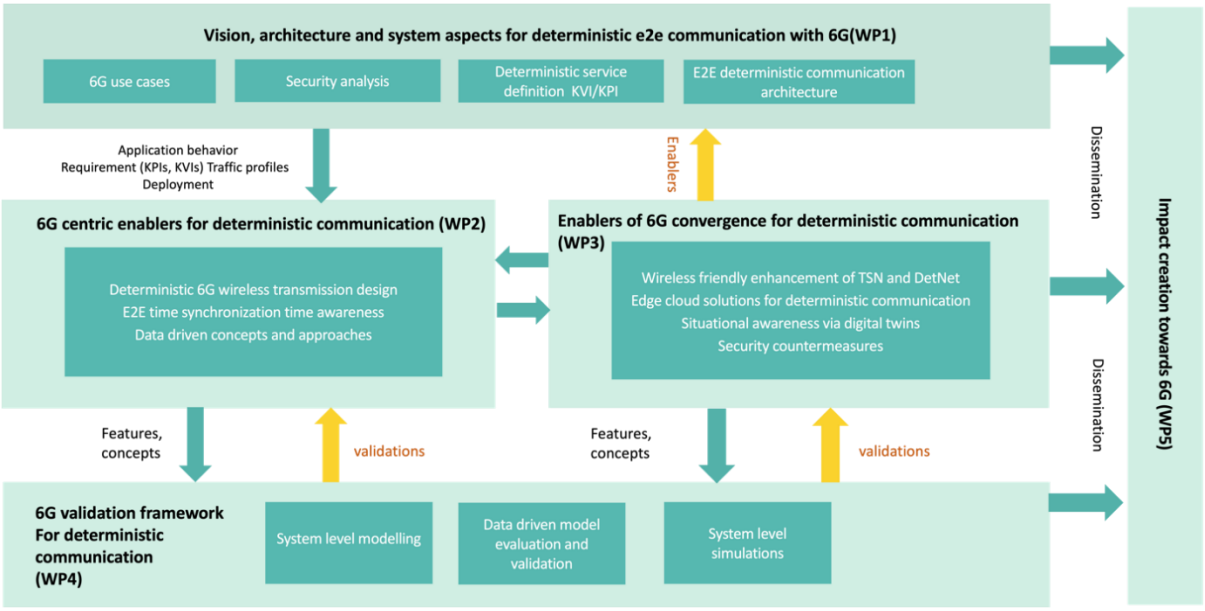


Figure 1-1: Relation of WP4 to other work packages

1.3 Structure and Scope of the Document

The rest of this document is structured as follows.

Section 2 gives an overview of the validation frameworks used to produce the validation results. The major purpose of this section is to keep this report self-contained such that the validation methodology can be understood without first reading various other documents. More technical information about the frameworks can be found in the deliverables D4.1 "DETERMINISTIC6G DetCom Simulator Framework Release 1" [D6G-D4.1], D4.2 "Latency Measurement Framework" [D6G-D4.2], D4.3 "Latency Measurement Data and Characterization of RAN Latency from Experimental Trials" [D6G-D4.3], and D4.4 "DETERMINISTIC6G DetCom Simulator Framework Release 2" [D6G-D4.4].

The main part of this document is Section 3, which presents the validation results in detail.

2 Overview of the Validation Frameworks

Within the DETERMINISTIC6G project, multiple validation frameworks have been developed. These were already explained in detail in the previous deliverables D4.2 "Latency Measurement Framework"

[D6G-D4.2], and both simulator releases D4.1 “DETERMINISTIC6G DetCom Simulator Framework Release 1” [D6G-D4.1] and D4.4 “DETERMINISITC6G DetCom Simulator Framework Release 2” [D6G-D4.4]. As the validation results presented in this deliverable are mostly based on these frameworks, we provide a brief overview of our validation frameworks, namely the 6GDetCom Simulator, 6GDetCom Emulator, the Latency Measurement Framework, and the Security Emulation Framework, in this section.

2.1 6GDetCom Simulator

The main objective of the 6GDetCom Simulator is to provide a simulation platform for converged 6G/TSN networks that allows for the evaluation of features and concepts at an early development stage. The initial release of the 6GDetCom Simulator was developed in WP4 and first presented in D4.1 “DETERMINISTIC6G DetCom Simulator Framework Release 1” [D6G-D4.1]. An extended and refined version was released and described in D4.4 “DETERMINISTIC6G DetCom Simulator Framework Release 2” [D6G-D4.4].

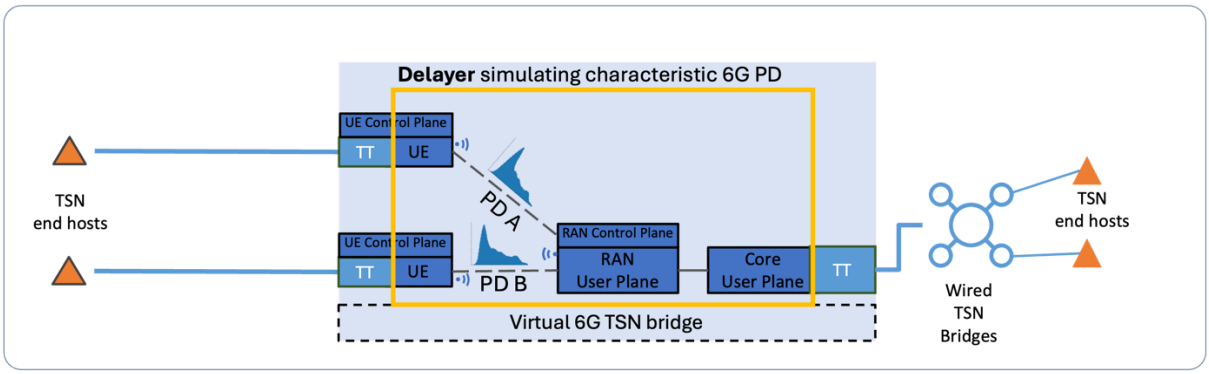


Figure 2-1: 6GDetCom Simulator Architecture.

The 6GDetCom Simulator is based on the OMNeT++ [OMN25] simulation framework and utilizes the TSN features and modules of the INET [INE25] framework. It extends these existing modules with a so-called DetCom node acting as a logical TSN bridge (see Figure 2-1). This DetCom node consists of one network-side TSN Translator (NW-TT) and an arbitrary number of device-side TSN Translators (DS-TTs). These TTs act as “half-bridges”, providing ingress TSN features (e.g. Per Stream Filtering and Policing (PSFP)) at the ingress TT, and egress TSN features (e.g. the Time-Aware Shaper (TAS)) on the egress TT. Thus, the whole DetCom node acts like a logical TSN bridge.

Instead of implementing sophisticated simulation models of the 5G/6G internals, such as Hybrid Automatic Repeat-Requests (HARQ), physical layer models, etc., our implementation follows a data-driven approach. In this data-driven approach, we utilize measurements from real 5G systems using our Latency Measurement Framework (see Section 2.3) and implement them in the simulator to produce the same delay patterns. To this end, the DS-TTs in our DetCom node contain a Delayer module (yellow box in Figure 2-1) which allows for specifying the uplink and downlink delay characteristics individually for each DS-TT. Our approach allows for specifying the delay characteristics in the following ways:

Packet Delay Histograms: Our simulator supports drawing random delay values from provided histogram files, which can be generated from real-world measurements.

Replaying Delay Traces: Our simulator supports replaying delay traces of real-world measurements. These can be either replayed in order or based on the current simulation timestamp.

Closed Form Distributions: Our simulator allows to delay frames on built-in probability distributions, e.g. a normal distribution.

Random Walk Process: Our simulator allows to produce delay values based on a random walk process.

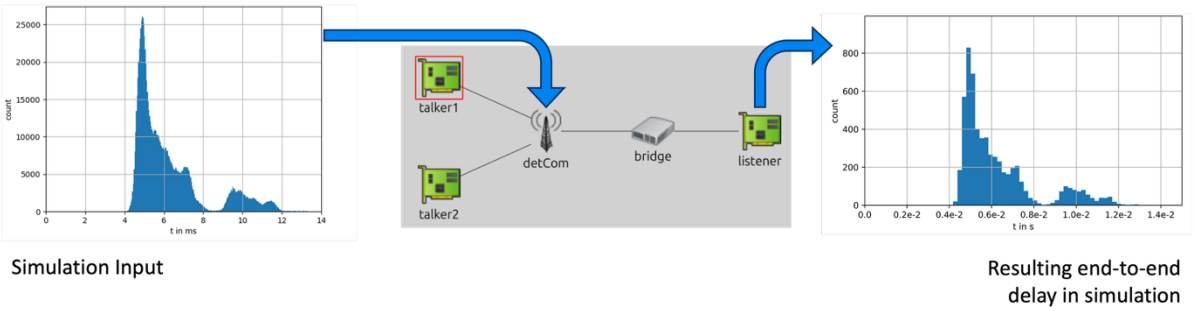


Figure 2-2: Example of a data-driven simulation

Figure 2-2 shows an example of a data-driven simulation with the DetCom node being configured to use a real-world 5G measurement on the left side and the resulting end to end delay of frames on the right side.

The above architecture allows for simulating the effect of the characteristic packet delay of a 6G logical bridge onto the TSN system in converged 6G/TSN networks. To also enable the evaluation of novel 6G concepts and their effects, our 6GDetCom Simulator implements the following additional concepts:

Packet Delay Correction (PDC): PDC was introduced in [D6G-D2.1] and [D6G-D2.3] and allows to buffer frames at the outgoing TSN Translator in order to reduce the variation of the packet delay distribution. The 6GDetCom Simulator directly implements the timestamped approach of PDC as well as a simplified virtual timeslot-based version of PDC. The implementation details of PDC are described in [D6G-D4.4]. Results using our PDC implementation are provided in Section 3.1.

Time Synchronization: gPTP-based time synchronization in converged 6G/TSN networks requires the DetCom node to support additional functionality. For example, this includes timestamping using the internal 6G clock to calculate the residence time of gPTP frames within the Logical 5G TSN bridge. A detailed overview of the time sync architecture can be found in [D6G-D2.2] and [D6G-D2.4]. Our simulation framework implements this architecture. Furthermore, it extends the gPTP implementation of INET with support of different clock servos, the Best Transmitter Clock Algorithm (BTCA) and Hot Standby mechanisms. The implementation details are provided in [D6G-D4.4]. Time synchronization results using this implementation are later presented in Section 3.3.

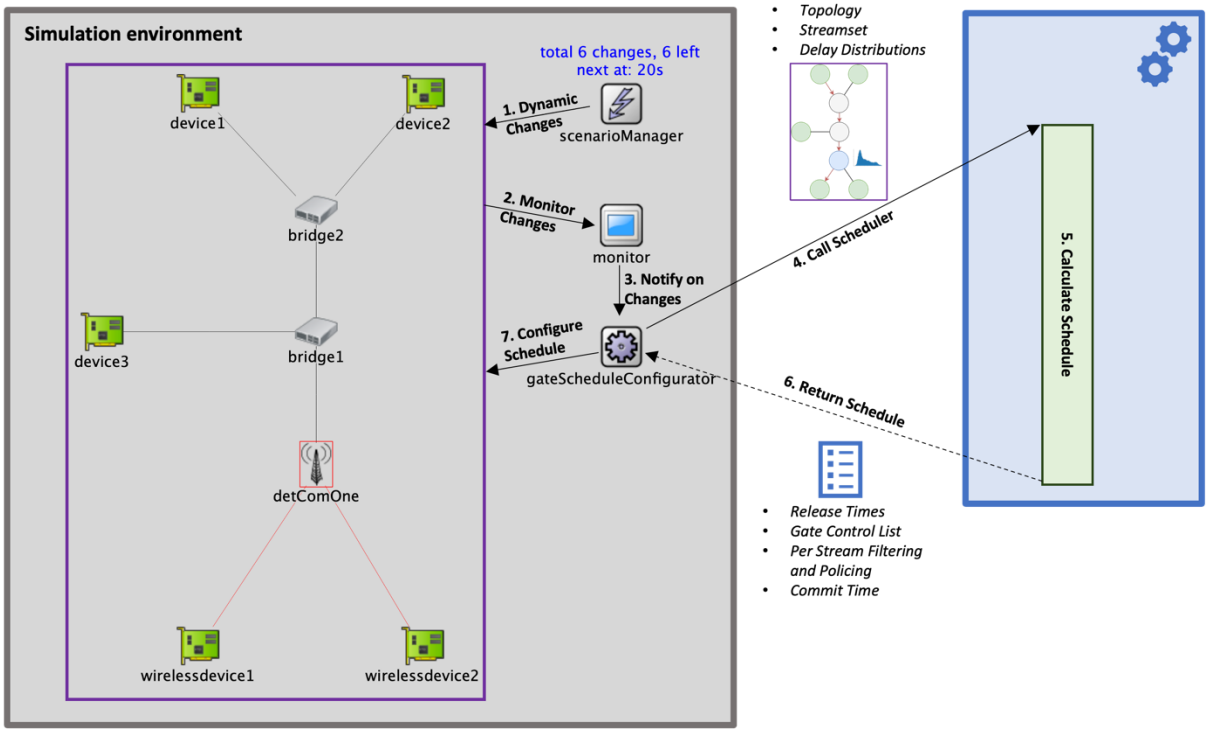


Figure 2-3: Control plane integration

Lastly, our 6GDetCom Simulator also provides a control plane integration using a direct scheduler interface. An overview of our control plane interface is provided in Figure 2-3:

1. The *scenarioManager* of INET is used to specify the dynamic behavior of the simulation. It allows to re-configure the parameters (such as the sending rate or packet size) of our newly introduced *DynamicSourceApp* as well as enabling and disabling them. Furthermore, it allows to modify the delay distributions of the DetCom node.
2. The *monitor* module monitors the simulation for changes executed by the *scenarioManager* in Step 1.
3. The *monitor* notifies the *gateScheduleConfigurator* directly upon detecting a change or after an optional configurable delay to gather multiple changes into one notification.
4. The *gateScheduleConfigurator* writes the schedule input files with information about the current scenario (i.e. the topology, stream set and delay distributions) and calls an external scheduler program (e.g. a Python script).
5. The scheduler uses the provided input files and calculates a schedule.
6. Upon finishing the calculation, the scheduler writes output files, including the release times of all apps, the Gate Control Lists, as well as the Per Stream Filtering and Policing configurations. Furthermore, it provides the simulation with a commit timestamp (in the simulation time domain) which tells when the new configuration should be applied in the simulation.
7. The *gateScheduleConfigurator* applies the configuration given by the scheduler at the given commit time.

The implementation of our control plane model is described in detail in deliverable D4.4 [D6G-D4.4]. This deliverable also includes a detailed architectural description as well as an overview of all configuration options of the previously mentioned modules.

2.2 6GDetCom Delay Emulator

While our simulator framework described above allows for the simulation and analysis of novel 6G features, it is not straightforward to simulate *real* applications, which would have to be translated to sophisticated simulations models first. To facilitate the testing and evaluation of real applications under 5G/6G networking conditions, we developed the 6GDetCom Emulator, which emulates the characteristic packet delay (PD) of a virtual 6G bridge or the end-to-end delay of an entire TSN network without the need for dedicated 5G/6G hardware. In particular, the 6GDetCom Emulator is used later for the validation of the exoskeleton use case (see Figure 3-33). In the following, we give a brief overview of the 6GDetCom Emulator. More technical details about the emulator can be found in D4.4 [D6G-D4.4]. A detailed tutorial on how to use and setup the 6GDetCom Emulator can be found in the GitHub repository[1], as well as in the DETERMINISITC6G blog post[2] describing the network delay emulator.

The core of the emulator is a Linux Queueing Discipline (QDisc) called sch_delay that can be assigned to network interfaces to add artificial delay to all packets leaving through this network interface. Figure 2-4 shows the system architecture consisting of two major parts: the QDisc running in the kernel space, and a user-space application providing individual delays for each transmitted packet through a character device. The provided delays are buffered in the QDisc, such that delay values are available immediately when new packets arrive. Whenever a packet is to be transmitted through the network interface, the next delay value is dequeued and applied to the packet before passing it on to the network interface (TX queue).

[1] https://github.com/DETERMINISTIC6G/6GDetCom_Emulator
[2] https://blog.deterministic6g.eu/posts/2024/10/26/network_delay_emulator.html

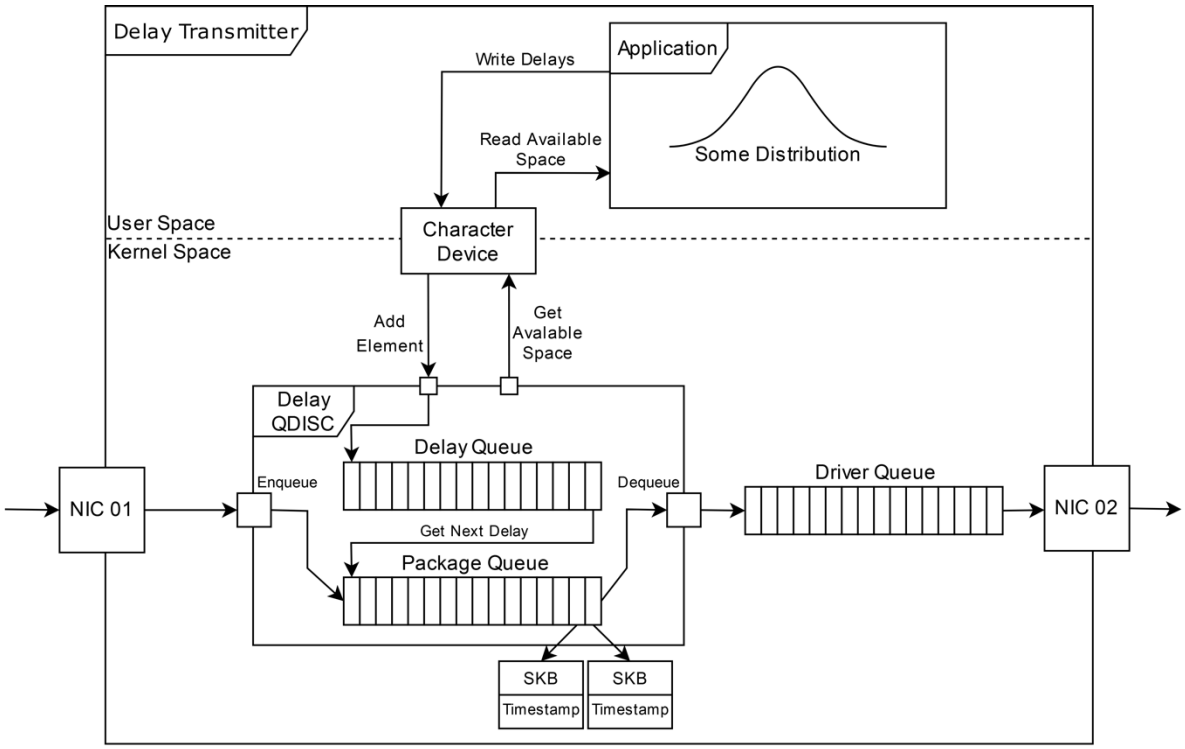


Figure 2-4: Architecture of the 6GDetCom Emulator

The QDisc can also be applied to network interfaces that are assigned to a virtual bridge on the emulation host to apply individual delay distributions to packets forwarded through different egress interfaces as shown in Figure 2-5, e.g., in uplink and downlink direction of an emulated 5G/6G network. This allows for emulating the end-to-end network delay of a whole emulated network with a single Linux machine within the scalability limits of the emulation host.

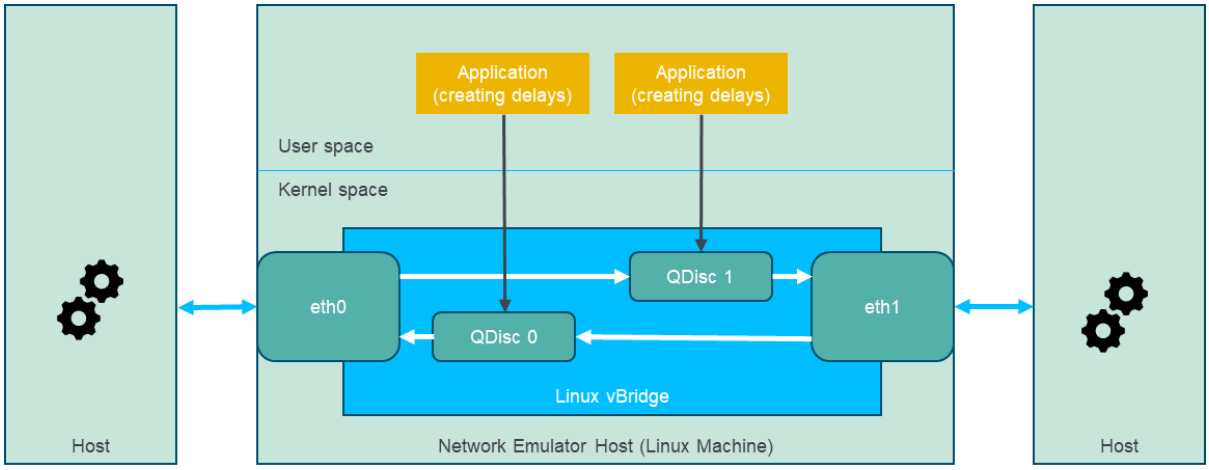


Figure 2-5: Network emulator with individual delay distributions on individual egress interfaces

Figure 2-6 shows the histogram of an emulated wireless delay distribution versus the given reference delay distribution to be emulated. Although we can observe a slight shift of the emulated distribution compared to the reference distribution, our evaluations of the emulation tool show that it can reproduce our target wireless delay distributions with sufficient accuracy. The throughput is limited to about 1 Gbps and the minimum delay that can be emulated accurately is about 100 micro-seconds,

whereas our wireless delay distributions are in the range of milli-seconds. Therefore, although this software solution might not be sufficient to emulate wireline delays in the range of micro-seconds, it is sufficient for emulating the PD of wireless 5G/6G networks.

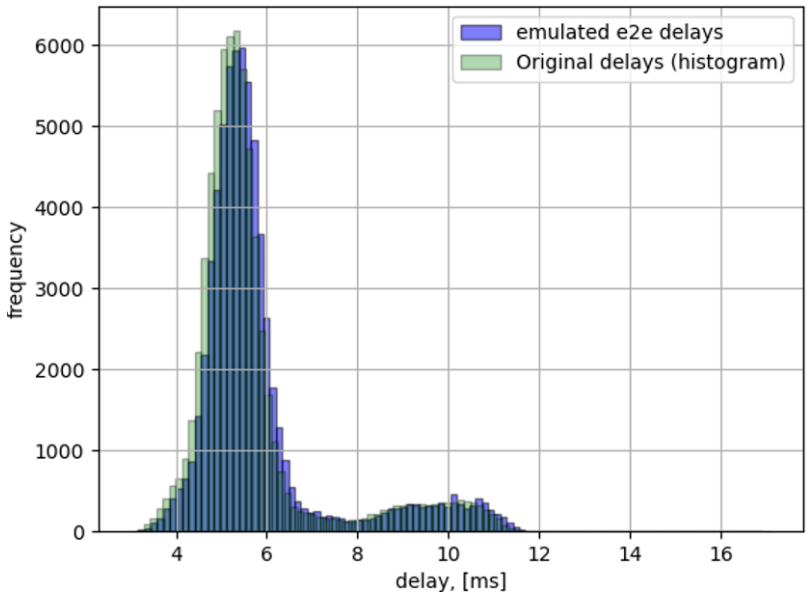


Figure 2-6: Emulated packet delay of wireless network vs. given reference delay

2.3 Latency Measurement Framework and Latency Measurements

The design of the latency measurement framework is based on the following key principles:

- *Out-of-band Time Synchronization*: To ensure accuracy in one-way delay measurements, the framework employs an out-of-band mechanism for time synchronization. This ensures that all components of the framework use the same time reference when timestamping packets / network conditions.
- *Microservice Architecture*: By adopting a microservice architecture, the framework ensures that different software components and processes are decoupled and can operate independently. This architecture supports scalability, resilience, and ease of deployment, which are crucial for handling the prolonged measurements in 5G networks.
- *Out of Band Data Collection*: Similar to time synchronization, the framework employs an out-of-band approach for data collection. In other words, the measurement data is transported from the measurement points to the data collection and aggregation module via channels distinct from the existing user plane traffic. This method helps in avoiding interference with the measurement process and ensures that the collected data is accurate and not affected by the measurement traffic itself.

The implementation of the latency measurement framework based on the above design involves several components and processes, each aimed at accomplishing a certain design goal:

- *Time Synchronization*: The time synchronization of various components in the framework is accomplished using a Grandmaster Clock (GM) connected to a Global Navigation Satellite System (GNSS) antenna. An out-of-band Ethernet network is used to distribute time synchronization messages of PTP to all hosts involved in the measurement setup. All hosts

were equipped with hardware timestamping-capable network interface cards (NICs) to PTP on an out-of-band network.

- *Traffic Generation and Timestamping*: The framework includes a traffic generator that injects network traffic into the 5G system that can mimic real-world application scenarios. The traffic generator is deployed in a client-server model, i.e., a UDP client instance sends traffic with a certain configurable period to the UDP server instance. In addition to traffic generation, the packets are timestamped at the client and the server. This approach helps in measuring the latency experienced by packets as they traverse through the network.
- *COTS 5G Measurement Points*: The Consumer Off the Shelf (COTS) 5G system currently does not have capabilities to collect metadata at the COTS base stations. Therefore, a measurement service was developed to be used at COTS UE devices to act as a measurement point. The proposed measurement service is capable of recording a few network conditions (e.g., reference signal receive power (RSRP) and reference signal receive quality (RSRQ) and exposing it on an HTTP server. The collected data provides insights into the latency characteristics of standard network equipment and devices.
- *OAI 5G Measurement Points*: The Open Air Interface (OAI) 5G setup is based on the software 5G implementation of OAI. OAI offers a flexible and open-source platform for experimenting with 5G, allowing for a deeper analysis of packet delay and its variation. OAI 5G provides an opportunity to collect rich amounts of data by inserting measurement points in the different layers of the 5G protocol stack at UE, Radio Access Network (RAN), and/or core. A packet when traversing a layer (e.g., Radio Link Control (RLC)) is timestamped along with the local identifier (e.g., RLC sequence number) as well as network metadata (e.g., number of Protocol Data Units (PDUs) waiting in the RLC queue). The identifiers are later used to track the end-to-end journey of each packet through the OAI 5G system.
- *Data Collection, Aggregation and Storage*: Collecting data with minimal interference in the user plane traffic is essential, ensuring that the process does not impact the measurement, i.e., packet delay should not be significantly changed due to the introduction of measurement points. Therefore, we selected the LatSeq project as our primary tool for data collection. LatSeq is tailored to extract timestamped information across different layers within OAI.

Real-time aggregation of data from different measurement points in the 5G system and processing the aggregated data is important for the latency measurement framework. To meet this requirement, we follow a microservices architecture (as mentioned in the design) for developing our framework by moving away from the file-based data exchange approach used in LatSeq. The data-aggregator is packaged as a Docker container, which is responsible for gathering data from all measurement points over via socket-based connections.

As probabilistic analysis of packet delay is important for certain applications, it is useful to optimize data storage with respect to that objective. To this end, the latency measurement framework incorporates InfluxDB, a database specialized in time-series data, ensuring the efficient storage of measurements and swift response to queries. The aggregated data can optionally undergo some analytical processing to derive meaningful insights into the network's latency performance. We have implemented simple analytics to demonstrate packet delay decomposition at different delay targets.

It is worth mentioning that implementing and evaluating the latency measurement framework requires a flexible experimentation platform that can facilitate detailed end-to-end experiments. ExPECA [EXPU25] serves as an ideal testbed for wireless communication and edge-computing studies,

offering the ability to conduct experiments through the use of COTS 5G as well as Software-defined Radios (SDRs) and OAI 5G for enhanced reproducibility.

Next, we provide a description of how the latency measurement framework can be used to collect measurements on the COTS 5G system.

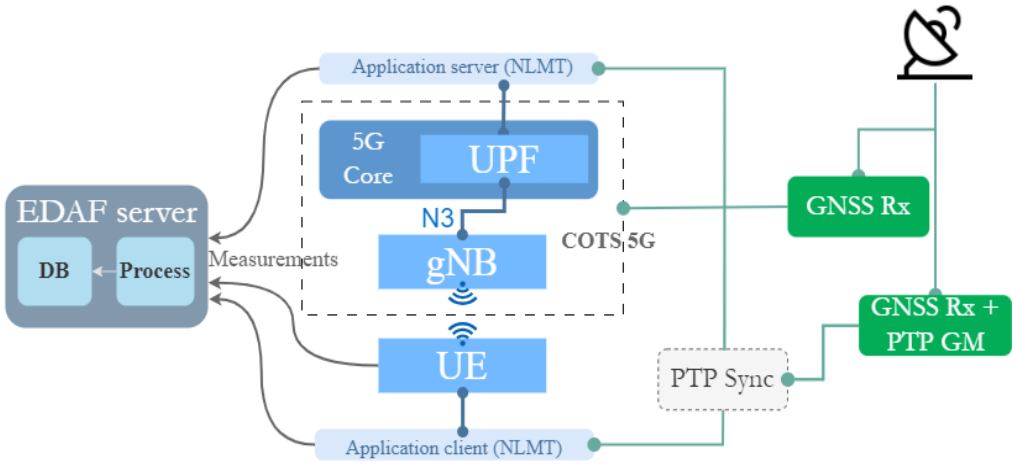


Figure 2-7: Measurement setup for data collection in COTS 5G setup.

The measurement setup for data collection on COTS 5G setup is shown in Figure 2-7. A key component of the latency measurement framework is the Network Latency Measurement Tool (NLMT). NLMT is based on an existing network measurement tool called IRTT (Isochronous Round-Trip Tester). This tool accomplishes two tasks: (i) traffic generation and (ii) packet timestamping. The UDP packets of a given payload size are generated periodically at a fixed (configurable) interval by the client NLMT instance and are sent towards the server NLMT instance. The packet sequence numbers and timestamps are recorded at both client and server. In addition to this information, network information (RSRP and RSRQ) is sampled at the COTS UE at a fixed interval using a measurement service continuously. This recorded information at client, server and the UE is transported to a remote server for storage and possible analytics.

The workflow for data collection on the COTS 5G system using the ExPECA testbed is shown in Figure 2-8. The procedure is documented in detail in the ExPECA user guide [EXPU25]. Here, we only discuss the high-level steps. Two hosts (baremetal server nodes) and the COTS 5G system are reserved in the ExPECA testbed. Two containers are instantiated on the two reserved server hosts, to host the NLMT client and server, respectively. The NLMT server listens for packets sent by the NLMT client at UDP port 2112. The measurement session is initiated by running a script at the first host that triggers the NLMT client to run one or more measurement rounds. Using the object storage service, the collected data (e.g., JavaScript Object Notation (JSON) files) can be stored in *containers* to access them after the measurements are finished.

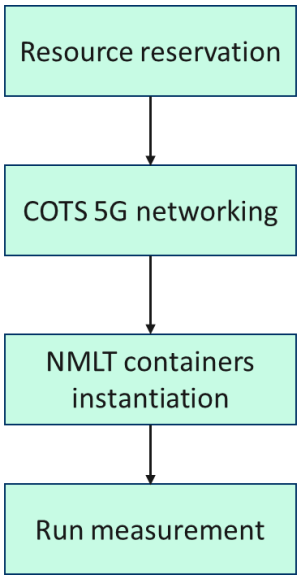


Figure 2-8: Procedure for latency measurements on COTS 5G setup.

The data collected on the existing COTS 5G provides limited information about the 5G system. As the COTS 5G does not offer much flexibility for such data collection, we have enhanced the framework for data collection in software-based 5G system implementation, e.g., OpenAirInterface 5G. OAI 5G is an open-source implementation of 5G that in conjunction with SDRs offers a flexible platform for 5G experimentation.

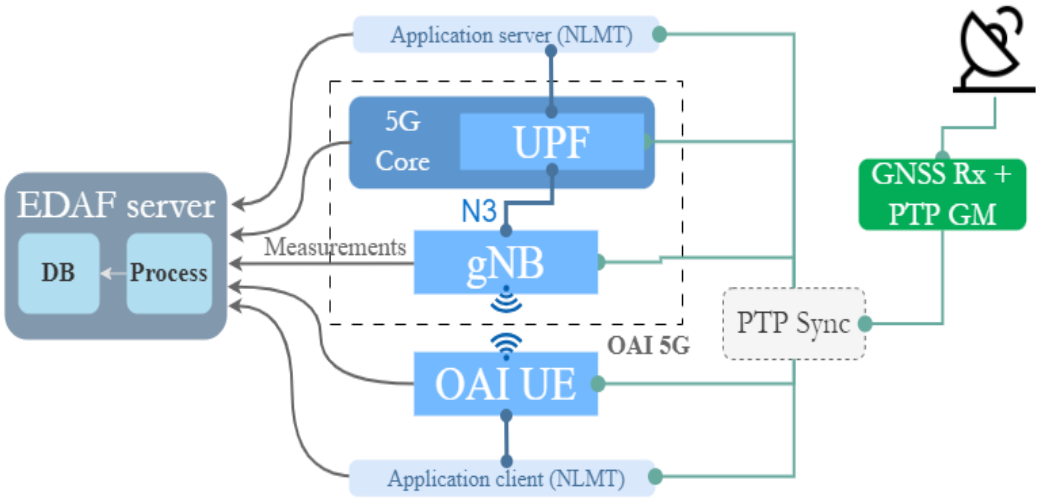


Figure 2-9: Measurement setup for data collection in OAI 5G setup.

The measurement setup for data collection on OAI 5G is shown in Figure 2-9. This setup is almost similar to the COTS setup, except that the 5G domain is composed of the more flexible OAI network stacks on the UE and infrastructure side

Figure 2-10 shows the high-level procedure to perform latency measurements on the OAI 5G setup on the ExPECA testbed. First, the required resources (e.g., three hosts, two USRP SDRs) are reserved. Next, the OAI 5G core network services are set up by instantiating corresponding docker containers on one host. The End-to-End Data Analytics Framework (EDAF) service, which is responsible for data aggregation and processing, is then instantiated on the core network host. Next, containers for UE

and gNB are instantiated on the remaining two hosts. Finally, the measurement session(s) are initiated by instantiating the NLMT containers whose results are stored online through the EDAF service in a data base.

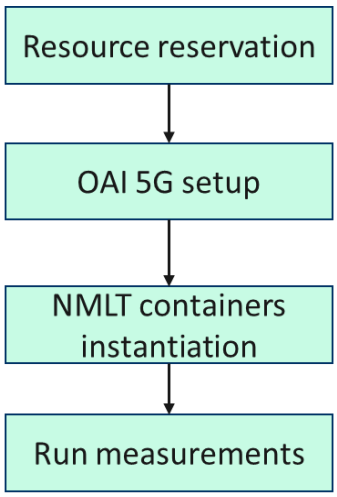


Figure 2-10: Procedure for latency measurements on OAI 5G setup.

2.4 Security Emulation Framework

2.4.1 Introduction

In D1.2, "First report on DETERMINISTIC6G architecture", [D6G-D1.2], we provided a comprehensive overview of potential threats to the deterministic properties of networks and presented a range of remediation strategies to prevent, detect, and mitigate those threats. In D2.2, "First Report on the time synchronization for E2E time awareness" [D6G-D2.2], we focused specifically on how these threats affect time synchronization protocols, summarizing the threat model and its impact on time-aware networks.

While cryptographic solutions, such as end-to-end encryption, are effective in protecting against many threats like message tampering, spoofing, packet replay, and communication disruption, they fall short when it comes to delay-based attacks. These include Denial-of-Service (DoS) attacks or insider threats where the attacker has access to a trusted part of the network or possesses encryption or authentication keys.

A notable example is the Time-Delay Attack (TDA). Unlike conventional cyberattacks that alter packet contents, a TDA works by delaying a Precision Time Protocol (PTP) packet without modifying it. An attacker may intercept a synchronization message, hold it for a brief period, and then forward it to its destination. This intentional delay skews propagation delay measurements, causing synchronization errors across downstream clocks.

Detecting TDAs is especially challenging because their behavior closely mimics normal network latency, leaving no obvious signs like those found in traditional attacks. By subtly manipulating packet delivery timing, these attacks can bypass standard cryptographic defenses. This highlights the need for continuous monitoring of timing patterns to identify deviations before they impact system integrity.

To effectively counter such threats, detection and prevention mechanisms must be able to identify timing anomalies quickly. Intrusion Detection Systems (IDS) tailored for deterministic networks should

not only detect malicious activity but also monitor the timing and sequencing of packet arrivals to spot irregularities. In these environments, where precise timing is critical, anomaly detection must extend beyond payload inspection to include behavioral analysis of time-sensitive flows.

To address this gap, a security-by-design approach was introduced in D3.2, "Report on the Security Solutions" [D6G-D3.2]. This approach integrates high-precision telemetry with a programmable data plane to enhance security management in deterministic networks, providing the visibility and control needed to detect and respond to subtle timing-based attacks. To ensure a self-contained explanation, we provide a brief summary of this approach below.

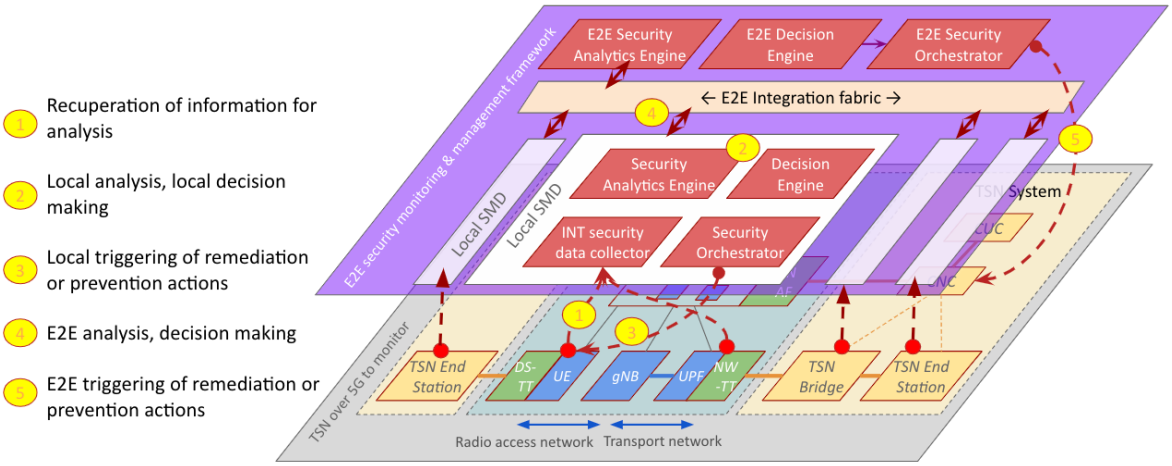


Figure 2-11: High level architecture of E2E security monitoring & management framework.

Figure 2-11 illustrates a deployment architecture for security and enforcement services, incorporating High-Level Architecture (HLA) functional blocks and services. This architecture is designed not only to enhance security but also to foster trustworthiness and accountability in the management of 5G/6G network infrastructures across multiple domains. Within the DETERMINISTIC6G project, the architecture is being extended to more comprehensively address the requirements of deterministic networking. This enhancement includes the integration of high-precision monitoring techniques and the support for both Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet) domains. The goal is to establish a robust, efficient framework capable of meeting the stringent demands for precision, reliability, and consistency that are fundamental to deterministic communication environments.

The architecture is organized into multiple Security Management Domains (SMDs), each designed to ensure system resilience and to separate concerns across different parts of the network, such as the Radio Access Network (RAN), Edge, and Core Network. Each SMD is responsible for intelligent, AI-driven security automation within its domain and includes a set of functional modules, including the Security Data Collector, Security Analytics Engine, Decision Engine, Security Orchestration, and others. These modules collectively deliver a wide range of security services, accessible both within their respective domains and across domains via an integration fabric.

A specialized domain, known as the End-to-End (E2E) SMD, is dedicated to managing the security of E2E services. By decoupling E2E security management from domain-specific SMDs, the architecture avoids monolithic designs, reduces complexity, and supports independent evolution of security mechanisms both at the domain and cross-domain levels.

All functional modules operate in an intelligent closed-loop cycle, consisting of data collection, threat detection, and response (as outlined in Steps 1 to 5 of Figure 2-11). This enables AI-driven, software-defined security (SD-SEC) orchestration and management, aligned with Security Service Level Agreements (SSLAs) and regulatory compliance requirements.

2.4.2 Security Emulation Framework

This section presents an implementation of the security-by-design approach proposed earlier, aimed at securing End-to-End (E2E) time synchronization using PTP. Further technical details of this implementation are provided in D2.4 "Report on the time synchronization for E2E time awareness" [D6G-D2.4].

Specifically, we emulate a PTP time synchronization network using Mininet, as illustrated in Figure 2-12, where each end-host functions as an Ordinary Clock (OC) and the switch operates as a Transparent Clock (TC). The OC is implemented using LinuxPTP, while the TC is developed with the P4 language (Programming Protocol-Independent Packet Processors), enabling custom data-plane logic tailored to timing security. By leveraging programmable data planes, the framework utilizes In-band Network Telemetry (INT) to embed monitoring data directly within PTP extension fields. This approach eliminates the need for additional probe packets and significantly reduces network traffic overhead. An INT collector module is responsible for capturing PTP packets, extracting embedded telemetry data, and enabling real-time detection and localization of TDAs.

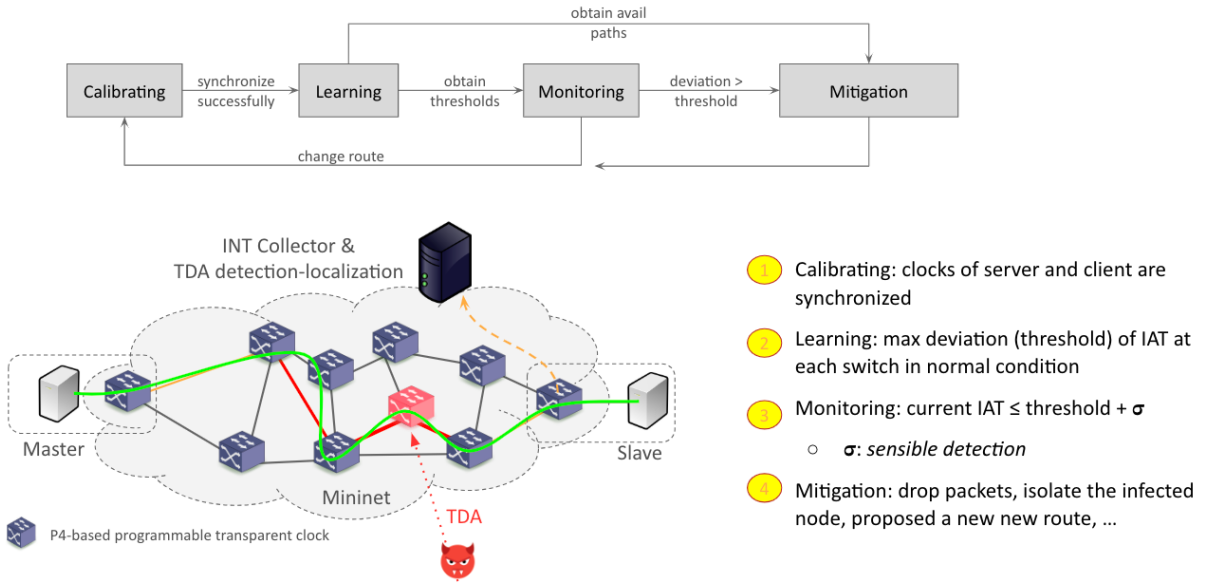


Figure 2-12: Mininet-based emulator of TDA & detection in E2E time synchronization.

It is worth noting that when PTP messages traverse a 5G system, the system effectively functions as logical TC. As can be seen in Figure 2-13, the 5G system consists of a core network with a User Plane Function (UPF), and a radio access network including base stations (gNBs) and User Equipments (UEs). The device-side TSN translator (DS-TT), appended to the UE, and the network-side TSN translator (NW-TT), appended to the UPF, are the entities in charge of handling the PTP messages in the 5G data plane. Calculating the 5G residence time and inserting it in a PTP message is part of the DS-TT's and NW-TT's responsibilities.

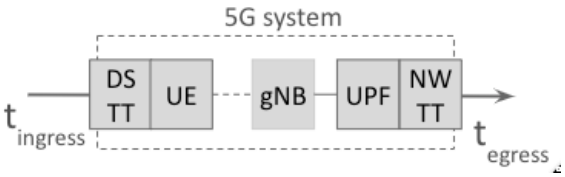


Figure 2-13: Time synchronization over a 5G system acting as a logical PTP transparent clock.

The TDA detection in our framework is carried out in four key phases: calibration, learning, monitoring, and mitigation, as described below:

- *Calibration Phase*: This phase occurs when the slave initially begins synchronizing with the master. During this period, the clock offset values are relatively large and exhibit significant variation. Over time, these offsets gradually decrease and stabilize. The calibration phase ends once the client achieves synchronization with the server.
- *Learning Phase*: Executed after successful synchronization, this phase assumes that the time synchronization network is operating under normal conditions, free of TDA interference. Its goal is to determine the maximum clock offset tolerated by the system, i.e., the accuracy the network can maintain. This accuracy must meet the predefined requirements. Additionally, the learning phase establishes a threshold representing the maximum acceptable variation in inter-arrival times (IAT).
- *Monitoring Phase*: In this phase, the framework continuously monitors the time synchronization network to detect and localized TDAs. It calculates updated values of IAT variation. An alert is triggered if (i) it is greater than the threshold determined in the learning phase above and (ii) it can cause a clock offset bigger than the given requirement.
- *Mitigation Phase*: This phase is planned as future work, as it may require interaction with a Centralized Network Controller (CNC). Potential mitigation strategies include isolating the malicious segment of the network and proposing an alternate path that maintains equivalent end-to-end delay while reusing as much of the original path as possible.

This emulation framework marks a step forward in integrating time synchronization security into software-defined and programmable networking architectures. The validation results of this approach are presented in the following section.

3 Validation Results

This section describes the main contribution of this report: the validation results for the concepts, mechanisms, and use cases developed by the DETERMINISTIC6G project.

3.1 Wireless-Friendly Scheduling and Packet Delay Correction Validation

Despite the early progress with standardization, it is becoming apparent that the initially standardized systems, and particularly the TSN integration into 5G systems, will not suffice for the convergence and scale anticipated for the future cyber-physical continuum. The design principle of TSN mandates Ethernet time-division multiplexing and router queue management based on tightly synchronized network nodes to facilitate end-to-end latency guarantees. The major invariant of TSN (and in principle also of DetNet) is to achieve these guarantees by assuming individual delay components to be small and deterministic. This relates to transfer times over Ethernet segments or within an Ethernet bridge, which both have granularities of a few microseconds with only marginal variations. This design philosophy is not compatible with typical latencies ranges and variations observed in state-of-the-art

wireless systems. Even if considering 5G Ultra-Reliable Low Latency Communication (URLLC) systems, wireless latency variations are in the range of hundreds of microseconds. Deployed commercial 5G systems today, i.e., non-URLLC systems, have typical latencies in the range of milliseconds and significant stochastic variations due to retransmission recovering occasional random losses. However, the current 5G integration into TSN/DetNet neglects these mismatches, leading essentially to hugely underutilized wireless systems, while still jeopardizing the end-to-end latency guarantees of high-criticality streams.

In this section, we start with an overview of novel enablers that we presented in previous deliverables to address the above challenges. Our key insight is that while wireless-friendly TSN solutions will allow for more efficient integration, wireless systems will have to become more predictable (but not necessarily more deterministic) to fully utilize our solutions.

3.1.1 Background

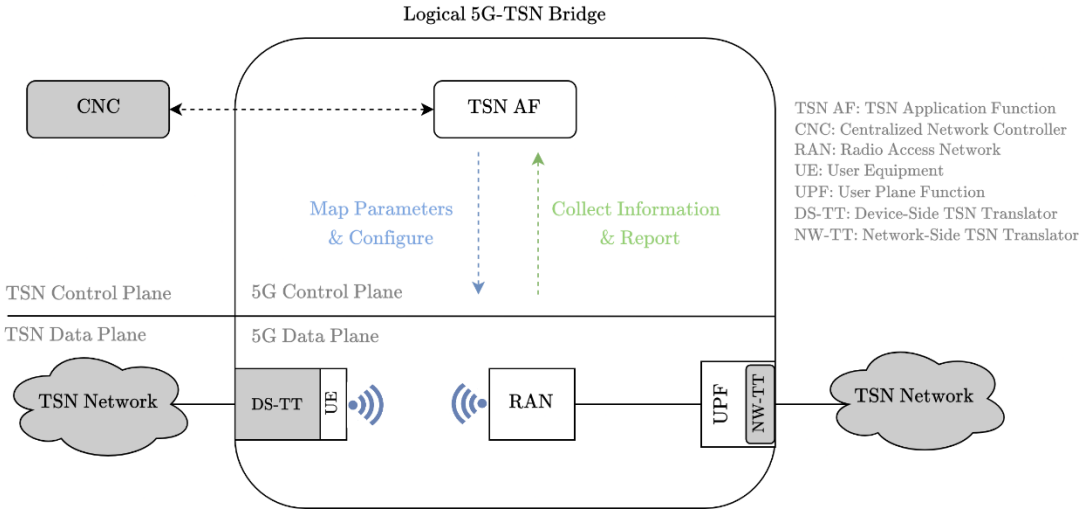


Figure 3-1: 5G logical TSN bridge as modeled in 3GPP.

IEEE 802.1 TSN and IETF DetNet are networking technologies that can provide E2E dependable time-critical capabilities at Layer 2 and Layer 3, respectively. TSN (and DetNet) rely on certain traffic shaping mechanisms to achieve these dependable time-critical capabilities. For instance, stringent deterministic guarantees on delay and Packet Delay Variation (PDV) are typically provided by employing the IEEE 802.1Qbv Time-Aware Shaper [IEEE15-8021Qbv], which works by scheduling time-critical traffic into time slots, based on a precalculated schedule, at the egress port of a TSN bridge. These calculations are typically performed within a centralized network controller (CNC), which has a global view onto the TSN network and is responsible for configuring the TSN features in both bridges and end-stations.

Wireless communication such as 5G comes into the picture due to its ease to scale and deploy, as well as its support for mobility that offers great advantages. Support for TSN and DetNet has been standardized in 3GPP through some extensions needed to ensure deterministic functionalities [3GPP-TS23501]. The most important integration decision relates to the way 5G systems are mapped to TSN/DetNet functional elements: A 5G system – comprising in the user plane the user equipment (UEs), radio access network (RAN), and core network user plane function (CN UPF) – is modeled as a logical TSN/DetNet node by means of defining translation entities both in the data plane and in the

control plane. In other words, a 5G system is integrated as a logical Ethernet bridge into TSN/DetNet. As this modeling abstraction is not straightforward, two data plane entities had to be incorporated: the device-side TSN translator (DS-TT), co-located with the device or user equipment (UE), and the network-side TSN translator (NW-TT), co-located at the data-network side at the User-Plane Function (UPF).[3] As shown in Figure 3-1, these two entities implement the interface or port with the external network or next hop. In the control plane, a translation entity interacts with the CNC (for TSN), namely the TSN Application Function (TSN-AF), while for interfacing the DetNet controller, a different network function is used, namely the Time-Sensitive Communication and Time Synchronization Function (TSCTSF).

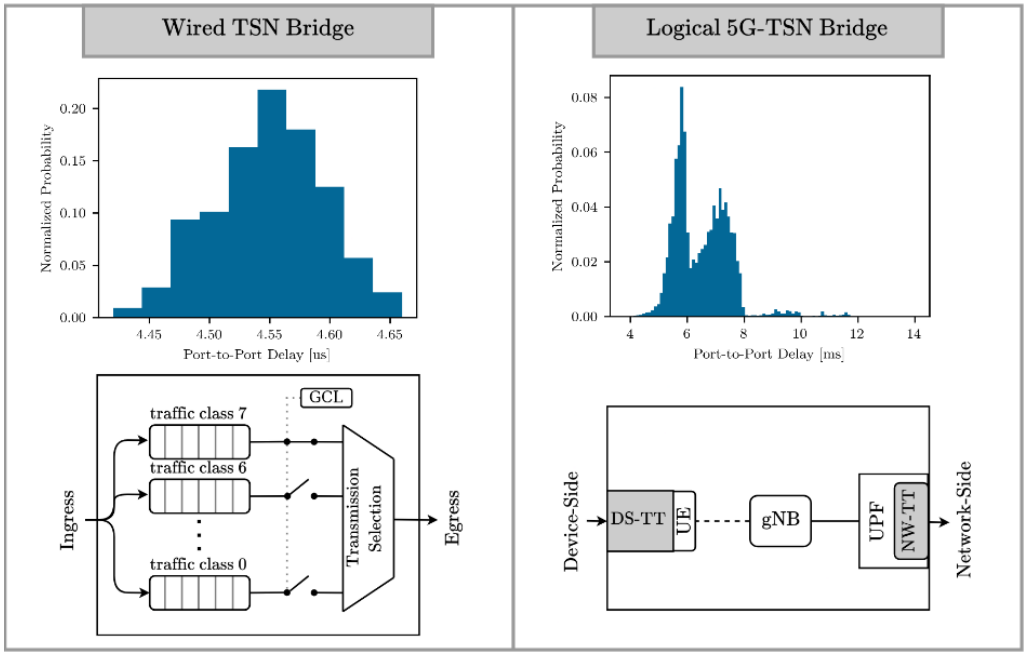


Figure 3-2: Port-to-port delay of wired TSN bridges vs. wireless 5G-TSN bridges.

Compared to deterministic system models that are commonly employed for wired TSN [SOL+23], logical 5G-TSN bridges introduce significant variations in the experienced packet delays. Figure 3-2 compares measurements of the port-to-port delay from a TSN bridge (i.e., excluding queueing delay) and the corresponding "port-to-port" delay of a 5G network deployment [HDM+23]. While the TSN bridge can achieve a port-to-port delay of less than 5 microseconds with a packet delay variation (PDV) in the range of 100's of nanoseconds, the port-to-port delay and PDV of the wireless logical bridge are in the range of milliseconds. A more exhaustive latency analysis for 5G is available in [D6G-D2.1]. Cleary, this difference makes an efficient end-to-end integration of 5G and TSN challenging if both wired and wireless network elements are present.

To exemplify the effects of port-to-port delays on common TSN traffic scheduling mechanisms, we investigate the behavior of the IEEE 802.1Qbv Time-Aware Shaper in this work as its delay guarantees are directly influenced by port-to-port delays. The Time-Aware Shaper can govern the exact transmission times of frames in each traffic class by introducing a Gate Control List (GCL) to specify the opening and closing time of gates at each egress queue. Hence, the central challenge of providing

[3] In the case of DetNet, only a NW-TT is required.

end-to-end QoS guarantees for each TSN stream is to compute suitable GCLs that are robust against stochastic port-to-port delays as introduced by the logical 5G-TSN bridges.

Packet Delay Correction

The underlying idea of PDC [D6G-D2.1] is to shrink the uncertainty interval by holding back early packets until a predefined minimum residence time in the wireless bridge has passed, before enqueuing them into the egress queue (see Figure 3-3).[4] This effectively folds the left side of the port-to-port-delay distribution to the minimum residence time value. If the minimum residence time is equal to or greater than the maximum value of the original distribution, this leads to a sharp delay peak, i.e., uncertainty is (ideally) eliminated completely. Obviously, artificially holding back frames increases the average delay. So PDC trades off reduced uncertainty (i.e., small PDV) for longer average delay. For the large part of time-critical applications the average latency does not play a role, but rather the certainty that all messages arrive before their maximum delay bound.

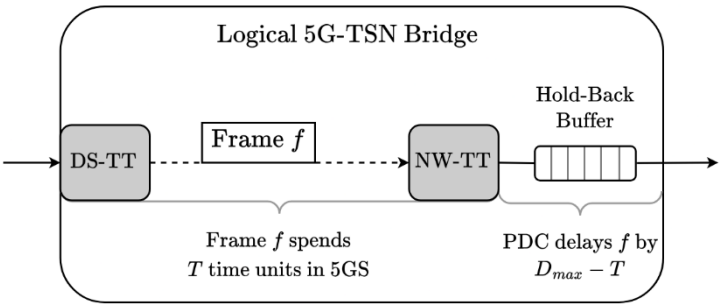


Figure 3-3: Packet delay correction mechanism.

One way to implement PDC is to timestamp packets at the ingress and implement a hold-back buffer before the egress queue whose release time is controlled by another timer. More specialized approaches are described in [D6G-D2.1]. Both the ingress timer and the release timer need to be synchronized, which can be realized using clock synchronization in the 5G system [3GPP-TS23501].

By shrinking the uncertainty interval to a smaller size, in the range of microseconds rather than milliseconds, existing wireless-unaware schedule planning algorithms can be applied without undesired frame re-ordering. However, the degree of delay correction influences the uncertainty interval size and, therefore, the efficiency of the schedule.

Wireless-Friendly Scheduling

As an alternative to the usage of dedicated hold-back buffers, a wireless-friendly scheduler can configure the IEEE 802.1Qbv schedule to compensate for a bounded packet delay variation $[d_{min}, d_{max}]$, as perceived in a logical 5G-TSN bridge[5]. This approach requires that the scheduler (which typically resides within the CNC) is provided with some statistical knowledge about the 5G packet delays, which can be provided for instance in the form of an empirical histogram. A robust IEEE 802.1Qbv schedule must provide formal end-to-end QoS guarantees for frames whose 5G delays are bounded by $[d_{min}, d_{max}]$. Importantly, these guarantees are made for each individual TSN stream, i.e., transmission faults of one stream must not impair the QoS guarantees of other streams.

[4] Previous discussions about standardizing such functionality in the “hold and forward buffering mechanisms” were held in 3GPP WG SA2 (e.g., 3GPP S2-2002056), but were eventually not standardized.
[5] As shown later in this section, im combination PDC and a wireless-friendly scheduler clearly outperforms the benefits that each of the two approaches provide individually.

Moreover, leaving the wireless-friendly scheduler to influence the 5G packet delay budgets $[d_{min}, d_{max}]$ has two main advantages:

1. Instead of having to compensate for the maximum PDV, e.g., $[4\ ms, 14\ ms]$ for the uplink delays, the scheduler can choose a sufficiently large packet delay budget to satisfy the stream's reliability requirement. For instance, a budget of $[4\ ms, 10\ ms]$ suffices for streams that solely require a $99\ \%$ reliability guarantee.
2. In case of a significant disruption in the 5G channel conditions, a wireless-friendly scheduler can realize graceful degradation in the streams' latency and reliability guarantees, instead of having to drop individual streams entirely. For example, reducing the 5G packet delay budgets to $[4\ ms, 9\ ms]$ may already be enough to find an eligible schedule, while the reliability guarantees are only reduced to $97\ \%$.

Striving towards this goal, it is also necessary to address the problem of low schedulability with conventional IEEE 802.1Qbv scheduling approaches. We therefore introduce the concept of controlled frame interleaving. Intuitively, controlled frame interleaving moves away from the conventional approach to exclusively reserve transmission slots for individual frames. Instead, it enables wireless streams to share the same transmission slot by creating frame batches. While allowing frame re-ordering within the same batch to improve link utilization, streams are only allowed to interleave if the streams' latency guarantees can be satisfied (i.e., each frame arrives before its deadline even when being the last one in its batch). For further details, we refer the interested reader to [EGS+25].

3.1.2 Methodology and Setup

Next, we evaluate both presented methods, PDC and Wireless-Friendly Scheduling, and show that they significantly increase the efficiency of end-to-end scheduling in networks with both wired and wireless network entities. We quantify the efficiency of the considered methods in the number of wireless streams that can be scheduled. To this end, we consider the scenario shown in Figure 3-4, where an automated guided vehicle (AGV) is connected to the backbone TSN network of a factory. The logical 5G-TSN bridge thus connects two wired TSN network segments: the AGV-internal network (left segment) and the wired TSN backbone (right segment). All Ethernet links within the wired segments have a conservative data rate of $100\ Mbps$ and a propagation delay of $50\ ns$ (we note that the results stay quantitatively the same for different Ethernet settings). For packets traversing the logical 5G-TSN bridge, we consider the uplink/downlink 5G packet delay histograms that were obtained in a real 5G system [HDM+23]. Moreover, we verify the correct operation of the obtained IEEE 802.1Qbv schedules with our 6GDetCom [D6G-D4.1] extension for the OMNeT++/INET framework.

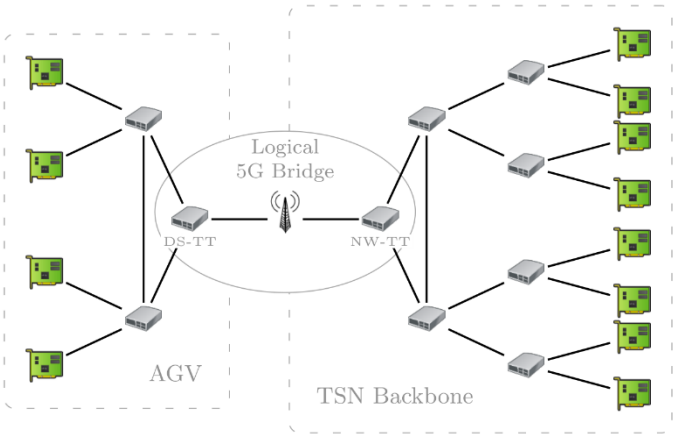


Figure 3-4: Network topology used for the simulation

We consider both wired and wireless streams: Wired streams stay within their respective network segments and have stringent latency requirements of $500\ \mu s$. In contrast, wireless streams traverse the logical 5G-TSN bridge (either in uplink or downlink direction) and have more relaxed latency requirements of $20\ ms - 25\ ms$. The latency relaxation is selected as the utilized 5G histograms already have a maximum packet delay of $14\ ms$ (uplink) and $17\ ms$ (downlink). Throughout our experiments, we include a fixed number of wired streams (five per wired segment) and try to maximize the number of wireless streams that can be scheduled without violating any stream's QoS requirements. The routes of both wired and wireless streams are randomly chosen. Moreover, we solely consider a single Qbv-governed queue per egress port to amplify the bottleneck of compensating 5G packet delay variations. As time-triggered streams typically have the highest traffic priority, we do not consider any other traffic classes like video, audio, or background traffic.

3.1.3 The Need for Wireless-Aware Traffic Engineering

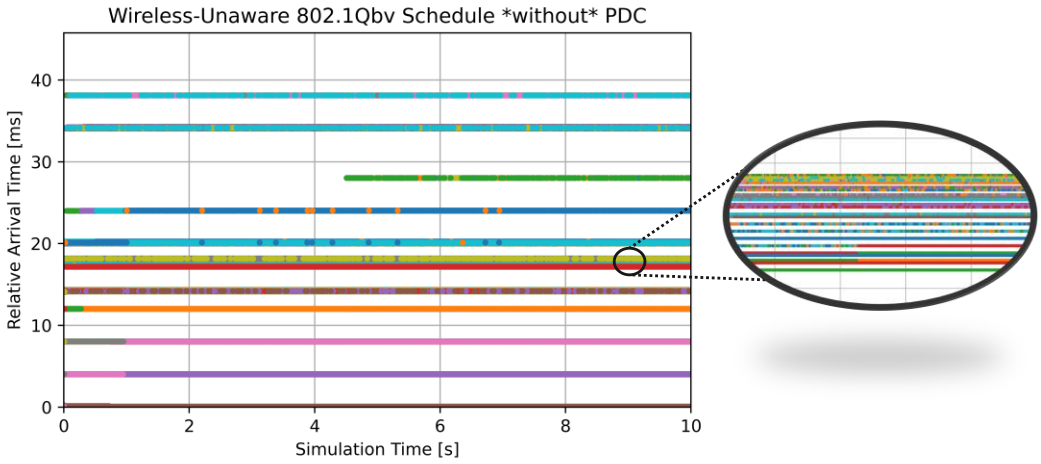


Figure 3-5: Relative arrival time for wireless-unaware Qbv schedule and without PDC

We start by illustrating the need for wireless-aware traffic engineering. To this end, we start by evaluating conventional (i.e., wireless-unaware) scheduling approaches for wired TSN that would simply consider the maximum 5G packet delays to compute the TSN schedule. Contrary to the intended outcome, however, Figure 3-5 shows that the resulting latency of multiple streams (each

stream is shown in a different color) can lie far beyond the acceptable deadline of $20\ ms$ in this setting. This is caused by the fact that solely considering the worst-case delay does not sufficiently account for the 5G PDV. In detail, it does not constrain two frames f_1 and f_2 to arrive within a fixed order at the NW-TT/DS-TTs. Instead, the two frames can be enqueued in the order $f_1 \prec f_2$ or $f_2 \prec f_1$. As a result of this unintended frame reordering within the egress queues, frames can miss their transmission slots and can even be delayed beyond their deadline. Moreover, these effects do not necessarily stay isolated to individual streams but may spread throughout the entire network and nullify any QoS guarantees as a result.

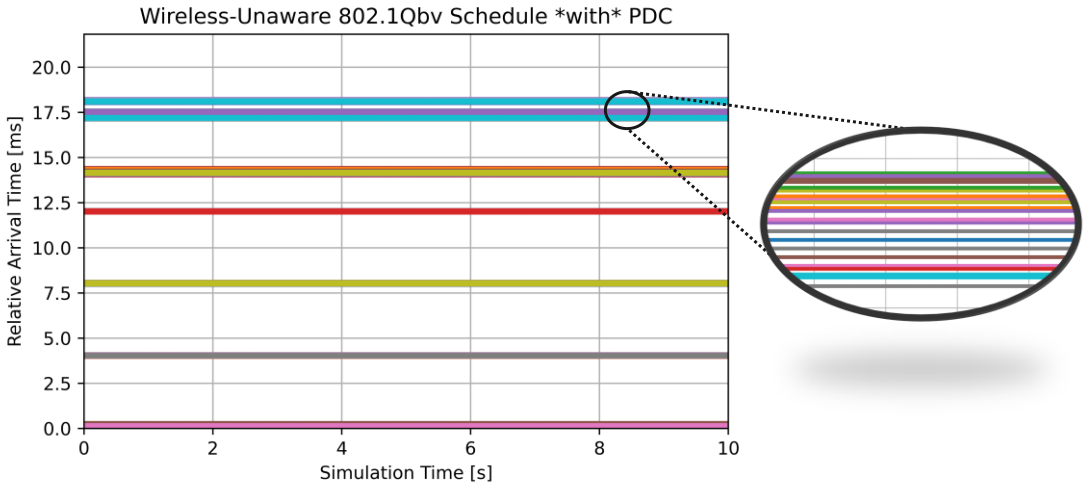


Figure 3-6: Relative arrival time for wireless-unaware Qbv schedule and with PDC

These issues can be resolved by employing either PDC or wireless-friendly scheduling. Figure 3-6 demonstrates this claim by deploying PDC in the previous scenario. It can be seen clearly that there are no more frame reordering effects and that each frame is delivered at its listener before reaching the deadline of $20\ ms$. The main reason behind this is that PDC effectively eliminates any packet delay variation of the 5G system by de-jittering the frames at the egress side (i.e., the NW-TT/DS-TT for uplink/downlink streams) until the maximum 5G delay d_{max}. Conventional IEEE 802.1Qbv schedulers can therefore safely use this value to synthesize their Gate Control Lists (GCLs). Although wireless-friendly scheduling yields qualitatively the same results as in Figure 3-6, it is worth highlighting again that the reasons are entirely different: That is, our wireless-friendly scheduler does not rely on dedicated per-stream hold-back buffers, but allocates a sufficient de-jittering time interval at the respective egress queues (again, the NW-TT/DS-TT for uplink/downlink streams). To provide a scalable solution, our wireless-friendly scheduler also allows frame batching to reduce the impact of blocking an egress queue for the purpose of de-jittering.

3.1.4 Results on Combining Wireless-Friendly Scheduling and Packet Delay Correction

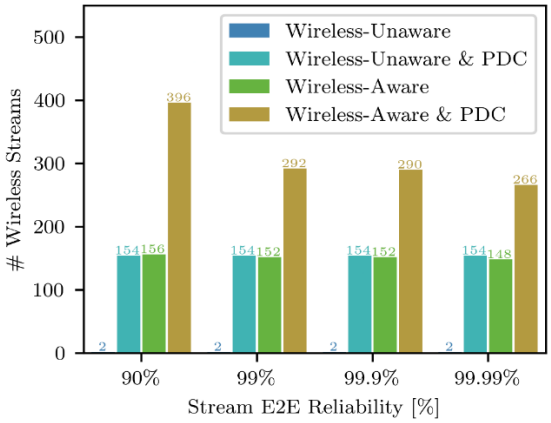


Figure 3-7: Scalability results

Figure 3-7 shows the maximum number of wireless streams for which a feasible IEEE 802.1Qbv schedule is found within 5 minutes of scheduling. The results are shown in dependence of different per-stream reliability requirements (i.e., for each wireless stream, the percentage of frames that satisfy their respective QoS requirements) that range from 90 % to 99.99 %. First and foremost, the results show the poor scalability of conventional scheduling approaches: Their exclusive slot allocation effectively reserves the egress queue at the NW-TT (DS-TT) for the entirety of the uplink (downlink) min-max 5G packet delay bounds, which only leaves room for two wireless streams. This bottleneck is reduced by both of our approaches: PDC utilizes dedicated per-stream hold-back buffers that eliminate the need to reserve an egress at the NW-TT (DS-TT). Similarly, wireless-friendly scheduling does not rely on exclusive slot reservation and instead allows for controlled interleaving (or batching) of frames that still upholds the QoS guarantees of each stream. Both approaches, on their own, can already improve scalability by a factor of more than x74. However, the results also show that the best performance is achieved when allowing the wireless-friendly scheduler to have control over PDC. Indeed, the scheduler thereby does not necessarily have to use PDC for 100 % compensation of the 5G packet delay variations but can fine-tune the compensation for the specific reliability requirement and cut the remaining long-tail. While this impact fades for ultra-reliability requirements like 99.99 % (reducing d_{max} only by 0.7 ms), we find that the batching capability of wireless-friendly scheduling allows the scheduler to find a feasible solution faster (before the 5-minute timeout). The results thereby demonstrate that the complementary design of PDC and wireless-friendly scheduling also enables the combination of their advantages.

3.1.5 Key Takeaways

We have shown for a wireless 5G system integrated into a TSN network, how the large PDV of 5G limits significantly the performance and scalability of the TSN network when conventional wired approaches to TSN network configuration and traffic management are applied. We have in particular studied the TSN 802.1Qbv mechanism where traffic is time-scheduled through the network. Nevertheless, through novel TSN approaches the performance and scalability can be significantly improved. We have presented two such approaches: one where uncertainties of the wireless system in terms of packet delay variation are removed in the 5G system via Packet Delay Correction. This transforms the 5G system into a system with quasi-deterministic latency characteristics, and it improves the applicability of conventional TSN scheduling mechanisms. A more advanced approach is

to enhance the end-to-end scheduling mechanism of TSN such that the packet delay variations of the 5G system are explicitly considered in the TSN scheduling scheme. To this end, it is required to be able to characterize the wireless system delay, for which we have proposed a data-driven approach. While our solutions are very promising, we acknowledge that many other approaches for optimizing TSN and DetNet communication with wireless 5G networks are possible. As an in-depth analysis of TSN networks and wireless network characteristics are essential for future studies, we have extended the open-source TSN simulator INET with extensions for wireless systems accordingly [D6G-D4.1] and contributed an open-source wireless-friendly scheduler [EGS+25].

3.2 Measurements and Characterization of RAN Latencies

3.2.1 Introduction

We focus on a series of measurements demonstrating different effects in 5G systems around segmentation and end-to-end latency, and how different latency components within the network stack contribute to the overall system behavior. We leverage here extensive project results regrading principal delay decomposition as outlined in D2.1 [D6G-D2.1] and more extensive delay measurement campaigns as presented in D4.3 [D6G-D4.3].

3.2.2 Validation Methodology

We leverage the latency measurement tool EDAF discussed in Section 2.3. Specifically, we focus on the OAI-based implementation, which allows for more fine-grained resolution into latency effects in the system. We run three different experiments:

1. Experiment (a) serves as the baseline experimentation for the OAI 5G packet delay measurements.
2. In Experiment (b) we attempt to eliminate the segmentation delay by increasing the uplink grant PRBs from 5 to 10, which allows for a transport block size (TBS) of 880 bytes.
3. In Experiment (c), not only segmentation delay is eliminated, but also the frame-alignment delay is minimized by reducing queuing delay.

3.2.3 Validation Results

The measurement results and the decomposition analytics are presented in Figure 3-8 and Figure 3-9. Each subfigure (a, b, and c) in Figure 3-8 and Figure 3-9 corresponds to Experiments (a), (b), and (c), respectively. Experiment (a) serves as the baseline experimentation for the OAI 5G packet delay measurements. Figure 3-8 shows, on the right y-axis, the Complementary Cumulative Distribution Function (CCDF) for the measured end-to-end packet delays. The CCDF can be used to derive the Delay Violation Probability (DVP) for various end-to-end delay targets. Furthermore, the figure's left y-axis breaks down the proportional contributions of different components to the total end-to-end packet delay. Initially, we evaluate the DVP for a specific threshold. For Experiment (a), achieving a 15 ms target results in a DVP of 10^{-2}, whereas for a 5 ms target, the DVP nearly reaches 1. We then analyze the breakdown of delay contributions for each target. In both scenarios, the segmentation delay is the main contributor, accounting for 45 % and 40 %, respectively.

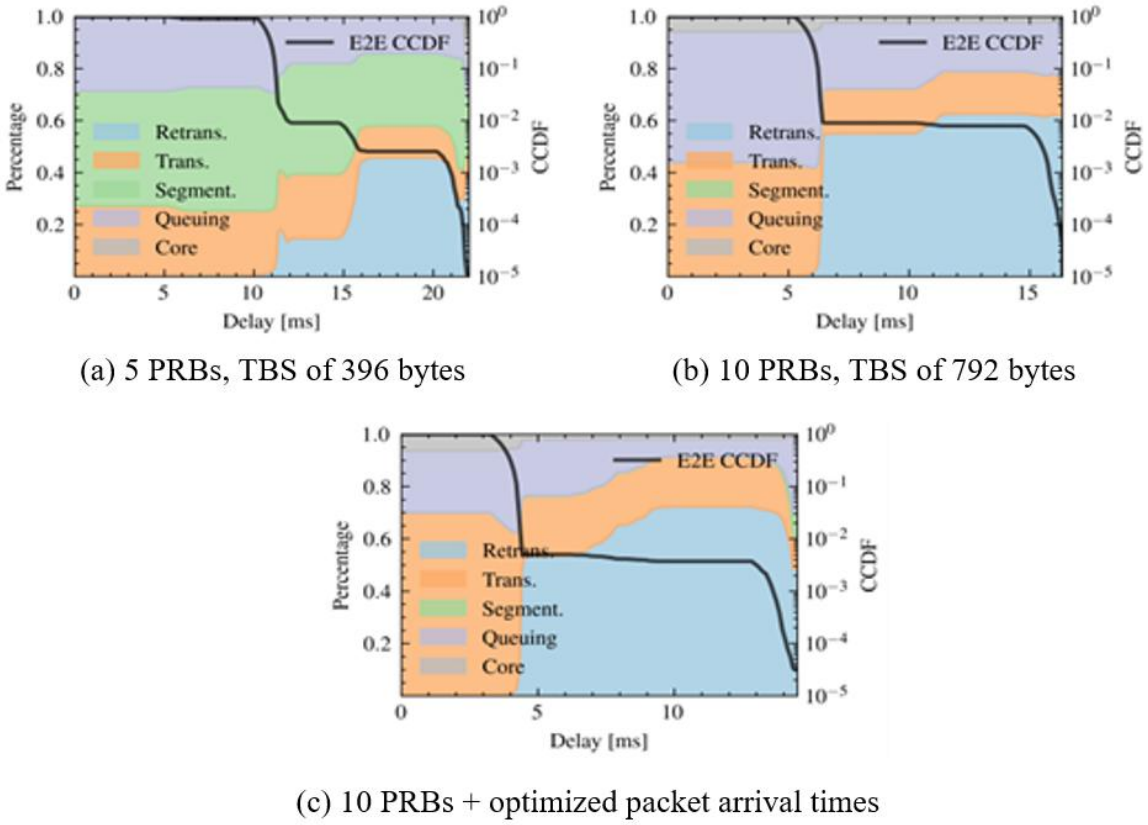


Figure 3-8: End-to-End CCDF and decomposition in experiments feature a fixed MCS index of 23 and 500-byte packets.

In Experiment (b), we attempt to eliminate the segmentation delay by increasing the uplink grant PRBs from 5 to 10, which allows for a TBS of up to 792 bytes. This size can accommodate a full 531-byte packet along with its headers, in contrast to the TBS of 396 bytes in Experiment (a). The smaller TBS in Experiment (a) resulted in a packet segmentation, thus, adding a 5 ms increase in the end-to-end delay. Figure 3-9 depicts the distribution of radio arrival times, service times, and departure times, plotted over three consecutive frames to illustrate the packets' journey over time. Specifically, Figure 3-9 (a) highlights the delay caused by segmentation, with the second segment's service times lagging 10 or 20 slots behind the first, thereby delaying packet departure. This issue is addressed in Figure 3-9 (b) and Figure 3-9 (c), leading to significantly improved end-to-end delays. Nonetheless, Experiment (b) maintains the same DVP for both 15 ms and 5 ms delay targets, requiring further optimization.

In Experiment (c), not only segmentation delay is eliminated, but the frame-alignment delay is minimized by reducing queuing delay. The observed time gap in Figure 3-9, ranging from the packets' arrival in slots 7 to 10 to their first service in slots 12 to 14, results from packets arriving too early. By optimizing the arrival time offset, to minimize queuing delays, a reduction in end-to-end delays by 3 ms is expected.

The search for the optimal arrival, as shown in Figure 3-10, confirms our hypothesis, with frame arrival offsets between 1-2 and 6-7 achieving the lowest end-to-end delays. Here, we again measure the UL latency between a UE configured with 10 PRBs. As a result, this experiment limits DVPs for both 5 ms and 15 ms targets to 10^{-2} and 10^{-4}, respectively, fulfilling the application requirements.

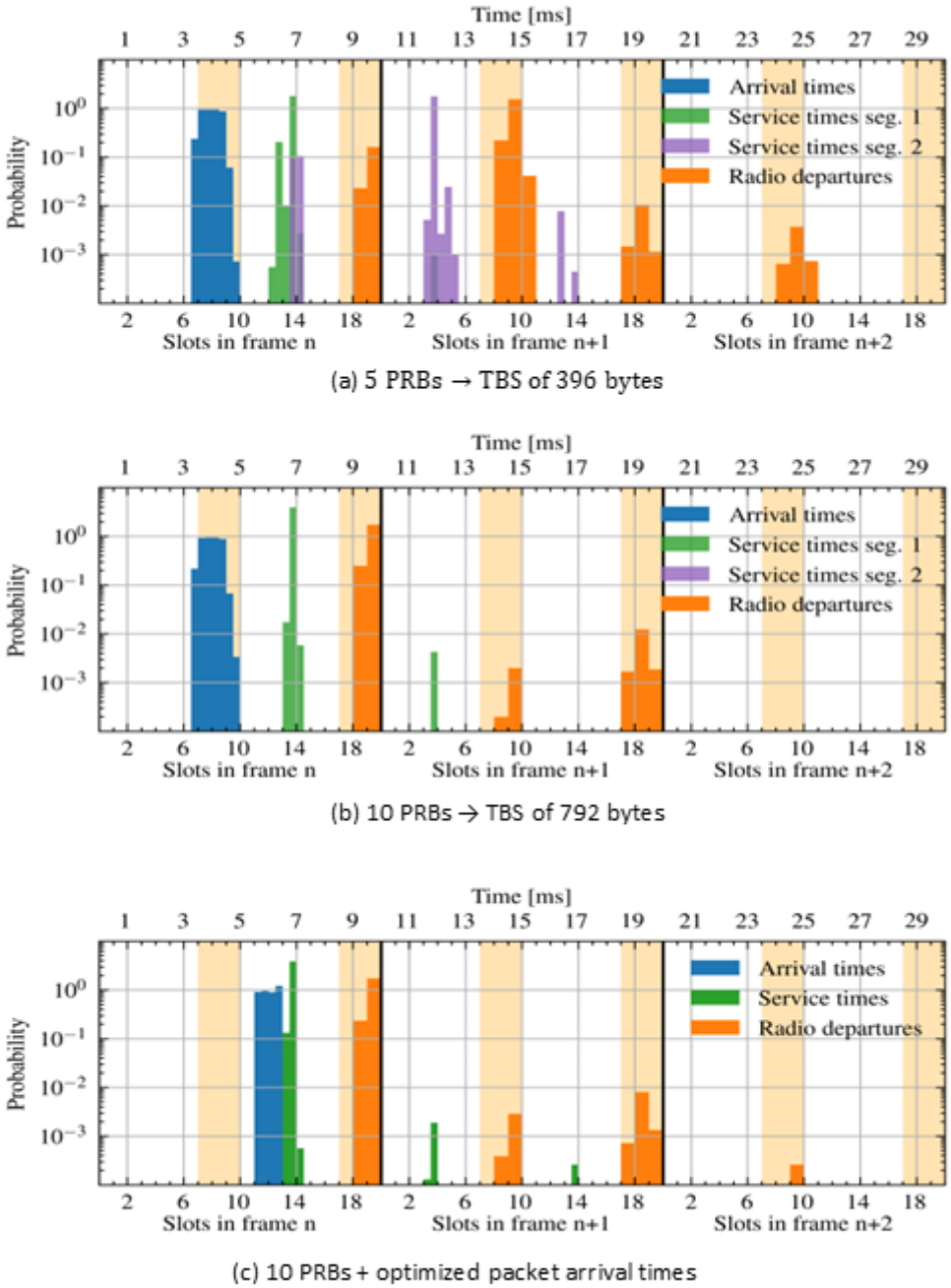


Figure 3-9: Histograms of service times and radio departure times.

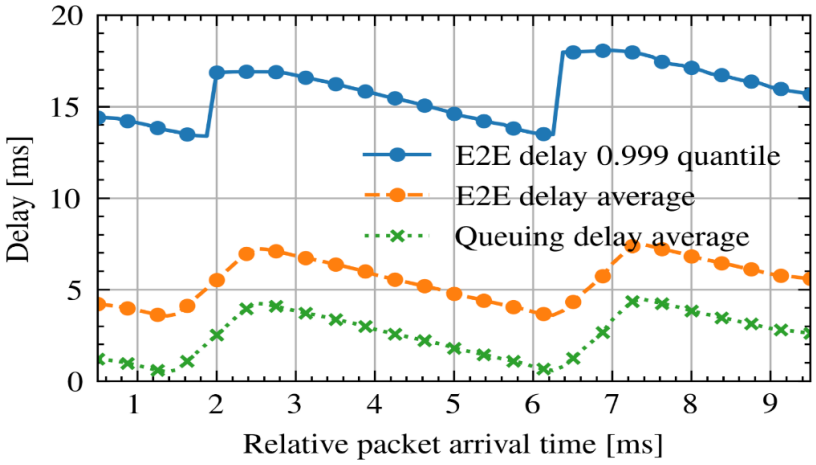


Figure 3-10: Minimizing end-to-end delay through optimizing arrival times offset relative to the 5G TDD period.

An additional insight from our analysis highlights the difference between lower and higher end-to-end delays, with retransmission delays becoming increasingly significant, constituting up to 50 % of the total packet delay. This pattern is consistent across a larger set of experiments presented in D4.3 [D6G-D4.3], emphasizing that infrequent but significant retransmissions are the primary factor behind outliers in end-to-end packet delays. This can be explained by the importance of retransmissions to provide high system reliability with good spectral efficiency, as explained in D2.1 [D6G-D2.1].

3.2.4 Key Takeaways

- EDAF is a tool that allows for delay decomposition in a running mobile network.
- Through a staged experimentation we show how different delay elements vary and contribute to the overall end-to-end delay, providing a novel level of explainability.

3.3 Time Synchronization Validation

3.3.1 Introduction

In this section we present the simulation-based validation of hot standby Grand Master (GM) in 6G-TSN networks using the simulation framework for time synchronization as presented in [D6G-D4.4]. We explore the performance analysis of different time synchronization architectures as presented in [D6G-D2.2] under failures, focusing on the comparison of hot standby operation and the Best timeTransmitter Clock Algorithm (BTCA).

The validations model a network consisting of TSN end stations, switches, and 6G DetCom nodes. It evaluates the synchronization performance across various failure scenarios by measuring metrics such as clock offset and out-of-sync time.

3.3.2 Validation Setup and Scenarios

The simulated 6G-TSN network includes three TSN end stations: Station 1 connects to the NW-TT through Switch A, Station 2 to the DS-TT 1 via Switch B, and Station 3 to the DS-TT 2. The network setup is shown in Figure 3-11 and Figure 3-12.

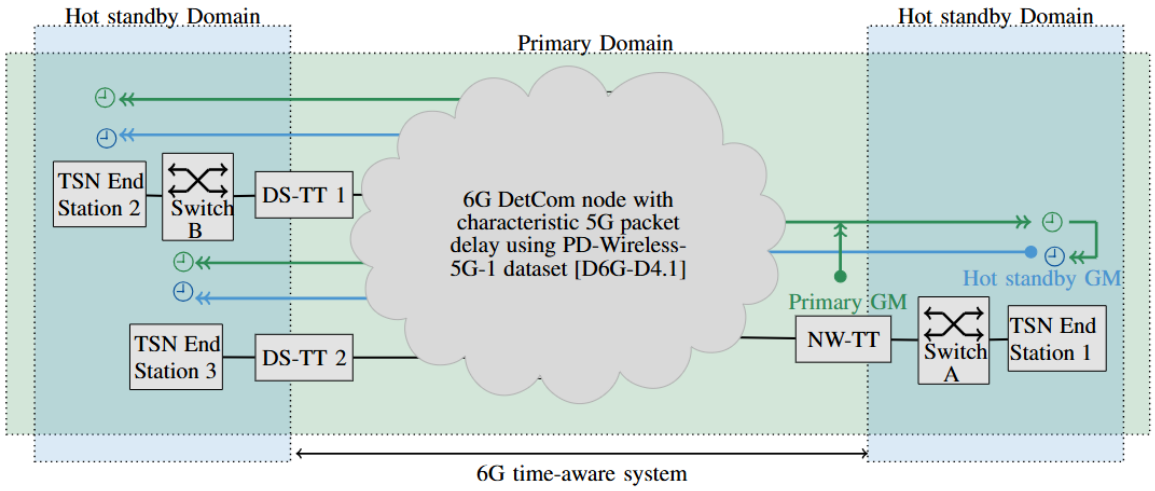


Figure 3-11: Hot standby scenario 1: NW-TT as primary GM and network-side TSN end station as hot standby GM.

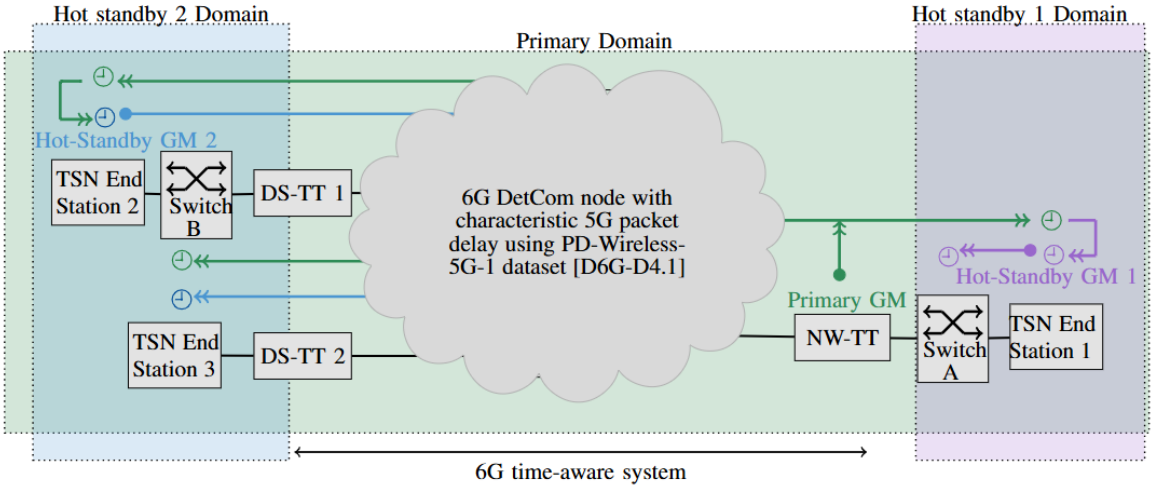


Figure 3-12: Hot standby scenario 2: NW-TT as primary GM and the hot standby GM 1 on a network-side TSN end station and the hot standby GM 2 on a device side TSN end station.

The 6G system (6GS) is modeled using a DetCom node, incorporating uplink/downlink delays based on real-world 5G datasets. TSN translators are allowed a clock drift of ±450 ns to meet the 3GPP 900 ns timing budget. The TSN clock behavior is aligned to IEEE 802.1AS standards [IEEE20-8021AS] – featuring a simplified maximum time deviation of 5 ns over 1 second and frequency offsets up to 100 ppm. All gPTP devices follow IEEE-recommended settings: Sync messages every 125 ms and Announce messages every second, both with a three-interval timeout. Two BTCA variants are evaluated: one sends Announce messages immediately after a GM change (BTCA), and the other delays them until the current interval ends (BTCA2).

Three synchronization scenarios are tested. In **Hot Standby Scenario 1** (Figure 3-11), BTCA is off, and Station 1 serves as the hot standby GM, covering both sides of the 6GS when the NW-TT (primary GM) fails. In **Hot Standby Scenario 2** (Figure 3-12), Station 1 and Station 2 act as separate hot standby GMs

for the network and device sides, respectively, forming two independent synchronization domains. In the **BTCA scenario**, GM selection is dynamic, with settings tuned to ensure NW-TT is favored – replicating the GM hierarchy of the standby setups.

Each scenario is tested against three failure cases: (1) loss of 6G timing at NW-TT (10–50 s), (2) disconnection between NW-TT and DS-TTs (70–110 s), and (3) failure of the link between NW-TT and Switch A (130–170 s).

3.3.3 Validation Results

Impact of 6G Time-aware System:

The validations first examine how 6G clock drift influences synchronization accuracy in Hot Standby Scenario 1. During normal operation, TSN clocks remain tightly aligned with 6G clocks as shown in Figure 3-13, staying within the 900 ns budget, confirming that 6G timing plays a crucial role in overall synchronization. To focus purely on failure impacts, later simulations assume ideal (drift-free) 6G clocks.

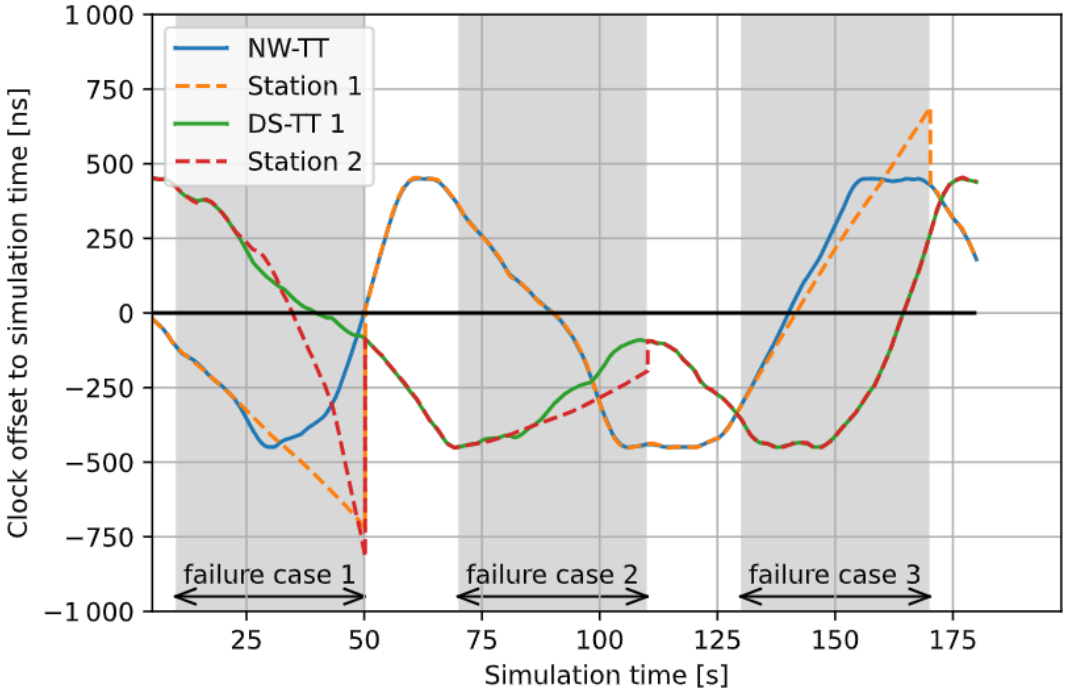


Figure 3-13: Clock drifts for devices in the Hot Standby Scenario 1 with drifting 6G clocks. The clock offset = 0 s is shown with a black line.

Adaptability to Failures:

Here we compare how hot standby operation and BTCA respond to failures for different time synchronization scenarios, as shown in Figure 3-14.

- *Failure 1 (NW-TT loses timing)*: In Hot Standby Scenario 1, all devices smoothly switch to the backup GM (Station 1), maintaining unified synchronization. In Scenario 2, the network splits into two time domains – one syncing to Station 1, the other to Station 2 – leading to drift

between segments. BTCA behaves like Scenario 1, selecting Station 1 as GM and preserving alignment.

- *Failure 2 (NW-TT to DS-TTs link breaks)*: The network splits due to a lack of redundancy. In Scenario 1, the device-side nodes drift due to no backup GM. In Scenario 2, Station 2 keeps part of the device side synchronized, but Station 3 drifts due to isolation. BTCA behaves similarly, eventually electing Station 2 as GM as part of the device side.
- *Failure 3 (NW-TT to switch A link fails)*: All approaches show similar performance. The device side stays synced to NW-TT, while the network side syncs to Station 1, which either pre-exists (hot standby) or is selected (BTCA) as GM.

In summary, BTCA consistently elects one GM per connected segment, preventing multiple competing domains and ensuring alignment within each network segment. In contrast, a static hot standby configuration can leave network segments without a GM or cause dual-GM synchronization, even when a single-GM configuration is still possible.

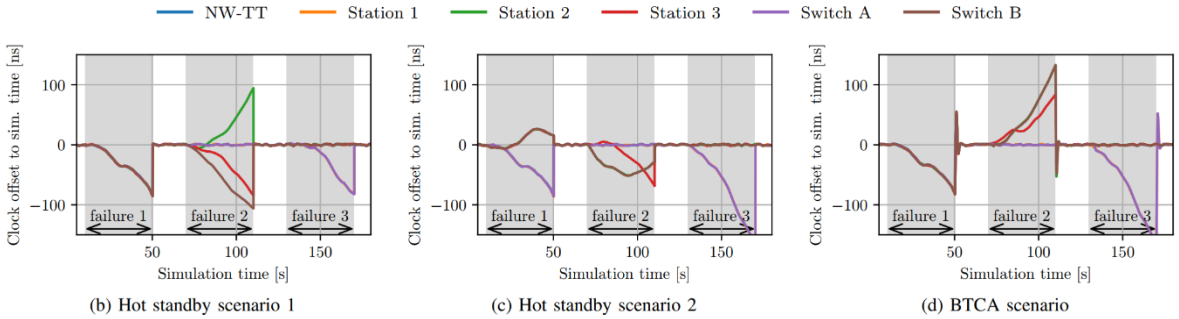


Figure 3-14: Drift of TSN clocks in a setup with ideal 6G clocks. Each grey area corresponds to one of the three failure cases.

Out-of-sync time

Even though BTCA is able to adapt to changing network environments, its reaction time to failures is categorized as slow. In order to analyze this claim in more detail, we compare the out-of-sync time (Figure 3-15) and the resulting clock offset (Figure 3-16) for Switch A in Hot Standby Scenario 1 against the two BTCA implementations in failure case 3.

In Hot Standby Scenario 1, Switch A quickly switches to a preconfigured backup GM, resulting in minimal out-of-sync time and negligible clock offset. In contrast, BTCA must detect the failure and initiate GM re-selection, leading to longer synchronization delays and greater clock deviations. Simulations with 10 and 50 hops between Switch A and Station 1 further show that hop count has little effect on the hot standby performance, while it significantly impacts BTCA – especially BTCA2, where delayed Announce messages increase recovery time. The slower response in BTCA is primarily due to its longer Announce timeout compared to Sync timeout and the time required for GM selection messaging. Overall, hot standby proves faster and more stable in restoring synchronization, outperforming BTCA in both out-of-sync time and resulting clock offset.

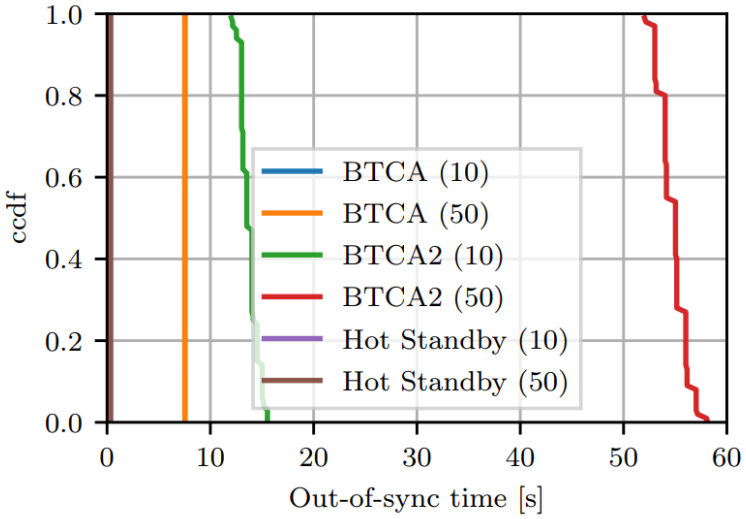


Figure 3-15: Complementary cumulative distribution of the out-of-sync time of Switch A. The number in parentheses indicates the number of hops between Switch A and the GM.

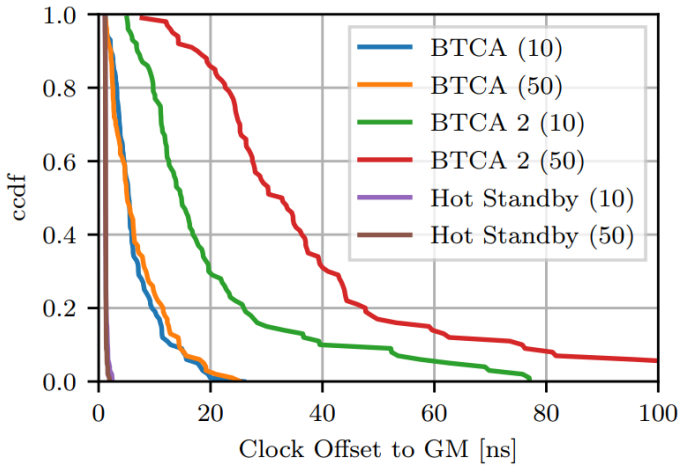


Figure 3-16: Complementary cumulative distribution of the maximum offset between Switch A and newly selected GM Station 1. The number in parentheses indicates the number of hops between the Switch A and the GM.

3.3.4 Key Takeaways

In summary, we learn from the validations that scenarios using a hot standby GM show better performance than BTCA in terms of clock drift and out-of-sync time during failures. However, static GM configurations (hot standby) can leave parts of the network without a GM in some failure scenarios. BTCA provides dynamic GM selection, offering better coverage in such cases, though with delayed response. The choice between hot standby (static) and BTCA (dynamic) should depend on the network's topology and redundancy level. In networks with multiple redundant links, static GM with hot standby is often more effective. In networks with limited redundancy, dynamic GM selection via BTCA can offer better resilience during failures.

3.4 Security Validation on Time Synchronization Process

3.4.1 Introduction

In the DETERMINISTIC6G project, the security-by-design task focuses on specifying and designing a comprehensive end-to-end architecture and framework for deterministic networking in virtualized and edge-based heterogeneous networking environments. In this section, we present an evaluation of the proposed monitoring security mechanism to enhance resiliency by detecting threats in an end-to-end scenario of time synchronization.

The approach includes a pipeline for traffic collection, feature extraction, and learning-based detection of performance and security issues affecting time synchronization process. Techniques employed include In-band Network Telemetry (INT), P4-programmable data planes, and high-precision telemetry.

The evaluation demonstrates that threats targeting deterministic applications, such as TDAs, cannot be effectively detected through traditional security mechanisms alone, but their detection relies on the continuous monitoring of timing behavior. By observing anomalies in clock offset and Inter-Arrival Time (IAT) variations, we enable timely detection and localization of subtle, delay-based attacks that would otherwise go unnoticed.

3.4.2 Methodology and Setup

The implementation of the emulator and its usage are detailed in D4.4. It is based on Mininet and emulates a PTP-based time synchronization network incorporating P4-programmable transparent clocks (TCs).

The testbed, illustrated in Figure 3-17, consists of a Dell laptop running Ubuntu 22.04 (14 cores, 32 GB RAM) and a Raspberry Pi 3 running Debian 11. The laptop runs the Mininet-based virtual network, which includes a PTP server and a chain of P4-based TCs. The Raspberry Pi acts as the PTP client and connects to the laptop via a RJ45 Ethernet connector.

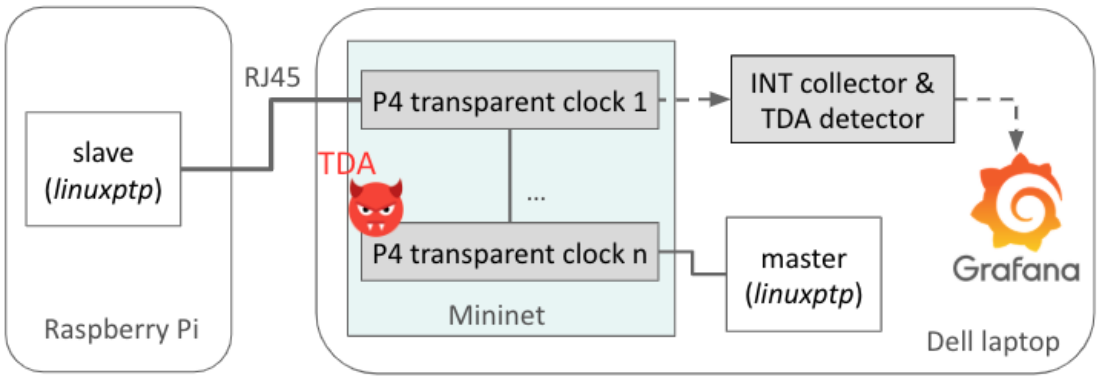


Figure 3-17: Testbed setup of security validation of time synchronization process.

In this setup, all PTP clocks, including server, client, and TCs, use Linux software timestamping although LinuxPTP supports PTP Hardware Clocks (PHC). The software-based approach generates timestamps through kernel system calls, providing sufficient precision for packet timing. While this method offers broader compatibility since it requires no specialized hardware, it inherently exhibits lower precision compared to hardware timestamping due to kernel scheduling latencies and software stack delays. This design choice reflects a deliberate trade-off: hardware timestamping delivers superior

synchronization accuracy, typically sub-microsecond precision, but mandates PHC-capable network interfaces, whereas software timestamping maintains adequate precision, typically tens to hundreds of microseconds, with universal deployability.

To support telemetry and TDA detection, an INT collector is deployed near the client to sniff traffic at the last TC. This collector captures telemetry data embedded in PTP packet extensions and feeds it to Grafana for visualization and analysis.

To emulate TDAs, we use the 6GDetCom Delay Emulator. A compromised TC is simulated by introducing asymmetric delays without updating the `correctionField`, thus, creating misleading measurements. This setup mimics a realistic attack scenario in which delays are injected only on one communication path.

Additionally, we validate the framework's implementability through:

- Integration of INT data into PTP extension fields.
- Packet-level detection and localization of attacks based on telemetry analysis.

3.4.3 Result Analysis

Correctness and Compatibility

We first assess the accuracy and correctness of our P4-based TCs. Using the testbed from Figure 3-17, we run time synchronization experiments across 10 transparent clocks, both with our P4-based implementation and with standard LinuxPTP-based TCs. Each run lasts 600 seconds and is repeated 10 times for consistency.

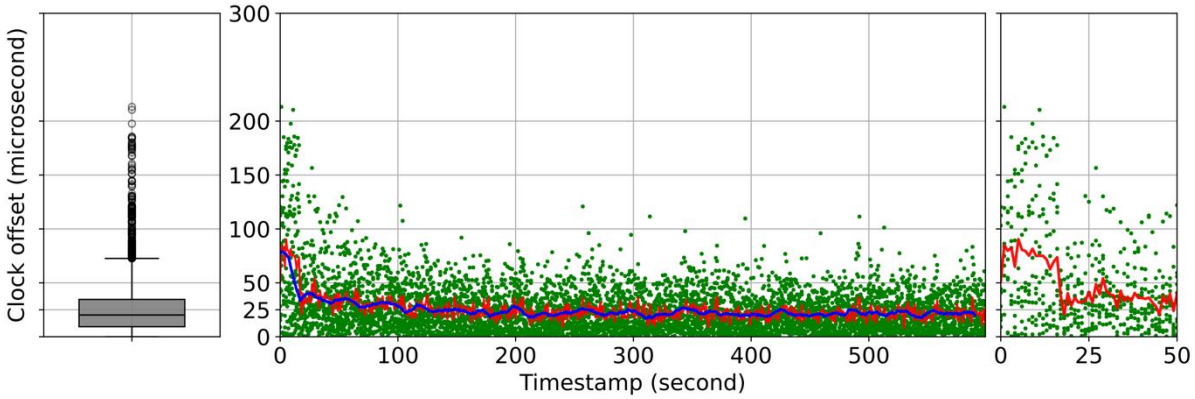


Figure 3-18: Clock offset during PTP time synchronization across 10 LinuxPTP-based transparent clocks

Figure 3-18 and Figure 3-19 compare the clock offsets observed in both setups. Each figure includes:

- A boxplot summarizing the offset distribution.
- A timeline of individual measurements.
- A zoomed-in view of the first 50 seconds.

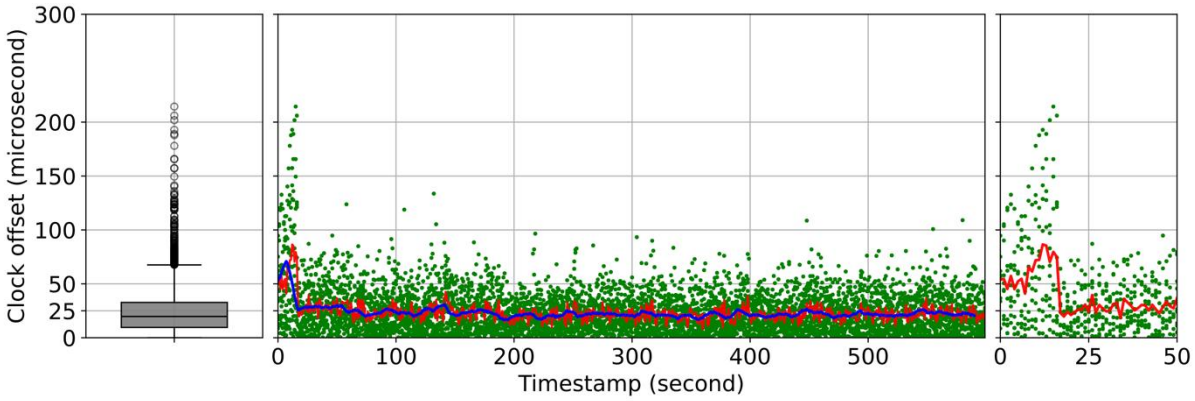


Figure 3-19: Clock offset during PTP time synchronization across 10 P4-based transparent clocks

Initial clock offsets are high due to synchronization delay but stabilize quickly. The results show that after the calibration phase, clock offsets remain below 150 µs, which is consistent with the expected accuracy of Linux software timestamping. This confirms that P4-based TCs are functionally equivalent to LinuxPTP-based ones in terms of synchronization performance.

INT Integration

Figure 3-20 shows a Wireshark capture of a Follow_Up message containing custom TLVs used to carry INT data. Each TLV includes the following fields:

- *Switch ID*: a unique identifier for the TC.
- *Ingress and egress timestamps*: the precise timestamps captured at the ingress and egress of the corresponding Sync packet.
- *Correction Ns*: the value of the correctionField before the TC applied its local residence time adjustment.

This figure illustrates how INT provides multi-hop visibility and enables precise delay tracking directly within PTP packets. By embedding telemetry data in PTP extension TLV fields, this approach avoids the need for extra probe packets, thereby minimizing overhead and leveraging the existing PTP infrastructure for efficient and scalable telemetry collection.

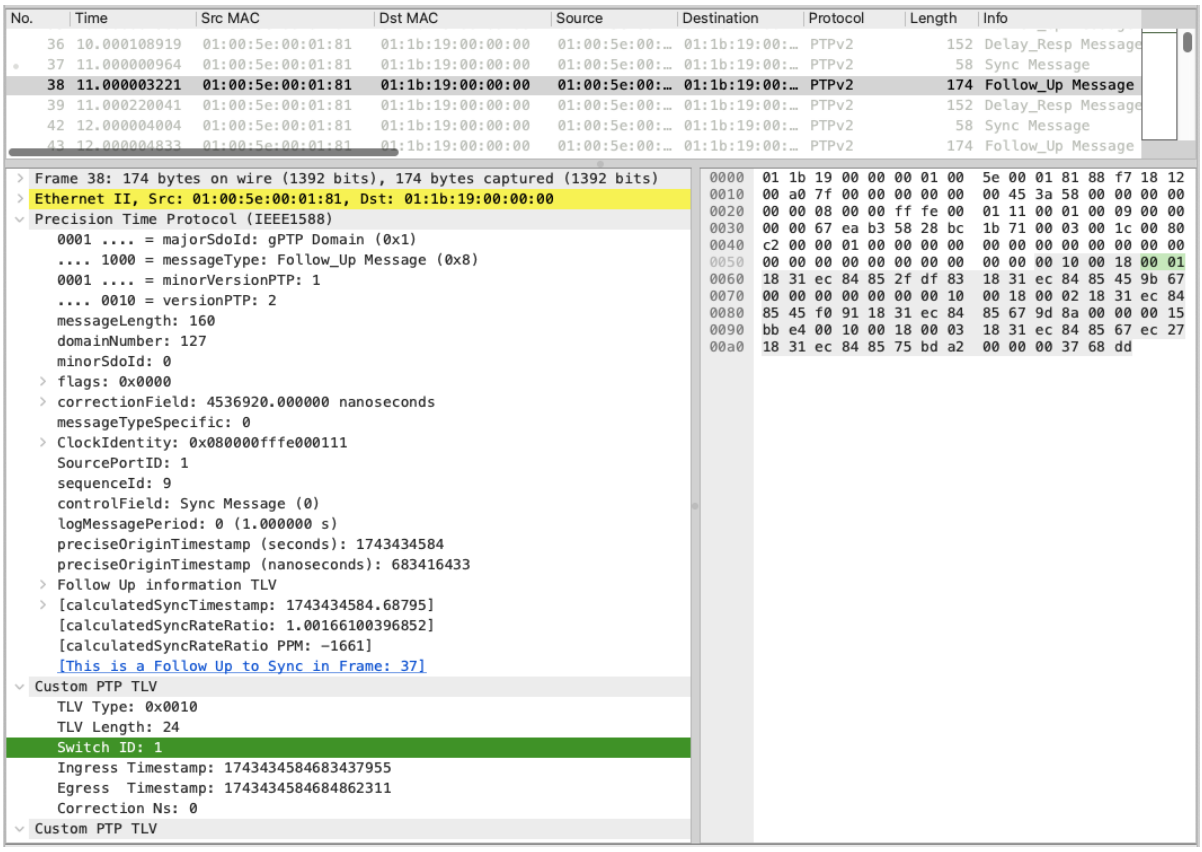


Figure 3-20: INT embedded in TLV extensions of PTP packets.

TDA Detection and Localization

Figure 3-21 offers a graphical representation of the framework's ability to continuously monitor, detect, and localize TDAs at the packet level for each PTP message. The charts are divided by message type: Sync messages are shown on the left side, while the Delay_Req messages appear on the right side. In all charts, the x-axis represents the sequence number of the corresponding messages.

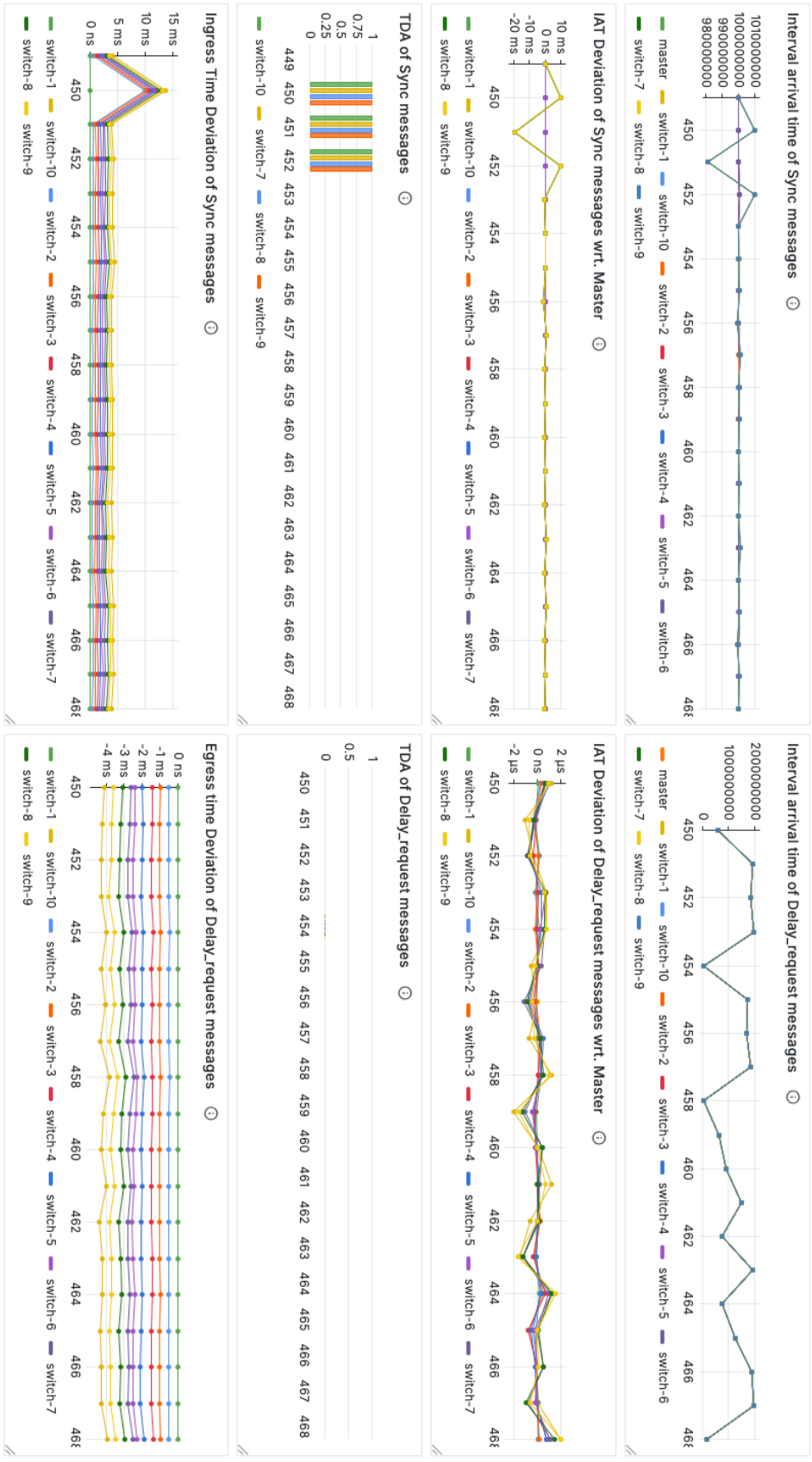


Figure 3-21: Monitoring time-delay attack on PTP time synchronization.

The top-row plots show the IAT variations for Sync and Delay_Req messages across the server and each TC. As expected, Sync messages arrive almost exactly one second apart. Delay_Req messages from the LinxuPTP client, however, vary between 0 and $2^{logMinDelayReqInterval} + 1 = 2$ seconds, as detailed in Section 9.5.11.2 of [IEEE1588-2019]. Despite this variability, the IAT measured at each TC closely track those seen at the server, confirming consistent propagation behavior across the network.

The second-row plots display the difference between the IAT values at each TC and those recorded at the server. These differences help identify deviations that may indicate potential TDAs.

The third-row plots translate these deviations into binary alerts, showing exactly which TCs lie on the path where a TDA has been detected.

Finally, the bottom-row plots compare packet arrival times for Sync, on the left side, and packet departure times for Delay_Req, on the right side, across the TCs. Although the TCs themselves may not be mutually synchronized, these relative timing graphs reveal propagation delays between hops. Importantly, because the detection logic relies on variations in timing rather than absolute timestamps, perfect clock alignment between TCs is not required. This reinforces the robustness of the approach for identifying TDAs based on behavioral anomalies.

3.4.4 Key Takeaways

Effective detection of Time-Delay Attacks: The validation demonstrates that the proposed monitoring framework, based on P4-programmable transparent clocks and INT, can accurately detect and localize TDAs in a time synchronization network, even when such attacks introduce subtle, asymmetric delays without modifying packet content.

Feasibility of a programmable, software-based security framework: The integration of INT, and real-time packet-level monitoring confirms the viability of a software-defined, security-by-design approach for enhancing the resilience of deterministic networking systems, without relying on specialized hardware.

3.5 Edge Cloud Validation

3.5.1 Delay Comparison with Multiaccess Edge Computing (MEC) vs Public Cloud

The architecture of a Multiaccess Edge Computing (MEC) system is split into Host Level and System Level components as illustrated in Figure 3-22. The MEC System Level Management as specified by ETSI consists of the component that holds the MEC Orchestrator. The latter is deployed as an application function (AF) interacting with the Network Exposure Function (NEF) for provisioning, managing, and monitoring of MEC hosts. The MEC host, however, is deployed with the UPF responsible for user plane traffic steering to MEC applications in the UE.

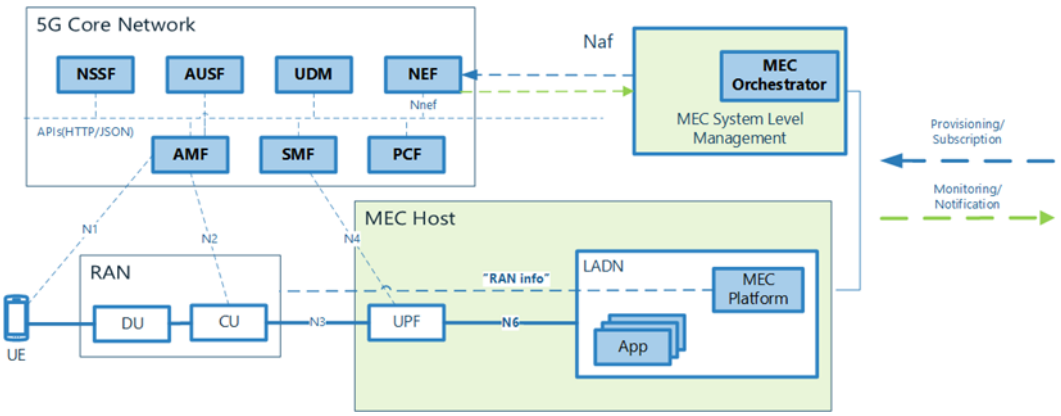


Figure 3-22: Multi-access Edge Computing, ETSI architecture.

Deploying a MEC system jointly with a 5G Service Based Architecture network is an opportunity to comply with deterministic communications as presented in [D6G-D3.5]. An important aspect of the distributed design of next generation mobile networks is the combination of on-premises and cloud-based processing assets.

A prototype of the MEC architecture as specified in 3GPP was completed and an iperf3 application was used for collecting some performance data when using MEC on premises versus cloud installations.

Following is the flow diagram describing the selection of the Edge computing server based on application requirements. The edge application location is selected based on User Plane (UP) parameters combined with the profile and Key Parameter Indicators (KPIs) of Edge Application Server (EAS) and Application Client (AC) as shown in Figure 3-23.

1. The EAS registers to Edge Configuration Server (ECS) (considered as an AF) with its profile information. EAS's profile also includes KPIs supported by the application.
2. ECS deploys EAS's registration data in an internal mongodb database to be accessed by other NFs.
3. ECS then requests the list of active UEs from the Network Data Analytics Function (NWDAF) i.e., subscribers showing a CONNECTED status in the NWDAF. Since multiple slices can be active, the request is done per Group ID to identify UEs connected to different slices.
4. ECS discovers the active UE with the following steps:
   a. Using the list of active UEs from previous step, the ECS requests the aggregated delay information of each UE from the Network Data Analytics Function (NWDAF). The NWDAF collects information from the core network functions (NF) and in case of Edge computing it accesses the User Plane Function (UPF) information that includes traffic associated to network slices. NWDAF also retrieves information from Access Mobility Function (AMF) that includes UE status also required for Edge computing.
   b. The ECS computes the average delay per slice.
   c. ECS also requests the slice capacity from NWDAF.
   d. In this context, the ECS is considered as an internal AF. Therefore, it sends an NF_discovery request to Network Repository Function (NRF) to learn about active UPFs. The response includes IP addresses of UPFs and their associated slices.

5. The ECS then selects the slice with lowest delay and high enough capacity to support MEC traffic and updates the EAS information with the corresponding UP path (IP address).
6. To change a subscriber's traffic routing, ECS requests the Policy Control Function (PCF) for authorization providing traffic routing information.
7. The PCF instructs the Session Management Function (SMF) for a PDU session modification with the IP address of selected UPF.
8. Packet Data Unit (PDU) session modification messages between the UE, AMF, and SMF. SMF may also request EAS's deployment information.
9. UE's PDU session is updated with new traffic routing information to establish data traffic towards the edge application.

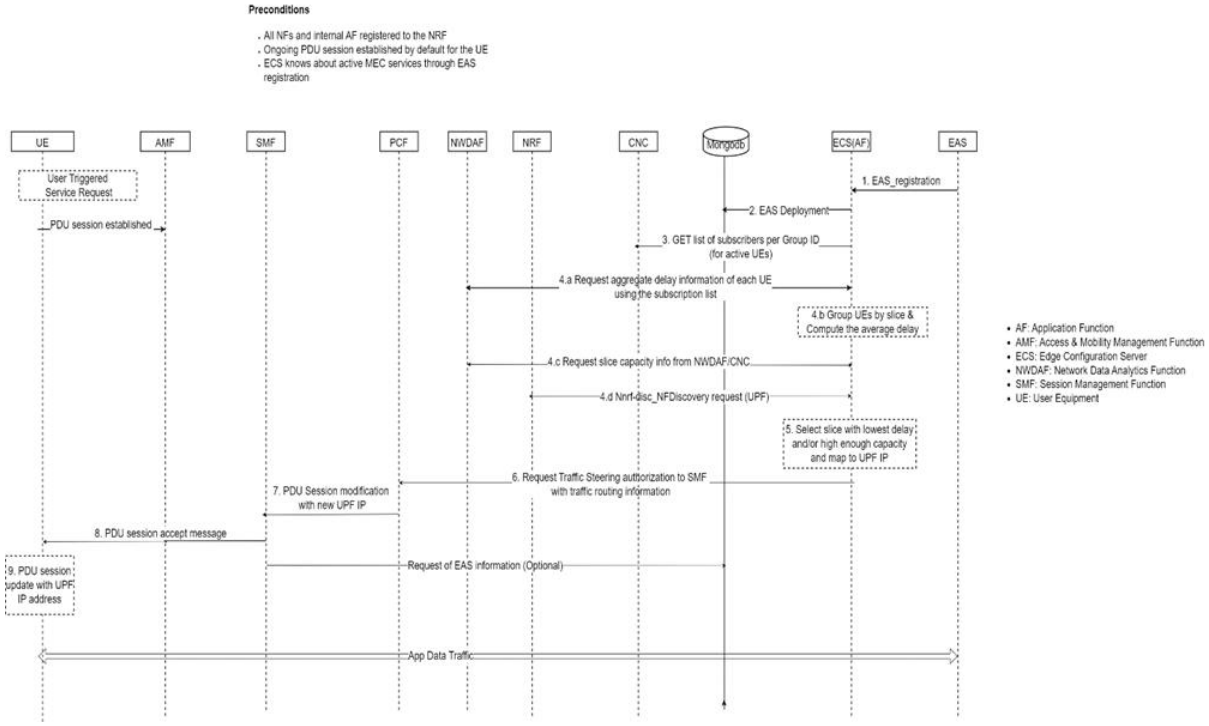


Figure 3-23: UPF Selection for EAS integration.

3GPP has defined a set of profiles or KPIs shown in Figure 3-24 that have to be considered when selecting the location of the edge application to fulfil those KPIs.

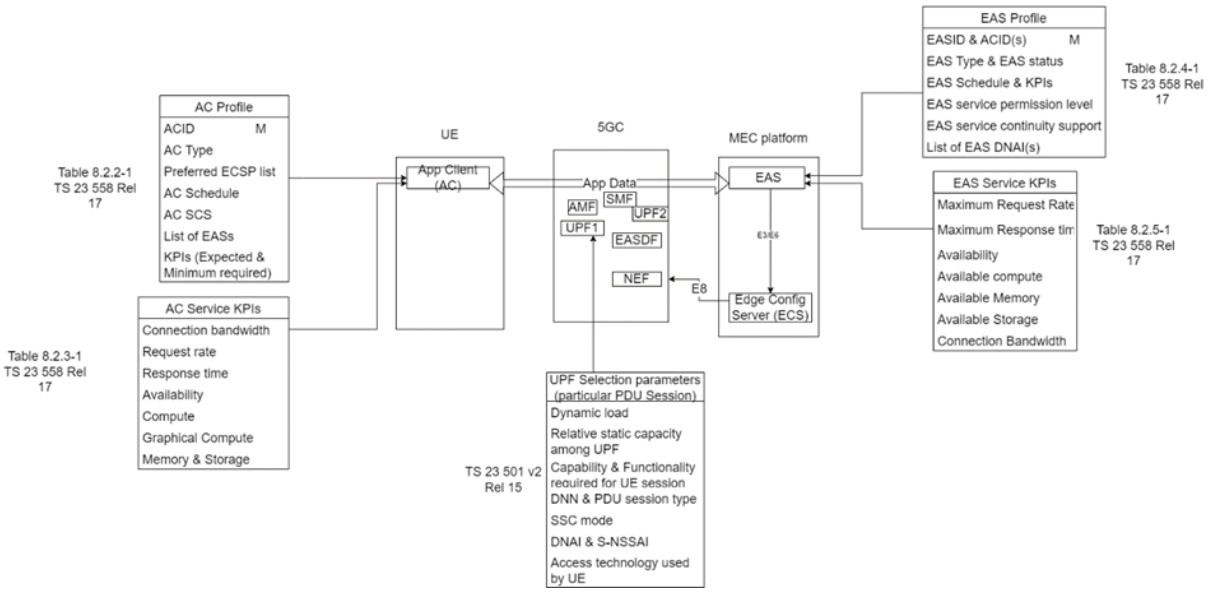


Figure 3-24: Profiles and KPIs of edge components and UPF parameters.

In order to be equipment agnostic, the measurements include all-in-one base stations and O-RAN with distributed Radio Unit (RU), Distributed Unit (DU), and Centralized Unit (CU) modules. The results include both throughput and delay measurements, but since the focus is on deterministic communications only delay measurements are shown.

The results presented in Figure 3-25 show that delay distribution has a larger deviation when accessing the applications on cloud compared to the results when measuring the delay to the applications deployed on the edge. Figure 3-26 shows similar results but this time using integrated cmWave gNB instead of distributed O-RAN gNB shown in Figure 3-25. Therefore, edge computing provides not only lower end-to-end delay for both O-RAN and integrated gNB, but the delay distribution is more sparse when the applications are deployed on the cloud instead of edge computing.

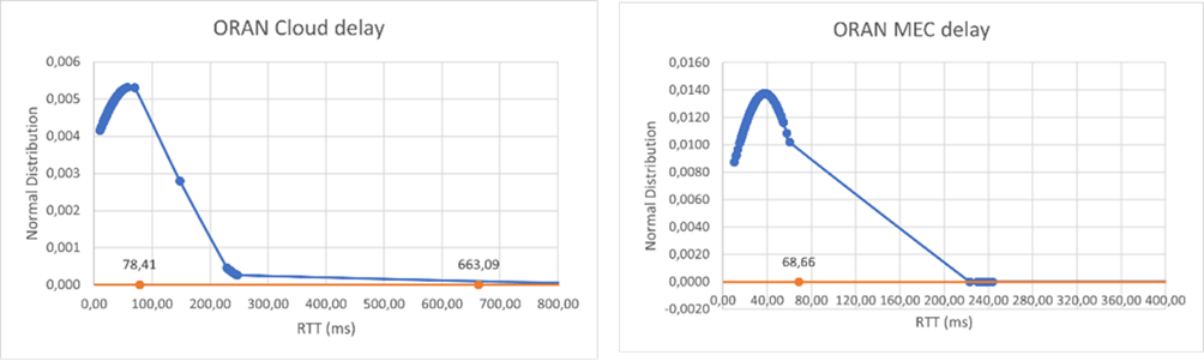


Figure 3-25: RTT delay with O-RAN gNB for cloud and on-premises MEC.

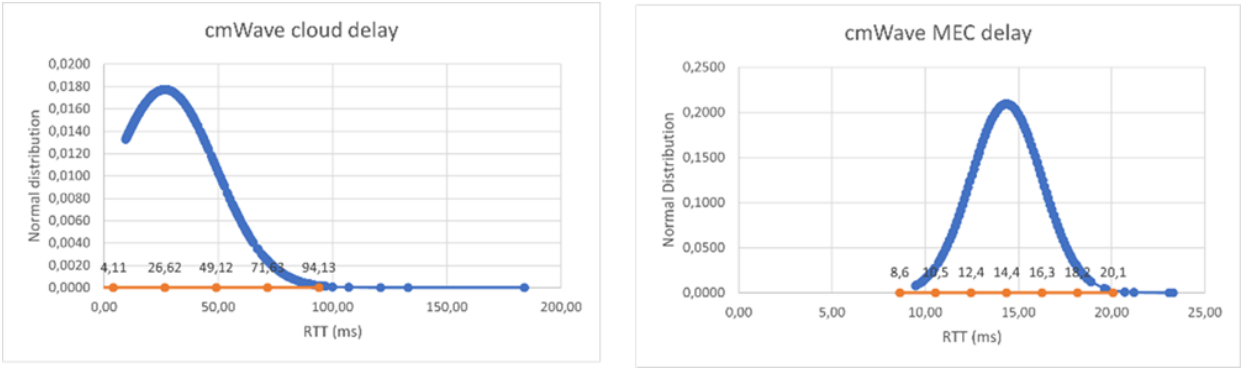


Figure 3-26: RTT delay for integrated cmWave gNB (N78) for cloud and on-premises MEC.

3.5.2 IEEE 802.1Qbv-aware Traffic Handling in Edge Computing Domain

In D3.3 [D6G-D3.3], two TAPRIO-based traffic scheduling methods have been presented for the seamless support of 802.1Qbv scheduling in virtualized deployment for containerized applications. One method applies the TAPRIO queuing discipline on the virtual Ethernet interface of the containers hosting time-critical applications – referred as *per-container shaping mechanism* –, while the other method uses a *centralized traffic handling scheme*, built on an Open Virtual Switch (OVS) – based Container Network Interface (CNI) plugin. The detailed description of the methods can be found in Section 2.4.3 of D3.3 [D6G-D3.3].

To evaluate and compare the methods, the DETERMINISTIC6G simulation framework [D6G-D4.4] is used to model the virtualized networking of a cloud host. In each simulation scenario, five containerized applications are used that periodically transmit 1000 byte packets every 10 ms towards the host's physical NIC. Each application is modeled on a separate host using deadline scheduling (SCHED_DEADLINE) without cross-application interference. The simulation time was set to 10 s, so 1000 packet transmission per application are simulated in each scenario.

The simulation setup of the per-container shaping mechanism can be seen in Figure 3-27.

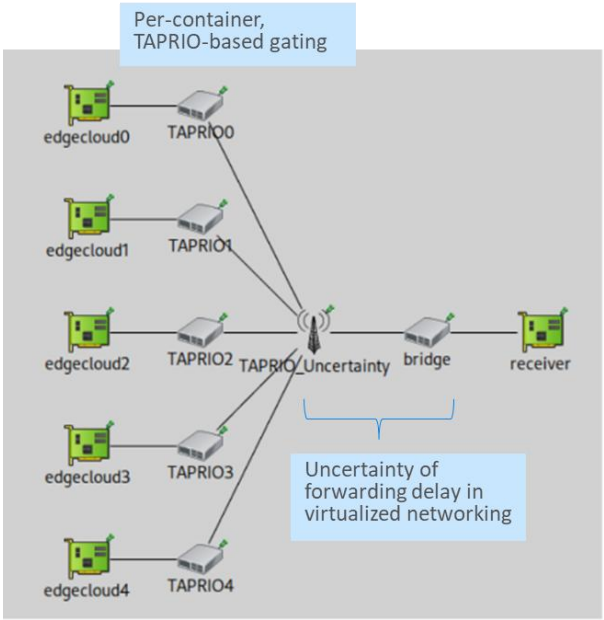


Figure 3-27: Simulation setup for the per-container shaping mechanism.

To evaluate the per-container shaping mechanism, each application's host (edgecloud0 – edgecloud4 in the figure) is connected to a separate bridge (TAPRIO0 – TAPRIO4), representing the per-container shaping. Each bridge is configured with the appropriate TAPRIO settings. For modeling the uncertainty of the forwarding delay in the virtualized networking, an additional bridge is used. Since the simulation setup covers only a single cloud host, the "receiver" represents the host's NIC.

Figure 3-28 showcases the results of the per-container shaping mechanism, modelling the uncertainty of the virtualized networking and TAPRIO by means of a random uniform delay component between $20\mu s$ and $50\mu s$, and $100\mu s$ and $200\mu s$, respectively.

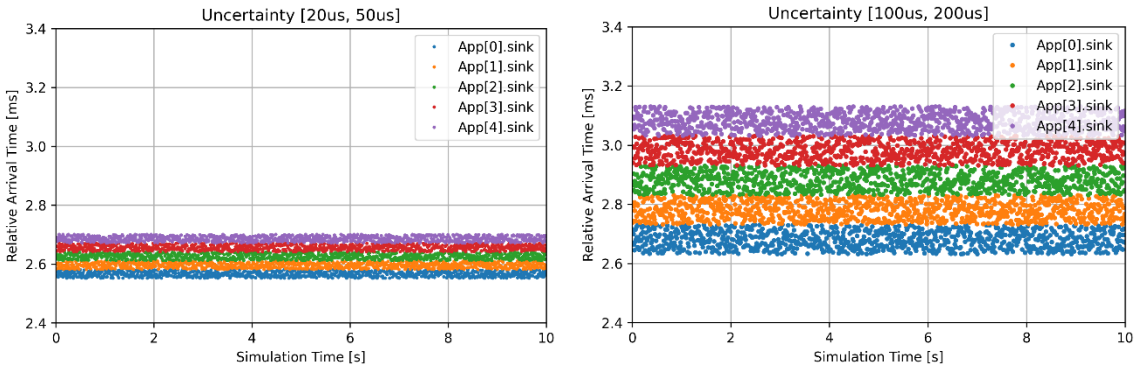


Figure 3-28: Packet arrival for per-container shaping, uniform forwarding delay distribution.

Figure 3-29 shows the results of the per-container shaping mechanism modeling the uncertainty of the virtualized networking and TAPRIO by a normal distributed random delay component. In the range of $20\mu s$ and $50\mu s$ (left side of the figure) a normal distribution with mean value of $35\mu s$, and a standard deviation of $5\mu s$ is applied; the values outside the $[20\mu s, 50\mu s]$ range are truncated. The uncertainty in the range $100\mu s$ and $200\mu s$ (right side of the figure) is modeled by a normal distribution with mean value of $150\mu s$, and a standard deviation of $15\mu s$; the values outside the $[100\mu s, 200\mu s]$ range are truncated.

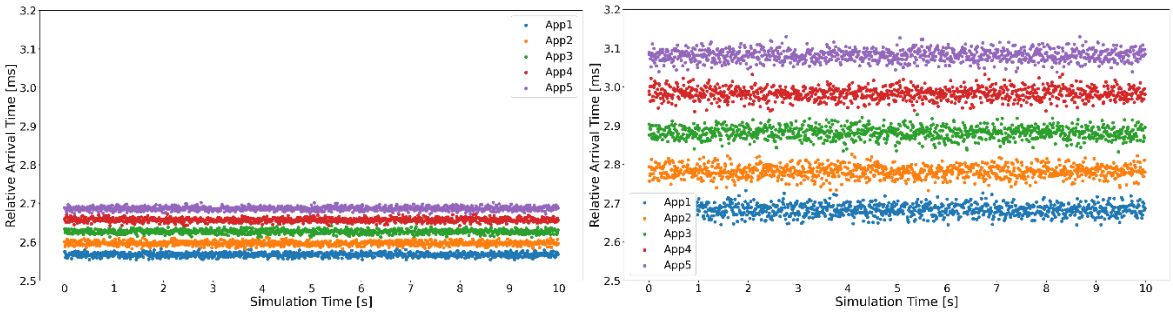


Figure 3-29: Packet arrival for per-container shaping, normal forwarding delay distribution.

The simulation results illustrate that the per-container shaping mechanism is capable to ensure the correct packet arrival order among the applications. However, it can also be concluded that employing the per-container shaping mechanism increases both packet delay variation and the end-to-end latency. The increase in the delay variation stems from the uncertainty, introduced by TAPRIO and the delay variance in virtualized networking. As discussed in Section 2.4.3 of D3.3 [D6G-D3.3], to

compensate this latency uncertainty, a guard timing structure is introduced, which requires to increase the length of the time slot for packet forwarding. A side effect of this is the increase of delay, experienced by each packet.

The simulation setup of the centralized traffic handling scheme is shown in Figure 3-30.

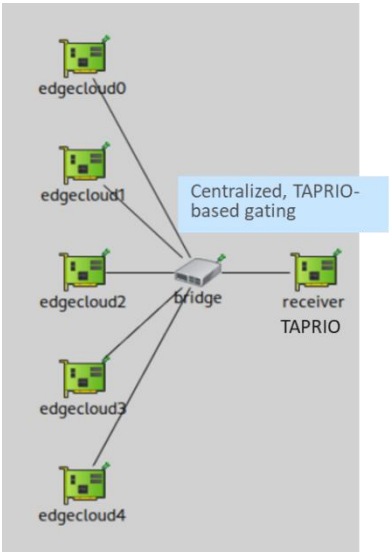


Figure 3-30: Simulation setup for the centralized traffic handling scheme.

Since a centralized approach is used, all application hosts are connected to a single bridge, where the TAPRIO-based gating scheme is configured. This bridge is directly connected to the receiver, which represents the host's physical NIC.

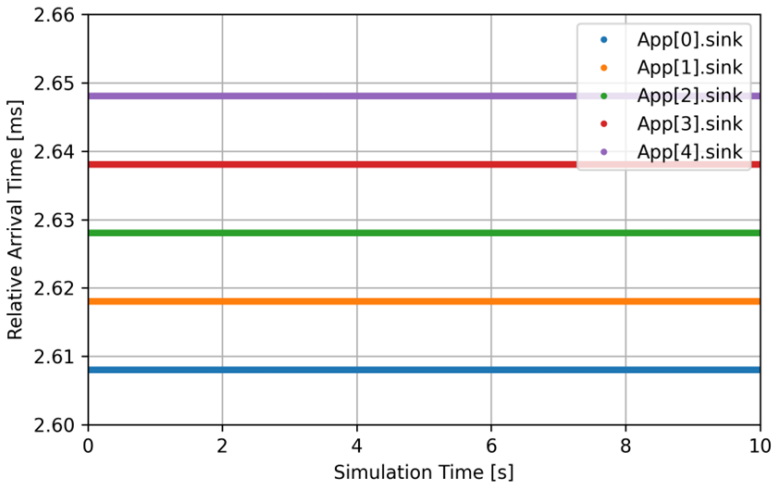


Figure 3-31: Packet arrival for centralized traffic handling scheme.

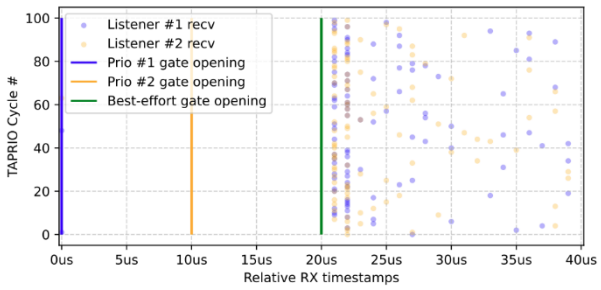
The simulation results in Figure 3-31 demonstrate that, due to the centralized operation (details can be found in Section 2.4.3 of D3.3 [D6G-D3.3], much stricter requirements can be met, as the delay uncertainty introduced by the virtualized networking does not affect the performance of this method.

As a takeaway, it can be concluded that the per-container shaping approach results in higher average delay and lower utilization (due to the guard times), but it requires simpler configuration, and the malfunction of any application does not cause negative effects. In contrast, the centralized approach can ensure minimized latency increase and maximized utilization; however, it requires a more complex scheduling configuration setup, which makes this solution more sensitive to any malfunctions in the timing of the virtualized application instances.

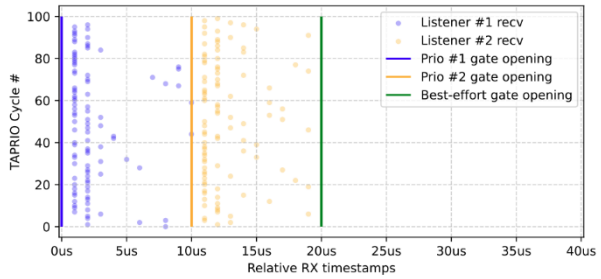
3.5.3 eBPF-Based Time-Aware Traffic Handling

In a Kubernetes environment, a time-sensitive application (TSA) microservice cannot properly define priorities. Even if the socket API allows the TSA to set priorities for its traffic, Kubernetes's network isolation mechanism removes the set priorities. This issue stems from the Linux kernel's network namespace isolation mechanism, which removes all metadata from packets.

The D3.5 [D6G-D3.5] deliverable's Section 2.2.2 presents a practical solution to this problem: the TSN metadata proxy. With that solution, precise scheduling is possible even if the TSA is running as a microservice and does not have access to the physical TSN NIC. To validate the operation of the TSN metadata proxy, a simple yet realistic environment is presented that uses an IEEE 802.1Qbv Time-Aware Shaper and traffic classes.



(a) Without TSN metadata proxy



(b) TSN metadata proxy enabled

Figure 3-32 Packet RX timestamps mapped into cycles of Time-Aware Shaper (802.1Qbv)

The following simple 802.1Qbv schedule on the TSN NIC with 40 microsecond cycle time can be used for evaluation:

- Priority 1: from 0 to 10 microsec
- Priority 2: from 10 to 20 microsec
- Best-effort: from 20 to 40 microsec

This cycle is repeated indefinitely, and the priorities are set by the TSAs.

Without the TSN proxy, the talker's priorities are deleted before they reach the NIC. As a result, they fall into the best-effort timeslot shown in Figure 3-32(a). Note that packets received outside of their timeslot are transmitted immediately after their gate opens. Therefore, we have more timestamps right after the gate opens than in the rest of the slot. With TSN proxy (Figure 3-32(b)), the priorities of the TSA packets are preserved. They are placed in their proper time slots and respect the correct gate openings. As one can see in the figure, Listener #1 receives packets in timeslot Priority 1 and Listener #2 receives packets in timeslot Priority 2.

3.5.4 Key Takeaways

The measurements obtained with the MEC prototype show the delay has shorter variation when using MEC compared to connecting through public Internet to cloud applications. The results show a long tail delay when connecting to cloud applications without MEC. Thus, running TSN applications on the MEC platform should provide smaller delay variation.

The simulation and the implementation-based results showcase that seamless support of 802.1Qbv traffic scheduling for cloudified applications can be achieved efficiently. Various traffic handling solutions can be applied in the virtualized networking of a cloud host deployment; the selection of the appropriate method depends on the timing requirements of the applications and the capabilities of the cloud ecosystem.

3.6 Exoskeleton Use Case Validation

3.6.1 Introduction

This section describes the work carried out to emulate the effects of a 6G-oriented network in a near-term occupational exoskeleton use case. As outlined in D1.1 [DET6G-D1.1] (see Section 4.3), the near-term scenario involves a wearable occupational exoskeleton worn by a worker to assist movements during repetitive and/or demanding physical tasks.

The active wearable robot described in the scenario includes an embedded processor that runs a low-level controller (LLC), which translates the desired joint assistance torque – computed by higher software layers – into current and voltage setpoints to drive the motors accordingly. The middle-level (MLC) and high-level controllers (HLC) are offloaded to the cloud relying on a 6G-based wireless network.

Offloading the control logic and reducing the onboard hardware components offers several advantages, including reduced cost, size, and power consumption. However, as the device architecture becomes less embedded and based on external computation, it must rely on highly dependable communication to ensure safety and consistency with its use and its risk analysis.

Indeed, emulating this scenario in a well-structured test bench is expected to produce preliminary results that will be extremely useful for effective 6G development.

3.6.2 Validation Methodology and Setup

The overall system architecture is shown in Figure 3-33 and includes:

- IUVO lower limb exoskeleton actuation unit test bench;
- 6GDetCom Network Delay Emulator;
- Edge Cloud Server and App emulator.

The test bench (TB), custom-designed by IUVO, is a robotic system capable of applying repetitive and reconfigurable work cycles to the exoskeleton's actuation unit (AU). The AU is defined as the subsystem responsible for converting electrical energy into mechanical power and transmitting it to assist the user's movements. The test bench is primarily used to perform endurance tests and to evaluate the performance of new exoskeleton designs.

The 6GDetCom Network Delay Emulator is an open-source software presented in Section 2.2 and developed by University of Stuttgart to assign an artificial delay to a network interface.

The edge cloud server and application emulator are a processing unit capable of executing in real-time (RT) the HLC and MLC off-loaded control layer of the exoskeleton.

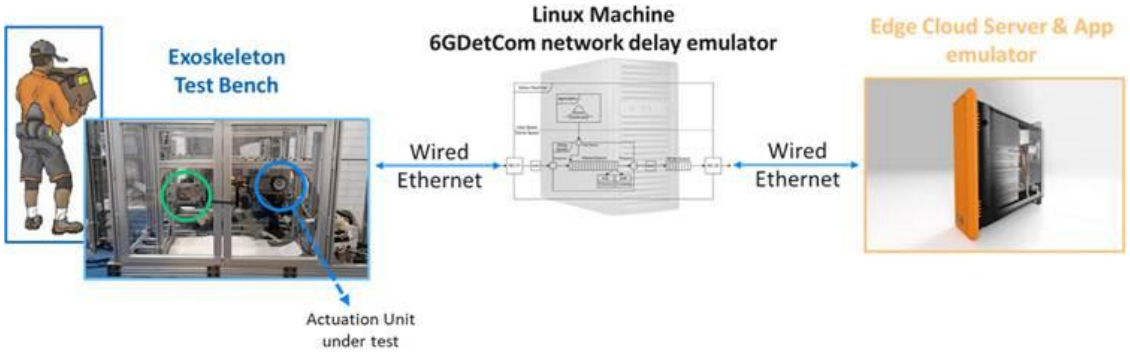


Figure 3-33: Exoskeleton use case validation, overall architecture.

The development of the mentioned architecture followed a roadmap that progressively addressed sequential activities hereafter explained:

- discussion and overall architecture definition;
- hardware integration;
- software integration;
- integration verification;
- test and result discussion.

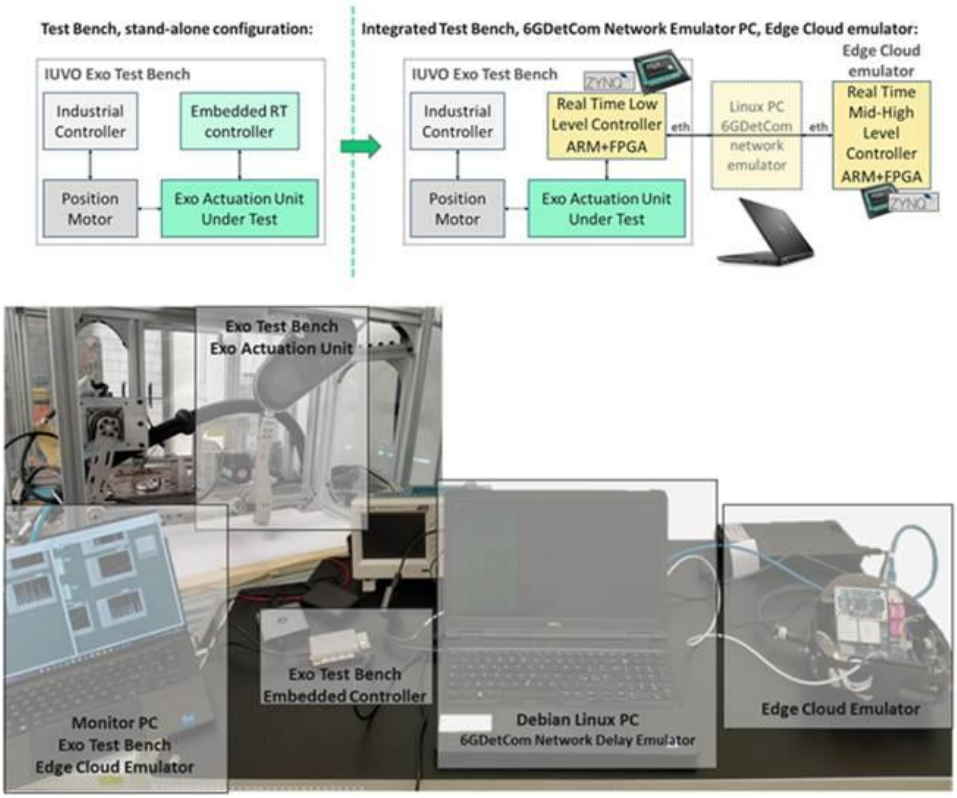


Figure 3-34: Exoskeleton use case validation, hardware integration.

Once the architecture was defined, the hardware was progressively integrated, with all components updated and implemented accordingly. As shown in Figure 3-34, the stand-alone configuration of the test bench includes an industrial controller and a motor that sets the position of the AU under test, emulating the load of a leg for a user wearing the exoskeleton. An AU from an active occupational exoskeleton is selected, installed, and mechanically coupled with the position motor to emulate limb joint movements during walking tasks at different, configurable velocities. The AU is controlled by an embedded RT processing unit, which executes the control algorithm and delivers assistance profiles in synchronization with the emulated leg position. The RT processing unit of the AU is then updated with a National Instruments SbRIO-9651, endowed with a Xilinx Zynq-7020 system-on-module. The powerful electronic embedded hardware architecture, based on a dual-core ARM and a Field-Programmable Gate Array (FPGA), enables external device integration over Ethernet bus. The same Zynq chipset is used for the edge cloud emulator, while a dedicated Debian PC capable of running the open-source 6GDetCom Network Delay Emulator is connected to achieve the final desired architecture.

Using the integrated hardware architecture described above, the software was progressively developed to emulate the desired exoskeleton offloaded control configuration. The LLC algorithms are executed on the AU's FPGA at a real-time frequency of 1 kHz. The AU's sensory data is down-sampled to a real-time frequency of 100 Hz by the RT operating system running on the ARM dual-core processor and streamed over UDP to the edge cloud emulator and its RT processing unit. Based on the received UDP packets, the MLC and HLC algorithms are executed at a real-time frequency of 100 Hz, and the results of their computations are sent back over UDP as input for the LLC controller. These sampling frequencies are commonly used in lower limb active exoskeletons [LBD+20]. In series with

the UDP/Ethernet bus, the Debian Linux PC runs the 6GDetCom Network Delay Emulator software. The execution within each Zynq processing unit is repeatable, with low jitter and high reliability. However, the use of non-deterministic UDP communication adds variability on top of the effects introduced by the 6G network emulator. A Serial Synchronous Interface (SSI), implemented in parallel with UDP directly at the FPGA level of each Zynq unit, allows synchronization of UDP transmission on both sides. It also enables sharing of a deterministically generated counter, which is useful for communication line diagnostics. The final integrated software architecture is shown in Figure 3-35, where the delay and the jitter on the shared SSI synchronization signal is measured and found to be negligible compared to the 10 ms (100 Hz) cycle time of the UDP-synchronized line.

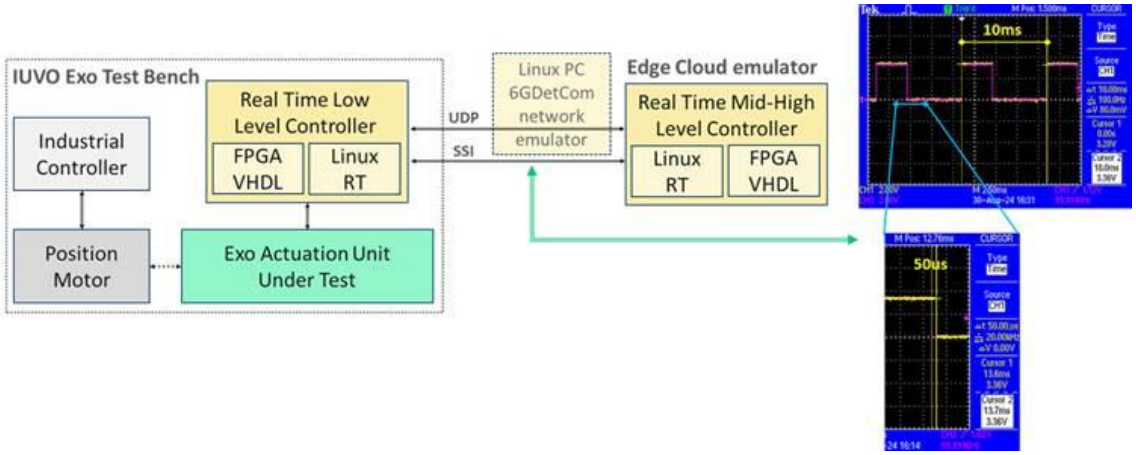


Figure 3-35: Exoskeleton use case validation, software integrated architecture and SSI line measurement.

Both the AU and the edge cloud emulator RT processing units are monitored by a LabView-developed graphical user interface running on an additional ancillary host-PC, capable of displaying real-time data and specifying some control parameters.

The integration process required several iterative verification steps. Before enabling the 6GDetCom Network Delay Emulator, a model of the communication delay in the integrated software architecture was developed to extract useful information about UDP line delays using the SSI-shared counter. This model is illustrated in Figure 3-36.

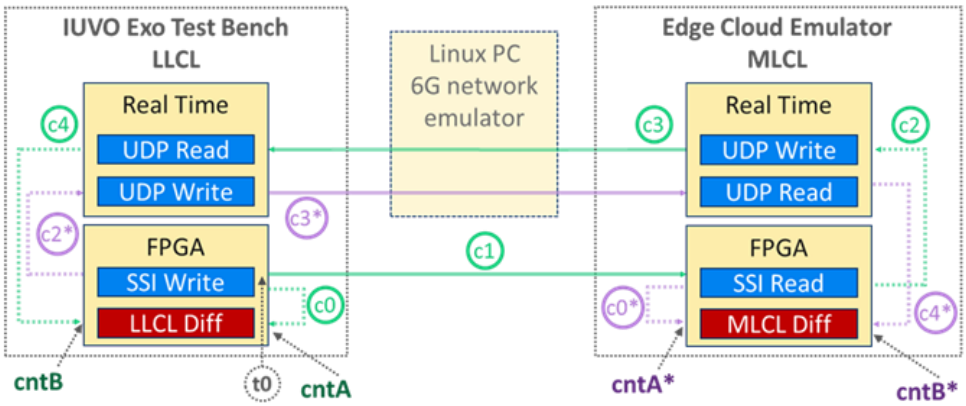


Figure 3-36: Exoskeleton use case validation, model of the communication delays in the integrated software architecture.

At time t_o, a counter is generated in the FPGA of the exoskeleton test bench and incremented every 10 ms (increment triggered by the SSI synchronization signal). The counter is:

- propagated inside the Exo test bench FPGA generating the counter namely “cntA”;
- shared over SSI line and sampled by the Edge Cloud Emulator FPGA as “cntA*”;
- propagated from the FPGA to the RT processor of both the exoskeleton test bench, the edge cloud emulator, and transmitted back over UDP to the RT processor and FPGA of the opposite unit, labelled as “cntB” and “cntB*”, respectively.

Each time delay contribution is evaluated and related to the differences between cntA and cntB, and cntA* and cntB*, respectively. At runtime the simple integer differentiation is computed as “n” and “n*” and stored inside the processing unit non-volatile memory. These results are useful during the postprocessing phase to have a fine estimation of the delay on the UDP lines, namely “c3” and “c3*”.

$$10 * n = 10 * (cntA - cntB) = -c0 + (c1 + c2 + c3 + c4) \rightarrow c3 = (10 * n - 4.15)\, ms$$

$$10 * n^{*} = 10 * \left(cntA^{*} - cntB^{*}\right) = -\left(c0^{*} + c1^{*}\right) + \left(c2^{*} + c3^{*} + c4^{*}\right) \rightarrow c3^{*} = \left(10 * n^{*} + 3.1\right) ms$$

Figure 3-37 reports the histograms for the distribution of cntA-cntB and cntA*-cntB* during a 30 min test. For communication delay of c3*, most of the packets (81.9 %) result in a negative difference (n* = −1), meaning that cntB* is updated faster than cntA*. This indicates that UDP is faster than the synchronous SSI line, which is used as a benchmark. For the communication delay of c3, the difference is always greater than 0, meaning that cntA is always updated before cntB. Most packets (75.4 %) result in a difference of 1, and the delay introduced by the UDP line – based on the formula described above – is calculated as c3 = 5.85 ms.

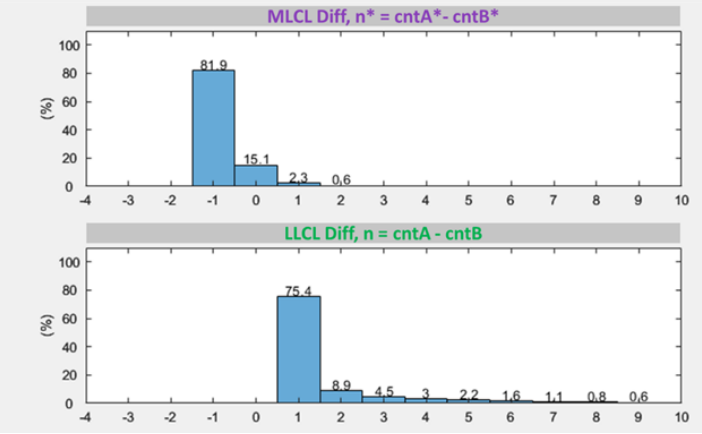


Figure 3-37: Exoskeleton use case validation, model of the communication delays histogram for 30min test.

UDP delays below 10 ms are not expected to impact on the control scenario described. The asymmetric results between the communication directions can be attributed to differences in computational load and in how each RT processor uses its real-time window to handle non-deterministic UDP communication. MLC algorithms are heavier than the off-loaded LLC RT processor and this means less time is required and the c3 UDP communication delay is slower than c3*.

The presented integrated architecture, its verification, together with the described model are used during the validation test described in the following section.

3.6.3 Validation Results

Building on the relevant example described in D1.3 [D6G-D1.3], Section 2.2, where a worker wears an exoskeleton that supports moving, lifting and handling loads in the warehouse, the assisted walking task is selected for the emulation. Two different walking velocities, 2.5 km/h and 5 km/h, are emulated on the test bench using the same AU assistance profile shown in Figure 3-38, which provides ±7 Nm of peak desired torque at the hip joint. The selected assistance profile is representative of a typical worker (body weight: 75 kg, height: 175 cm) walking across the shop floor while carrying a previously lifted small box, with assistance aimed at reducing physical effort and mitigating the risk of injury over repetitive shifts.

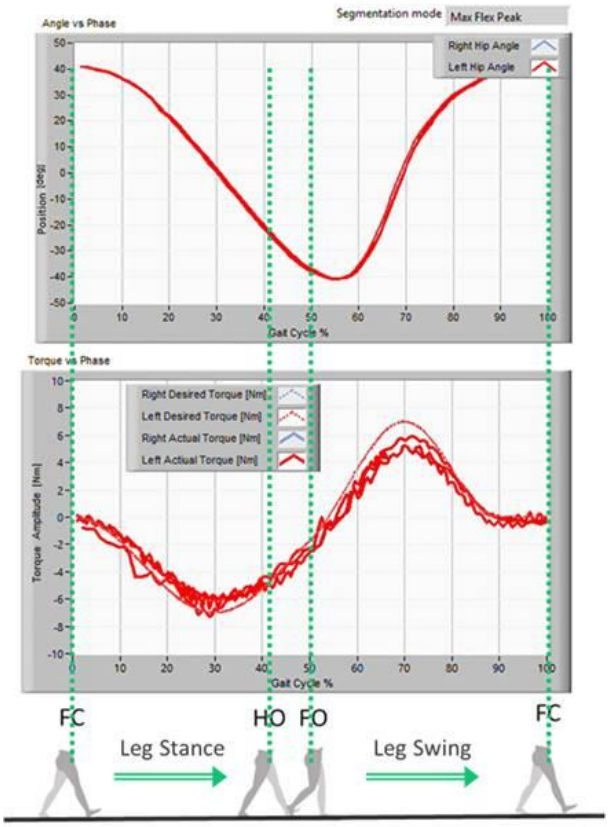


Figure 3-38: Exoskeleton use case validation, selected assistance profile during a test (velocity 2.5 km/h). The angle vs. phase chart shows the measured position of the AU's left joint during the last five steps (segmented and over imposed). The torque vs. phase chart displays the desired torque profile (dotted line) for the left leg along with the measured actual torque (continuous lines) over the same steps. The image highlights the main gait cycle events, starting from foot contact (FC) to foot off (FO), passing through heel off (HO) during the stance phase, and continuing through the swing phase until the next FC.

Two different types of joint torque controllers, each with specific gain settings, are used in the LLC to better follow the assistance profile at the two walking velocities. A smart algorithm capable of detecting the walking phase and tracking the movement is used in the MLC [YPG+17].

To establish reference baselines and evaluate the impact of the 6G network emulation, three different acquisitions are performed for each walking velocity. The first uses the same AU and test bench in a fully embedded architecture. All the control layers LLC, MLC, and HLC are deployed in the AU RT processing unit, and the UDP line is shut down relying on AU system on chip deterministic communication AXI (Advanced eXtensible Interface) protocols [XIL12]. The second uses the

architecture described in the previous section, but with the 6GDetCom Network Delay Emulator disabled, and the Debian PC acting as a transparent UDP bridge. The third employs the full hardware–software integrated architecture.

The parameters selected for the 6GDetCom Network Delay Emulator are the default values[6] (in particular, reordering of packets is allowed to closely follow the given delay distribution) and the imported delay distribution is the PD-Wireless-5G-2A which provides packet delay data of a wireless TSN bridge based on measurements from a 5G testbed in an industrial research shopfloor. The distribution is over imposed to the one already presented in Figure 3-2.

The first result presented relates to the time required by the phase detection algorithm to properly synchronize with the walking pattern. In the test bench, synchronization is manually activated using a "synch on" GUI button and during postprocessing the data should be aligned using the RT stored "enable synchronization" signal riding edge. Based on this event, MLC starts computing the acquired measured joint position delivered from the LLC, to detect properly the walking phase. Using exactly the same implementation for the algorithm and comparing the time to reach a synchronization detected with the same metrics, the following Table 3-1 shows the measurement results performing multiple consecutive acquisition at a walking speed of 2.5 km/h.

Architecture used for the test:	Phase detection algorithm, synchronization time (seconds):
HLC-MLC-LLC fully embedded	(14±2)s
HLC-MLC Off-loaded with 6GDetCom network emulator and Debian PC as transparent bridge	(16±2)s
HLC-MLC Off-loaded with 6GDetCom network emulator enabled	(18±4)s

Table 3-1: Exoskeleton use case validation, test at 2.5 km/h and evaluation of the time required to achieve gait phase detection synchronization.

The 14 seconds required by the fully embedded architecture follow the algorithm implemented in the test bench. Considering that each step takes approximately 2 seconds at the selected walking speed, the algorithm converges in 7 ± 1 steps. When the HLC and MLC are offloaded, but the 6GDetCom Network Delay Emulator is not enabled and the Debian PC acts as a transparent bridge, convergence is on average slower, requiring 8 ± 1 steps. Compared to the best case, 2 additional steps are needed on average to reach convergence when the network emulator is included in the architecture. In these final acquisitions, variability also increases in some trials, two extra steps are required to achieve synchronization, while in others this effect is negligible. We also published a video[7] showing these results.
Figure 3-39 shows the results of tracking the assistance torque profile. The torque controllers used are the same as those implemented in the final exoskeleton. However, due to the more rigid coupling of the test bench compared to a real human leg, the controller gain was reduced to maintain stability. As a result, the torque profile tracking performance is not optimal when the fully embedded architecture is considered as a baseline. Nevertheless, this architecture demonstrates low variability across consecutive steps, and the error between the desired and actual torque remains stable. On the other hand, the delay introduced by the MLC off-loaded architecture appears to improve the overall

[6] https://github.com/DETERMINISTIC6G/6GDetCom_Emulator
[7] https://ipvs.informatik.uni-stuttgart.de/vs/det6g/VID-PhaseDetRes.mp4

average error. However, it also increases variability between steps. Specifically, the variance increases from 1 % at an emulated walking velocity of 2.5 km/h to 25 % and from 2 % to 30 % at 5 km/h.

There are trials where, at a walking velocity of 5 km/h, a large error in the torque loop triggers the safety mechanism, which disables the motor and stops the delivery of the assistance profile. These results are also shown in a video[8].

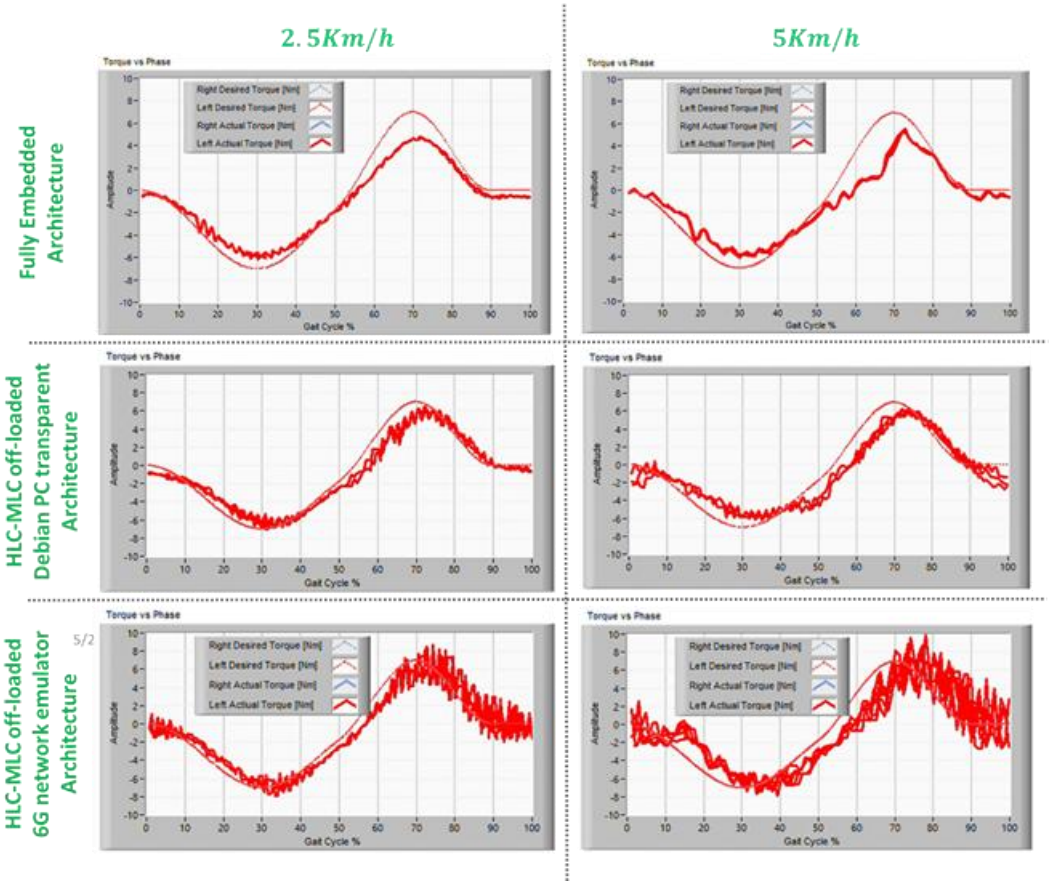


Figure 3-39: Exoskeleton use case validation, torque assistance profile degradation results.

3.6.4 Key Takeaways

This work described the development of an emulator designed to study novel paradigms for off-loading the control of an occupational exoskeleton and supporting the 6G wireless network development.

The observed degradation in the convergence time of the phase detection algorithms, along with reduced repeatability in tracking the assistance profiles, is particularly critical in real-world use cases involving a human in the loop – especially when the movement is non-periodic or when the high-level controller (HLC) detects a task transition (e.g., from walking to lifting).

The presented emulation system should be used in future studies to further investigate movements performed at different velocities, as well as other types of movements that involve varying assistance profiles. Additional tests using the 6GDetCom Network Delay Emulator – using PD-Wireless-5G-2a and considering Packet Delay Correction or with other distributions such as PD-Wireless-5G-3a, which

[8] https://ipvs.informatik.uni-stuttgart.de/vs/det6g/VID-AssistanceDegr.mp4

provides packet delay data for a wireless TSN bridge based on measurements from a 5G testbed implementing standardized ultra-reliable and low-latency communication in an industrial research shopfloor – will be conducted in the future. In general, to achieve the desired assistance specifications and deliver the intended benefits to the user, the described tool can serve as a powerful co-design verification platform for the exoskeleton control layers, algorithms, and communication network.

3.7 Adaptive Manufacturing Use Case Validation

3.7.1 Introduction

In this section, we present validation results derived from the adaptive manufacturing use case introduced in WP 1 (Deliverable D1.1 "DETERMINISTIC6G Use Cases and Architecture Principles" [D6G-D1.1]). We consider Automated Guided Vehicles (AGVs), moving on a shop floor to transport workpieces between machines. These AGVs are controlled through a networked control system (NCS) with a controller hosted in the edge cloud infrastructure of the factory. The AGVs send sensor values to the controller and receive actuator commands from the controller. Both, the uplink and downlink from and to the AGVs, are implemented with a wireless 5G/6G link of a TSN network (logical 5G/6G TSN bridge). We evaluate the influence of the characteristic PD of the wireless network onto the Quality of Control (QoC).

We first present the setup of the validation system and the key performance indicators (KPIs) used to evaluate the system performance, before we present the different validation results.

3.7.2 Validation Methodology and Setup

As methodology for the validation, we use simulations of the network and the NCS. The system to be simulated is shown in Figure 3-40. It consists of the following components:

1. **AGVs** moving on the shop floor for transporting workpieces between machines. AGVs are equipped with **sensors** and **actuators**. From the perspective of the NCS, an AGV is the **plant** with input and output signals. The plant is modelled in the simulation as an application, which is connected through a UE to the Device-Side TSN-Translator (DS-TT) of a logical 5G/6G TSN bridge described next.
2. A **logical 5G/6G TSN bridge**, with a wireless link to each AGV application. Since we simulate this logical bridge with the DETERMINISTIC6G DetCom Node simulation model [D6G-D4.1], we call this bridge **DetCom Node**. In particular, the uplink and downlink of the DetCom Node follow the characteristic PD as measured in experiments in real 5G networks (below we will give an overview of the different PD distributions used in the validation).
3. The controller of the NCS (called **NCS Controller**) hosted on a host in the **edge cloud**. In the simulation, the host is connected to the DetCom Node through a wireline TSN bridge connected to the Network TSN-Translator (NW-TT) of the DetCom Node. In the simulation, the NCS Controller is modelled as an application with processing delay as further defined below.

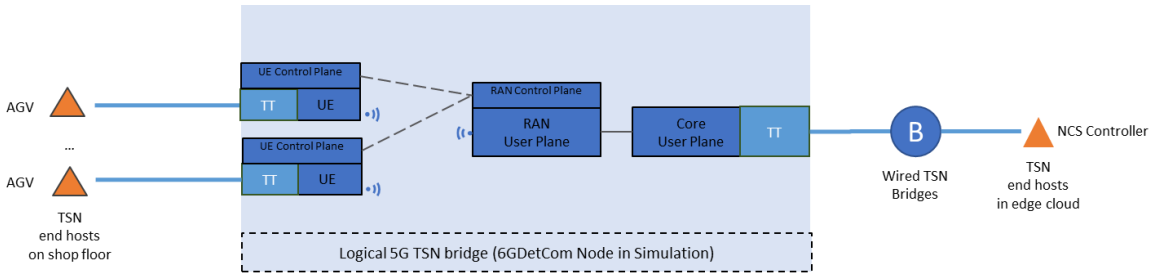


Figure 3-40: Simulation topology with wireless bridge

As a baseline, we use a wireline topology as shown in Figure 3-41 with the wireless logical bridge (6GDetCom node) replaced by a wireline TSN bridge. Note that this topology only serves to compare the performance of a wireline vs. a wireless system with different characteristic port-to-port delays. Obviously, the wireline topology is impractical for controlling mobile AGVs.

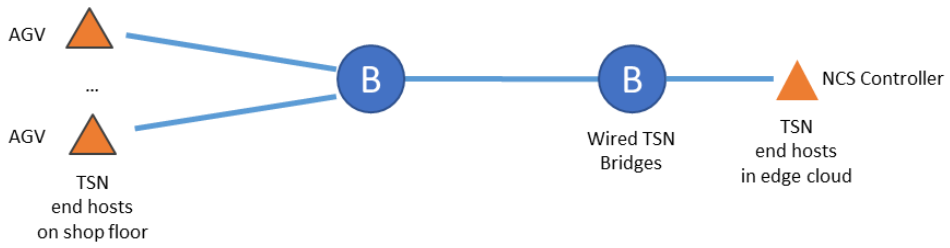


Figure 3-41: Simulation topology with wireline bridge (for comparison)

The PD distributions of the DetCom Node and processing delay distribution of the edge cloud host have been described in D4.1 [D6G-D4.1] in more detail. Section 2.3 of this report also gives a short overview of the measurement framework and the measured delay distributions. We use the following distributions, following the naming scheme in the open data repository:

- **PD-Wireless-5G-1**[9]: The dataset contains the data collected during the latency measurements performed on a "Commercial off the Shelf" (COTS) 5G system. The 5G network operates in band 78, in Time Division Duplex (TDD) mode, with a total of 106 Physical Resource Blocks (PRBs) which occupies 40 MHz of bandwidth.
- **PD-Wireless-5G-2a**[10]: Packet delay (PD) of a wireless TSN bridge based on measurements from a 5G testbed in an industrial research shopfloor. The 5G testbed corresponds to a 5G standalone trial network with 100 MHz carrier bandwidth operating in the 3.7 GHz band.
- **PD-Wireless-5G-3a**[11]: The data set described the packet delay of a wireless TSN bridge based on measurements from a 5G testbed in an industrial research shopfloor. The 5G testbed corresponds to a pre-commercial 5G URLLC standalone prototype network with 100 MHz carrier bandwidth operating in the 28 GHz band. It implements 5G standardized functionality for ultra-reliable and low latency communication

[9] https://github.com/DETERMINISTIC6G/deterministic6g_data/tree/main/PD-Wireless-5G-1
[10] https://github.com/DETERMINISTIC6G/deterministic6g_data/tree/main/PD-Wireless-5G-2a
[11] https://github.com/DETERMINISTIC6G/deterministic6g_data/tree/main/PD-Wireless-5G-3a

- **Processing delay**[12]: Processing delay according response time measurements of a cloudified application, considering the effect of the different delay components of the cloud execution environment, e.g., the application processing time, operating system, virtualization ecosystem, type of scheduling.

For our evaluation, we also need to define the NCS and the plant in more technical detail. The idea is that AGVs perform challenging transport tasks to move workpieces between machines. Examples of such challenging transport tasks could be a liquid transported in an open container on the AGV that needs to be transported without spilling over when the AGV accelerates, or a tall workpiece that must not tip over while the AGV moves. Another example is two AGVs transporting a long workpiece together such that their distance remains constant and the workpiece is not damaged by pushing or pulling on the workpiece. Obviously, due to the complexity of such real systems, it is hard to model the complete physical system realistically in the simulation. Therefore, we use a well-understood classic example from control theory instead that represents a challenging control task similar to the real tasks mentioned before: an inverted pendulum.

The inverted pendulum is a pendulum mounted inversely on a cart as shown in Figure 3-42. Obviously, the pendulum is instable and will tip over without control input. In our scenarios, it represents the workpiece that must not tip over while being transported or the liquid that must not spill over – in this respect, the pendulum is even more challenging to control than the real systems mentioned before since the liquid and tall workpiece only become "unstable" when the AGV starts accelerating. The cart can be accelerated by an external force F (control input), and the resulting torque and angular acceleration of the pendulum are used to bring the pendulum into the upright position. In the real examples above, the cart represents the AGV, although only moving into one dimension. Besides only controlling the angle of the pendulum, we can also control the position of the cart in addition to simulate the movement of the AGV moving along a given route. By considering the position of the cart, we can also represent the scenario of two AGVs transporting a workpiece together.

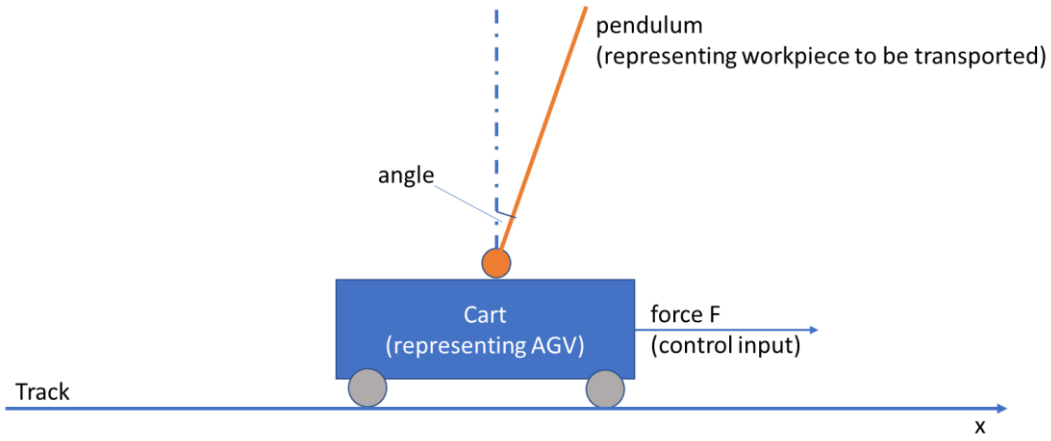


Figure 3-42: Inverted pendulum on cart

In summary, we consider two scenarios:

[12] https://github.com/DETERMINISTIC6G/deterministic6g_data/tree/main/ProcessingDelayDistribution2

Scenario 1: A single AGV transporting a workpiece. This scenario is modelled by an inverted pendulum where only the angle needs to be controlled to be (ideally) at the setpoint zero.

Scenario 2: Two AGVs transporting a workpiece together. This scenario is modelled by two inverted pendulums moving on two individual "virtually coupled" carts that must (ideally) stay within a certain constant distance while moving along a given route, without letting the pendulums tip over.

To allow for reproducing our results, we also provide the equations of motion and parameters of the simulated inverted pendulum as used in our evaluation. For the sake of simplicity, we do not model the friction of the cart and bearing of the pendulum, i.e., the system is not dampened, which will lead to some overshooting of the setpoint:

$$\dot{x} = v$$

$$v = \frac{-ml\sin\theta\,\omega^2 + (m^2gl^2\sin(\theta)\cos(\theta))/(I+ml^2) + F}{(M+m) - (m^2l^2cos^2(\theta))/(I+ml^2)}$$

$$\dot{\theta} = \omega$$

$$\omega = \frac{-ml^2\sin(\theta)\cos(\theta)\omega^2 + (M+m)gl\sin(\theta) + l\cos(\theta)F}{(I+ml^2)\left(\frac{M+m}{m}\right) - ml^2cos^2(\theta)}$$

Parameter	Value
Mass of pendulum: m	0.2 kg
Mass or cart: M	0.5 kg
Moment of Inertia: I	0.006 kg m^2
Length of pendulum to center of mass: l	0.3 m
Gravitational acceleration: g	9.81 m/s^2

As NCS controllers, we use Proportional-Integral-Derivative (PID) controllers and Linear Quadratic Regulators (LQR). Note that it is not our goal to find the optimal controller, neither do we claim that our controllers are optimal. In principle, one can try to solve the problem of large PD and PDV of wireless networks on the application layer through better controllers that are robust to PD and PDV, or by improving the real-time properties (QoS) of the network such as reducing deadline violations through wireless-aware schedules or reducing PDV through PDC. Our goal is the latter, i.e., we want to show that the network properties (QoS of network) might influence the application performance (QoC of application). In practice, application-layer mechanisms and network mechanisms could be combined.

In detail, we use the following controllers with the following parameters to control either only the angle of the pendulum (Scenario 1), or pendulum angle and position of the cart (Scenario 2), as also shown in Figure 3-43, Figure 3-44, and Figure 3-45:

- **PID-Angle** only controlling pendulum angle: P = 10.0, I = 1.0, D = 1.0
- **PID-Angle/Position** controlling pendulum angle and cart (AGV) position: This controller is implemented by three sequential PID controllers:
  - PID1 controls the position by adjusting the speed setpoint of the cart: P = 1.0; I = 0.0; D = 1.0. The output of this controller is clamped to v_{set} = 2.5 m/s.

- PID2 controls the speed by adjusting the pendulum angle setpoint. For instance, if the pendulum leans to the left (positive θ), the cart must accelerate to the left to keep the angle, thus, increasing the speed into the negative x direction:
  P = 0.06; I = 0.0; D = 0.0. The output of this controller is clamped to $\theta_{set} = \pm 20$ degrees.
- PID3 controls the angle of the pendulum:
  P = 10.0; I = 0.0; D = 1.0

- **LQR-Angle** only controlling pendulum angle:
  gain matrix K = [-1.0000000000001679, -2.7126628569811633, 42.94618303488281, 5.411763498735041]
- **LQR-Angle/Position** controlling pendulum angle and cart (AGV) position:
  gain matrix K = [-3.162277660168483, -6.105688949485788, 49.16351188321586, 7.204143097154165]

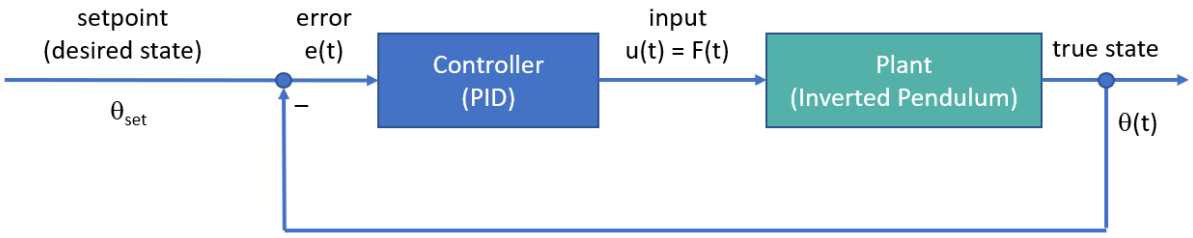


Figure 3-43: Control loop for angle control with PID controller (PID-Angle)

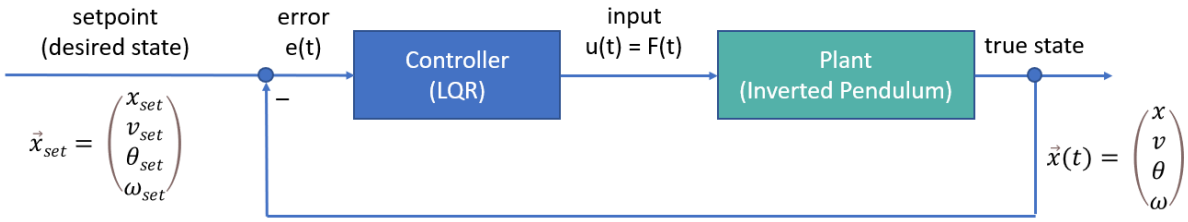


Figure 3-44: Control loop for angle and/or position control with LQR (LQR-Angle, LQR-Angle/Position). Note that always the full state is fed back. Therefore, the same structure applies to angle control and to angle & position control.

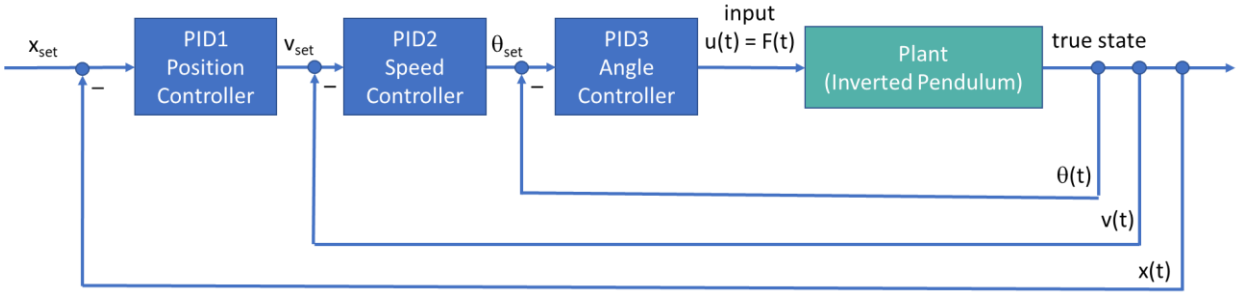


Figure 3-45: Control loops for angle and position control with PID controllers (PID-Angle/Position)

The parameters of the PID controllers have been tuned manually.

The LQR has been optimized using the following state-space equations, where $\vec{x} = \begin{pmatrix} x \\ v \\ \theta \\ \omega \end{pmatrix}$ denotes the state vector of the system:

$$\dot{\vec{x}} = A\vec{x} + B\vec{u}$$

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{m^2 g l^2}{I(M+m)+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{mgl(M+m)}{I(M+m)+Mml^2} & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Mml^2} \\ 0 \\ \frac{ml}{I(M+m)+Mml^2} \end{pmatrix}$$

The following matrices Q have been used to define the “cost” of position, speed, angle, and angular velocity; the R matrix defines the “cost” of the input u (force). All costs are considered in the optimization of the regulator to calculate the gain matrix K, which is then multiplied with the state vector to get the force depending on the system state (a script to automate this process of finding the optimal gain matrix K is included with the source code):

Q and R for LQR-Angle:

$$Q = \begin{pmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$R = (0.01)$$

Q and R for LQR-Angle/Position:

$$Q = \begin{pmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$R = (0.01)$$

The state of the plant (AGV with pendulum) is sampled periodically with a period of 1 ms if not mentioned otherwise. The samples are sent to the remote NCS Controller via the DetCom Node, which induces the characteristic upstream 5G/6G packet delay. The NCS Controller calculates the new input u (force F onto cart) and sends it back via the DetCom Node to the AGV, again inducing characteristic downstream packet delay.

We implemented a co-simulation approach by splitting the simulation into two parts: the network simulation and the control system simulation including the physics simulation of the pendulum, as shown in Figure 3-46:

1. **Network simulation:** packets are generated to simulate the periodic transmission of samples from the AGV (talker) to the NCS Controller (listener), and back from the NCS Controller (talker) to the AGV (listener). For all packets, timestamps are recorded to a trace file, recording the time when the state of the plant is sampled, when the NCS Controller calculates the new input u, and when the input arrives at the AGV. This network simulation is executed by OMNeT++/INET with the DETERMINISTIC6G extensions.

2. **Control system/physics simulation:** To simulate the control system, a simple event-driven simulator has been implemented including simulation models for the plant (physical system) and the NCS Controller as described above.[13] The (physical) state of the plant (position, speed, angle, angular velocity of cart and pendulum) is sampled according to the timestamps of egress packets from the plant (AGV) recorded in the packet trace of the network simulation, i.e., periodically with a sampling period of 1 ms (if not stated otherwise). The output of the NCS Controller (input *u* to the plant) is calculated and sent according to the timestamps of egress packets from the NCS controller as recorder in the packet trace, i.e., after an E2E network delay from the plant to the NCS Controller and processing delay of the NCS Controller. The input is applied to the plant at the ingress time of packets to the plant according to the timestamps in the packet trace, i.e., after another E2E network delay. The physical simulation of the cart and pendulum is executed periodically with a period of 100 µs (not to be confused with the sampling period of the control system; the pendulum is simulated more often to ensure an accurate simulation of the motion of the pendulum). The state of the pendulum (position, speed, angle, angular velocity) is recorded to a file with timestamps for later evaluation.

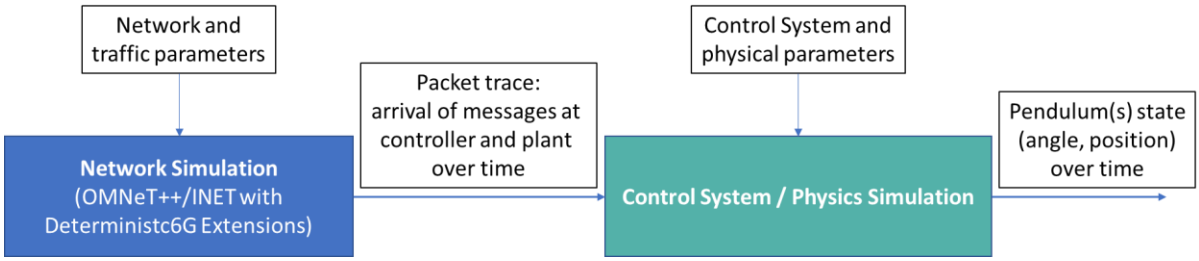


Figure 3-46: Co-simulation of network and control system incl. pendulum physics

3.7.3 Key Performance Indicators

We are interested in the influence of the network QoS (PD and PDV) onto the QoC of the NCS. We express QoC by the following Key Performance Indicators (KPI):

Settling time of the pendulum**:** In Scenario 1, we are only interested in the angle of the pendulum. We start the simulation with an initial angular error of 20 degree (corresponding to a step input) at simulation time t = 0, and record the angle of the pendulum over time (no further disturbances are introduced during simulation time in Scenario 1). We define the settling time as the time that it takes to reach a certain error band that the pendulum does not leave anymore. The error band is defined as $\pm$ 0.5 degrees around the angle setpoint (0 degrees).

[13] https://github.com/DETERMINISTIC6G/InvertedPendulumSimulator.git

Mean Squared Error (MSE) of the distance between two AGVs: In Scenario 2, two AGVs move together to transport a large workpiece, which is resting on both AGVs. It is crucial that both AGVs move at a constant distance *d* along a predefined path. To this end, the NCS Controller is controlling the position of both AGVs (cart of pendulum), in addition to the pendulum angle (pendulum should not tip over). Since the cart of our inverted pendulum only moves in one dimension, we only consider the x coordinate of the position. If the position setpoint of the tandem team of AGVs is at position x, AGV 1 should be at position x – *d*/2 and AGV 2 at position x + *d*/2. We define the difference between the actual AGV distance and *d* as the distance error. We sample the distance error of the AGVs every 0.1 s. The MSE is the mean squared distance error of all distance error samples over the simulation time (60 s).

3.7.4 Results

Influence of Characteristic Packet Delay onto Quality of Control

We start by evaluating the influence of the characteristic packet delay and delay variation of the wireless 5G/6G network (port-to-port delay of logical 5G/6G TSN bridge) onto the QoC in Scenario 1 where only the angle of the inverted pendulum is controlled. We exclude the influence of cross-traffic by only having a single control system with a single AGV in the network, i.e., without congestion – to isolate different streams of multiple control systems, we can use the wireless-aware traffic engineering methods also developed in the project. Thus, network delay is dominated by the characteristic port-to-port delay of the 5G/6G TSN bridge as measured in our 5G testbeds.

As defined above, the pendulum has an initial angle error of 20 degree, and the KPI is the settling time to reach an error band of ± 0.5 degree.

Figure 3-47 and Figure 3-48 show the pendulum angle over the simulation time and the settling times for the wireline topology with only wireline bridges (Figure 3-47) and the wireless topology with the 6GDetCom node (Figure 3-48), respectively, for one characteristic PD distribution (PD-Wireless-5G-1) and both controller types (PID-Angle and LQR-Angle).

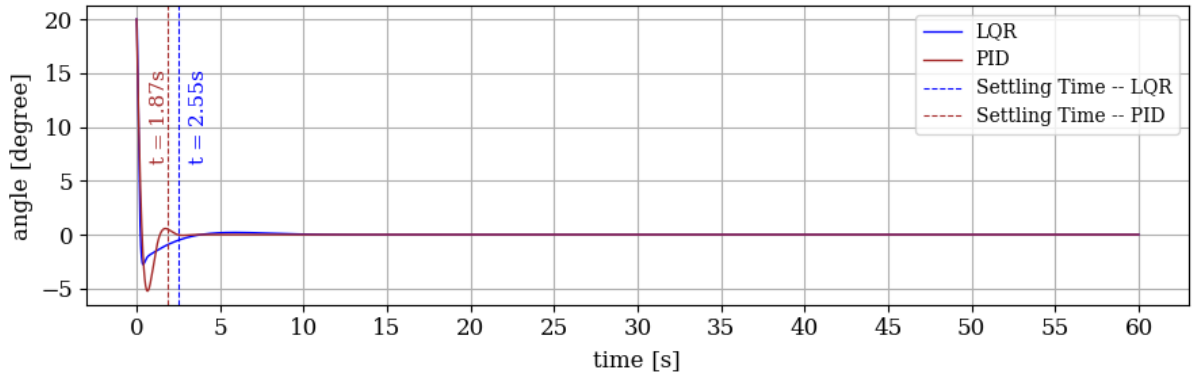


Figure 3-47: Pendulum angle over time – wireline topology

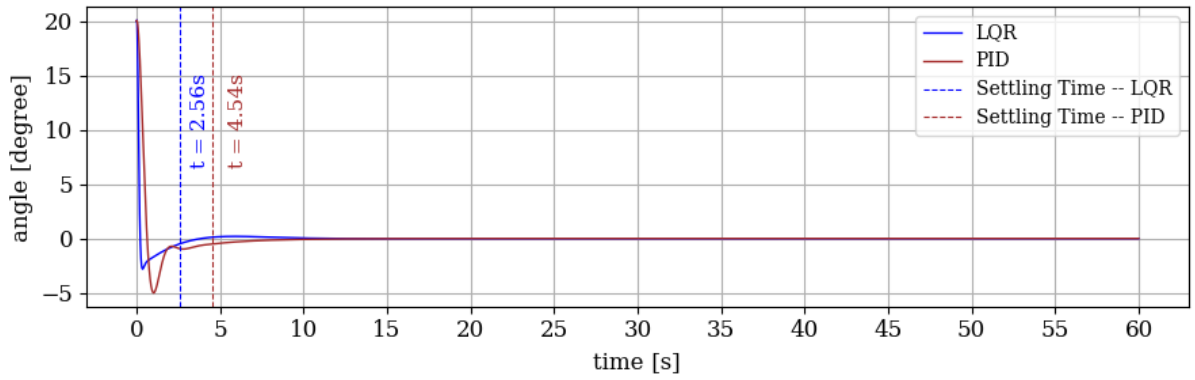


Figure 3-48: Pendulum angle over time – wireless topology, PD-Wireless-5G-1 distribution

First of all, the pendulum remains stable for both, the wireline and wireless system. For the PID controller, the settling time increases from 1.87 s to 4.54 s when switching from the wireline to the wireless system setup. Interestingly, the settling time remains constant for the LQR for all of our realistic delay distributions. This almost constant performance for the LQR did hold up to a threshold of about 25 ms mean PD (single trip) of an artificial normal distribution, where the performance rapidly decreased and the pendulum tipped over at about 35 ms PD.

A similar trend can be observed for the other realistic wireless delay distributions (although the settling time of the PID controller is slightly smaller for the distributions PD-Wireless-5G-2a and PD-Wireless-5G-3a) and also for artificial normal distributions with different mean μ and standard deviation σ. All settling times for the wireline topology, different realistic and artificial wireless distributions, and different controllers are listed in Table 3-2 (since the delay distributions are random functions, we report the 99 % confidence intervals of the median of the settling times of 30 simulation runs for each experiment).

Distributions	LQR	PID
Wireless PD Distributions (Histograms from Measurements)		
PD-Wireless-5G-1	[2.5596; 2.5717]	[3.0029; 4.9604]
PD-Wireless-5G-2a	[2.5645; 2.573]	[2.1509; 4.9606]
PD-Wireless-5G-3a	[2.5487; 2.5514]	[1.5552; 2.2498]
wireline	[2.5482; 2.5492]	[1.7152; 2.155]
Artificial PD Distributions (Normal Distributions)		
μ = 5 ms , σ = 5 ms	[2.5575; 2.5839]	[4.8839; 5.0042]
μ = 10 ms , σ = 2 ms	[2.5732; 2.5941]	[4.9281; 5.0503]
μ = 10 ms , σ = 5 ms	[2.5396; 2.6073]	[4.8339; 5.1758]
μ = 10 ms , σ = 20 ms	[2.5288; 2.625]	[4.7105; 5.0962]
μ = 10 ms , σ = 40 ms	[2.5733; 2.6752]	[4.8891; 5.2677]
μ = 5 ms , σ = 10 ms	[2.5274; 2.5882]	[4.825; 5.0475]
μ = 1 0ms , σ = 10 ms	[2.5568; 2.5999]	[4.9213; 5.0893]
μ = 20 ms , σ = 10 ms	[2.5465; 2.6525]	[4.8819; 5.1624]
μ = 25 ms , σ = 2 ms	[2.7161; 2.9649]	tipped over
μ = 40 ms , σ = 10 ms	tipped over	tipped over

Table 3-2: Settling Times [s] – 99 % confidence intervals of the median

With these results, we can make the following observations:

- For our test system, the pendulum can be stabilized under realistic wireless PD distributions.
- The characteristic PD of wireless systems decreases the QoC for some controllers (PID controller in our case) compared to a wireline system with smaller PD.
- Selecting an appropriate controller, such as the LQR in our example, for the wireless system – i.e. optimizing on the application layer – is a viable option, complementary to improving the QoS of the network. A cross-layer optimization approach jointly optimizing the control system (application) and network (QoS) would be a valuable extension (beyond the scope of this work).

Influence of Characteristic Packet Delay onto Synchronous Movement of Tandem AGVs

So far, we have considered only the impact of characteristic packet delay onto a single NCS. However, it is also interesting to see how a system of multiple NCS behaves, where different NCS need to be synchronized. Moreover, the following scenario takes advantage of the notion of a time-aware system with synchronized clocks of AGVs and NCS controllers as further described below.

To this end, we evaluate Scenario 2 with a tandem team of two AGVs following each other at a given distance d = 1 m to transport a large workpiece as shown in Figure 3-49. Note that in our simulation, both AGVs and the workpiece are modelled as two inverted pendulums on two carts. Besides controlling the position of the carts such that they move ideally at a constant distance of d = 1 m, a secondary goal is to move each cart such that the instable pendulums do not tip over to simulate that the workpiece must be transported very carefully.

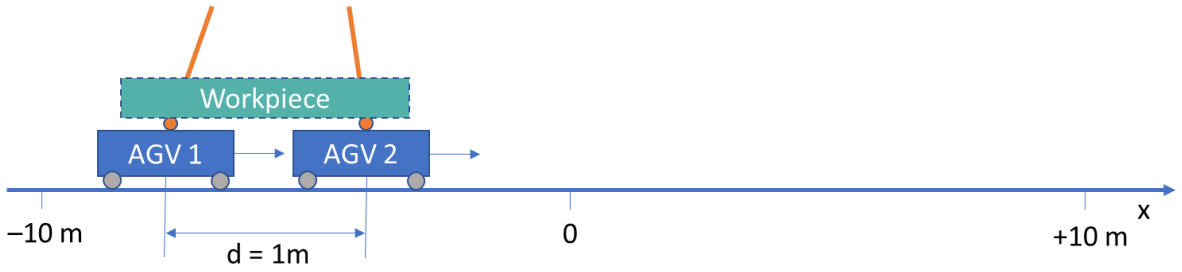


Figure 3-49: Two tandem AGVs transporting a workpiece

Since the physics simulation of the pendulum is constrained to one dimension, we define the position x over time as function x(t). The setpoints of AGV 1 and AGV 2 are x(t) – d/2 and x(t) + d/2, respectively. For a challenging setup with continuous acceleration, we define x(t) as:

$$x(t) = 10 \sin(\omega t)$$

Thus, the center position between the AGVs is between $\pm$ 10 m, and the speed is constantly changing. Consequently, the target speed of the AGVs is defined as:

$$\dot{x}(t) = 10\ \omega \cos(\omega t) = v(t)$$

We use three different **modes of operation** corresponding to different maximum speeds of the AGVs:

- Slow speed: v_{slow} = 0.5 m/s = 1.8 km/h
- Medium speed: v_{med} = 1 m/s = 3.6 km/h
- High speed: v_{high} = 2 m/s = 7.2 km/h

For instance, mode “slow speed” might be active when human personnel are in some areas crossing the tracks of the AGVs on the shop floor, whereas the AGVs can move in the high-speed regime in areas without such unexpected obstacles. The maximum speed defines parameter ω in the above equations. Obviously, the speed is maximum at time t = 0 where the sine function of position x has the greatest slope. According to the above equation, $\dot{x}(0) = 10\omega$. Thus, $v_{slow/med/high} = 10\ \omega_{slow/med/high}$.

The position of each AGV is controlled by individual position&angle controllers (PID-Angle/Position or LQR-Angle/Position as defined above). To realize this control system, we take advantage of synchronized clocks. Figure 3-50 depicts the sequence of actions for one cycle of both control loops. First of all, the clocks of the two controllers of AGV1 and AGV2 are synchronized such that both are aware of the reference position of the tandem team x(t) with respect to the same time base to calculate the setpoint position of their individual AGVs x(t) – d/2 and x(t) + d/2, respectively. Moreover, the clocks of the AGVs are also synchronized with the clocks of the controllers. This enables the controllers to accurately calculate the position error at the correct time, namely the time when the position was actually sampled by the AGVs (t1 in Figure 3-50), rather than the time when the position sample was received – after a network delay – by the controller (t2 in Figure 3-50). This time awareness is particularly beneficial if the network delay between AGV and controller is larger as for the wireless communication network in our scenario. We could go even further and demand that both AGVs trigger sampling at the same time (t1 = t5 in Figure 3-50), or use prediction of the motion of the AGVs based on elapsed time. But since we are mainly interested in the effects of network delay rather than controller optimization, we keep the controller deliberately simple.

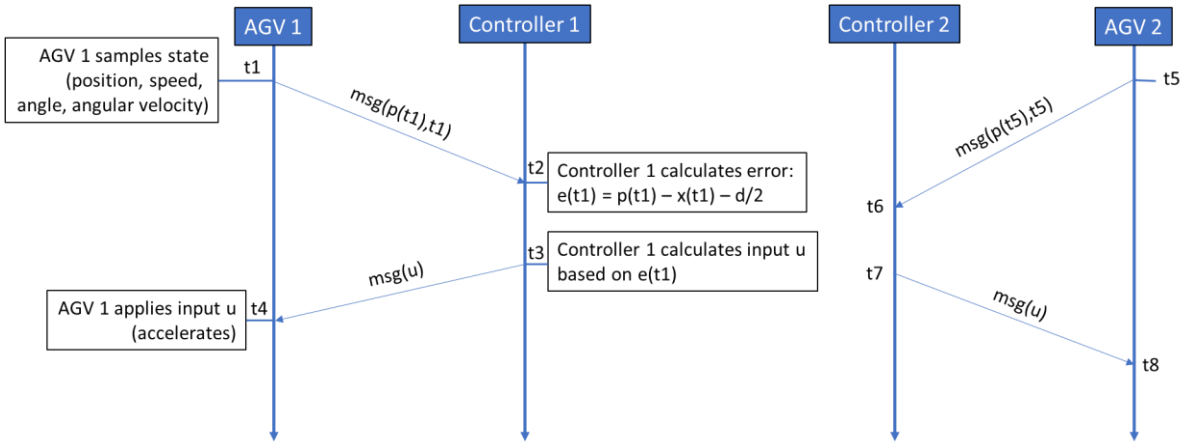


Figure 3-50: Sequence of actions of one control cycle

The network topology corresponds to Figure 3-40 (wireless system) and Figure 3-41 (wireline system for comparison).

Each simulation runs for 60 s simulated time. Moreover, at time t = 0, we introduce a small distance error (deviation from the setpoint distance of 1 m between AGVs) of 10 cm. The initial pole angle is at 0 degree.

Figure 3-51 shows the movement of both AGVs of the tandem team over time for the medium speed mode, PID controllers and delay distribution PD-Wireless-5G-3a. This figure just serves to verify that both AGVs approximately follow the given position according to the reference sine function x(t) (moving between $\pm$ 10 m) while trying to keep the given distance at d = 1 m.

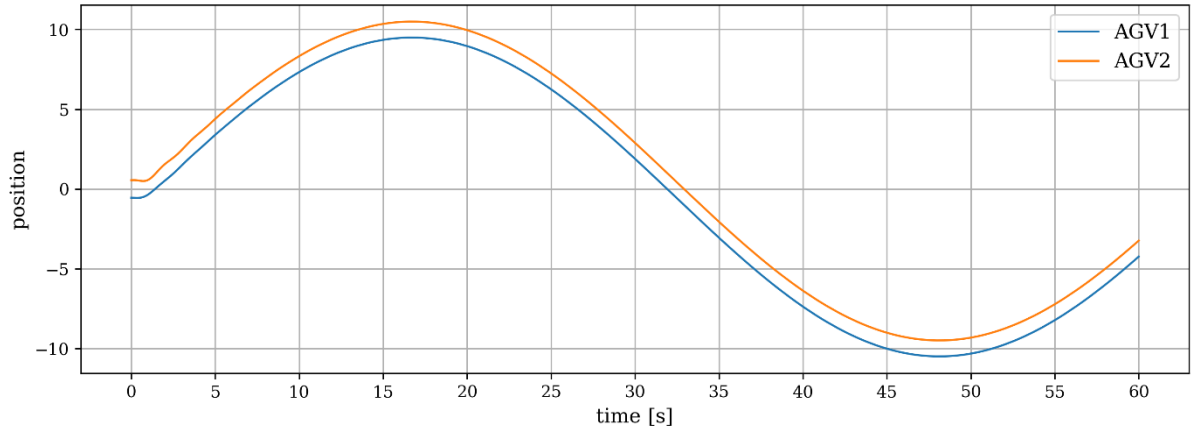


Figure 3-51: Position of tandem team of AGVs over time
(medium speed mode, PID-Position/Angle, PD-Wireless-5G-3a)

Figure 3-52 shows in more detail the distance error over time for both controller types (PID and LQR), medium speed mode, and wireline and wireless delay distributions (PD-Wireless-5G-3a). First of all, we see that both controllers are able to correct the initial distance error over time, in the wireline network as well as in the wireless network. Also the pendulums did not tip over in these experiments (not shown in the figure). Moreover, despite the required continuous acceleration over time to follow the reference position (sine function), the distance error is very small in all cases after the initial error has been corrected.

The LQR has again almost the same performance for wireline and wireless PD (both curves overlap in the figure), which is consistent with the observed behavior for the angle(-only) control above.

The PID controller is much more affected by PD and shows substantially different performance for wireline and wireless PD. In fact, the PID controller could not prevent the pendulum from tipping over anymore for the other two wireless PD distributions (PD-Wireless-5G-1, PD-Wireless-5G-2a). By clamping the maximum speed (cf. v_{set} in Figure 3-45) to 1.0 m/s, the PID controller was able to stabilize the pendulum again, but at the cost of not being able to follow the position in medium-speed and high-speed modes. Although this is also partially due to the non-optimal manual parametrization of the PIDs controllers, this shows again that the PD can have significant influence onto the performance and required parametrization of some controllers.

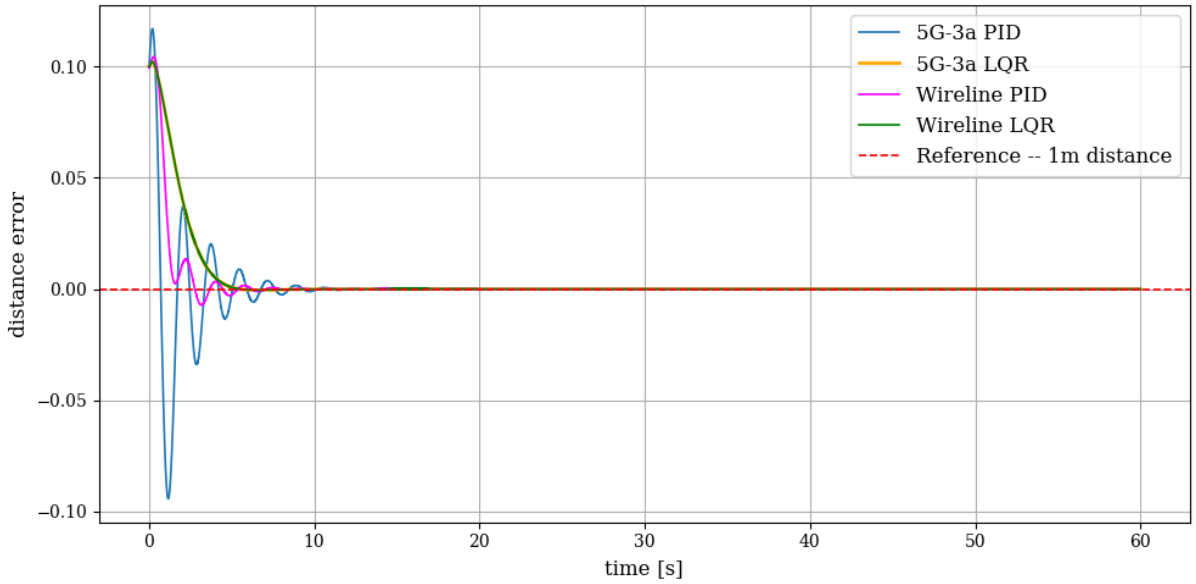


Figure 3-52: Distance error over time for LQR and PID controller and with wireline PD distribution and wireless distribution (PD-Wireless-5G-3a). Note that the orange curve of the LQR regulator with wireless PD distribution is hidden by the green LQR curve for the wireline PD distribution (same performance of the LQR for wireline and wireless system)

Table 3-3 shows the MSE and RMSE of the distance for all setups (all delay distributions, controller types, modes of operation). For the PID controllers, we also used versions with different clamping values of v_{set} due to the problems of the PID controller mentioned above (i.e. pendulum not stable at high v_{set} values, large distance error at low v_{set} values). These results underline the already mentioned observations: LQR performance is constant over a large range of wireline and wireless PD distributions and able to follow the position closely while keeping the pendulum stable for all realistic wireless delay distributions. PID controller performance is significantly affected by PD; as the PID controller is not able to stabilize the pendulum at higher speeds or follow the reference position closely for realistic wireless PD distributions.

Distribution	MSE [m^2]			RMSE [cm]		
	PID		LQR	PID		LQR
	small v_clamp	large v_clamp		small v_clamp	large v_clamp	
Slow Speed Mode						
5g-1	0.000737173	tipped over	0.000233666	2.72	tipped over	1.53
5g-2a	0.000760795	tipped over	0.000233876	2.76	tipped over	1.53
5g-3a	0.000126596	0.000201712	0.000233865	1.13	1.42	1.53
wireline	0.000152325	0.000149898	0.000233887	1.23	1.22	1.53
Medium Speed Mode						
5g-1	0.000940305	tipped over	0.000233636	3.07	tipped over	1.53
5g-2a	0.000858074	tipped over	0.000233867	2.93	tipped over	1.53
5g-3a	0.000126578	0.000201853	0.000233843	1.13	1.42	1.53
wireline	0.000152025	0.000149937	0.000233866	1.23	1.22	1.53
High Speed Mode						
5g-1	0.006693371	tipped over	0.000233528	8.18	tipped over	1.53
5g-2a	0.006767515	tipped over	0.000233798	8.23	tipped over	1.53
5g-3a	0.000159436	0.000214018	0.000233751	1.26	1.46	1.53
wireline	0.000629598	0.000149806	0.000233775	2.51	1.22	1.53

Table 3-3: Distance error for all controller types, PD distributions, and speed modes.

3.7.5 Key Takeaways

Based on the presented results from this use-case evaluation, we can draw the following conclusions:

- For both of our test systems, the systems could be stabilized under realistic wireless PD distributions. Therefore, for our exemplary time-sensitive control systems, a wireless 5G/6G network is a viable option, especially attractive for mobile systems.
- The characteristic PD of wireless systems decreases the QoC for some controllers (PID controller in our case) compared to a wireline system with smaller PD. Therefore, further improving the QoS of future 6G networks with respect to latency will have positive impact onto the performance of such time-sensitive systems.
- Selecting an appropriate controller, such as the LQR in our example, for the wireless system – i.e. optimizing on the application layer – is a complementary option to improving the QoS of the network. A cross-layer optimization approach jointly optimizing the control system (application) and network (QoS) would be a valuable extension for future.
- Although our results show that in terms of end-to-end delay the system could be stabilized under realistic PD distributions, we also see that the delay margin until the system becomes unstable is in the range of 10s of milli-seconds only. Therefore, any further delay as added for instance by network congestion must be bounded. To this end, we have proposed wireless-aware traffic engineering methods that ensure bounded delay also in a network with many streams competing for network resources (see validation results in Section 3.1).
- Time-awareness, where components are synchronized to a common time base, has proven to be a useful concept to realize our second exemplary control system with multiple synchronized control systems. This exemplary control system design relies on the availability of a time-synchronization service. To ensure its availability, we have proposed hot-standby mechanisms in our project (see validation results in Section 3.3).

- The DETERMINISTIC6G validation framework including the network simulator with 6GDetNet extensions and the latency measurements from the real 5G testbed proved to be a powerful and useful tool to test and evaluate the system under test with realistic wireless networking conditions.

4 Conclusion

In this deliverable, we have presented the final validation results for the concepts and mechanisms developed in the DETERMINISTIC6G project.

As a first result, the DETERMINISTIC6G validation framework consisting of the 6GDetCom Network Simulator, the 6GDetCom Network Delay Emulator, a latency measurement framework to gather characteristic 5G/6G packet delay data, and an emulation framework dedicated to security evaluations has proven to be very effective in validating our concepts for dependable 6G communication. The 6GDetCom Simulator together with realistic latency measurement data from the measurement framework offers a unique tool that, for the first time, enables the systematic evaluation of the impact of characteristic 5G/6G packet delay in TSN networks with wireless and wireline TSN bridges and end systems. Using realistic network delay and processing delay data, gathered in an edge cloud environment, we are also able to evaluate the end-to-end performance “over-the-loop” from sensor, to controller, to actuator in time-sensitive networked control systems. The 6GDetCom Network Delay Emulator enabled the performance evaluation of real applications – in our case an exoskeleton – with realistic 5G/6G packet delay. The security emulation framework enabled the validation of our security-by-design approach targeting attacks on the Precision Time Protocol. Using this comprehensive validation framework, we have produced a number of interesting validation results, from which we can draw the following conclusions.

From our validation of wireless-aware traffic engineering and packet delay correction mechanisms, we can draw several conclusions. First of all, conventional approaches designed to calculate packet schedules for wireline TSN networks perform poorly with respect to scalability (number of schedulable flows) since they allocate large time slots for packets based on conservative min/max delay bounds. On the one hand, we could show that our wireless-aware scheduling approach based on explicit knowledge of packet delay distributions improves scalability while providing meaningful guarantees on delay. On the other hand, we showed that packet delay correction removing uncertainty in terms of packet delay variation is another effective approach to improve schedulability in wireless networks with large uncertainty (packet delay variation). Both approaches are orthogonal, yet when combined provide best performance.

With respect to measurements and characterization of RAN latencies, the EDAF tool allowed for delay decomposition in a running mobile network. We could show through a staged experimentation how different delay elements vary and contribute to the overall end-to-end delay, providing a novel level of explainability.

The validation of hot-standby mechanisms for time synchronization with the Precision Time Protocol showed that using hot-standby Grand Masters (GM) leads to better performance than the Best TimeTransmitter Clock Algorithm (BTCA) in terms of clock drift and out-of-sync time during failures. Another result is that static GM configurations with hot standby might lead to problems leaving parts of the network without GM. BTCA with dynamic GM selection avoids this problem, but with a delayed

response. Thus, we conclude that the choice between hot standby and BTCA should depend on the network topology and redundancy level.

The validation of security mechanisms was focused on time delay attacks. Our validation shows that our proposed monitoring framework, based on P4 and in-band network telemetry (INT), enables the accurate detection and localization of time-delay attacks even in challenging situations where the attacker introduces only subtle, asymmetric delays without modifying the content of packets. Therefore, we conclude that our software-defined security-by-design approach based on INT and real-time packet-level monitoring is effective to enhance the resilience of deterministic networking systems, without relying on specialized hardware.

Moreover, we were able to show, by using a Mobile Edge Cloud (MEC) prototype, that using MEC reduces delay variation compared to public cloud applications significantly. Through simulations and implementation-based results we showed that seamless support of 802.1Qbv traffic scheduling for cloudified applications can be achieved efficiently. Moreover, various traffic handling solutions can be applied in the virtualized networking of a cloud host deployment.

In our first use case validation, we studied the effects of offloading the control of an occupational exoskeleton to a remote server via a wireless network using our exoskeleton prototype and 6GDetCom Network Delay Emulator. In this system, we observed an impact of the characteristic packet delay onto the phase detection algorithm together with a reduced repeatability in tracking the assistance profile. In particular, this is critical in real-world use cases with humans in the loop, non-periodic movement, or task transitions (e.g., from walking to lifting). The results encourage further studies as part of future work like movements performed at different velocities, as well as other types of movements that involve varying assistance profiles.

Our second use case validation of an adaptive manufacturing scenario showed the impact of the characteristic packet delay (as measured with our latency measurement framework) of wireless 5G/6G networks onto different networked control systems (simulated with the 6GDetCom Simulator). This validation showed that it is possible to implement a control loop over a wireless network with realistic packet delay to stabilize the open-loop-unstable system. An increase in characteristic packet delay decreased the quality of control for some controllers, while others showed to be more robust to packet delay. Therefore, we conclude that, on the one hand, it is beneficial to further decrease the latency of wireless networks. On the other hand, optimizing the controller on the application-layer is a complementary approach to optimizing the end-to-end performance of the networked control system. Moreover, isolating streams in the network and avoiding congestion through wireless-aware traffic engineering techniques as discussed above are required since the investigated control systems showed a small delay margin before becoming unstable. We also showed in this use case how time-awareness facilitates the implementation of control systems relying on synchronized components. Such systems directly benefit from techniques ensuring high-availability of a time synchronization service, building, for instance, on our hot-standby techniques already mentioned above.

References

[3GPP-TS23501]	3GPP, "TS 23.501: Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS)", Release-19, v19.1.0, 2024
[D6G-D1.1]	D1.1, "DETERMINISTIC6G Use Cases and Architecture Principles", June 2023, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D1.1-v1.0.pdf
[D6G-D1.3]	D1.3, "Report on Dependable Service Design", December 2024, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D1.3-v1.0.pdf
[D6G-D1.4]	D1.4, "Final Report on DETERMINISTIC6G Architecture – A Dependable Network Architecture for 6G", June 2025
[D6G-D2.1]	D2.1, "First Report on 6G Centric Enabler", December 2023, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D2.1-v2.0.pdf
[D6G-D2.2]	D2.2, "First Report on the Time Synchronization for E2E Time Awareness", December 2023, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D2.2-v1.0.pdf
[D6G-D2.3]	D2.3, "Second Report on 6G Centric Enabler", April 2025
[D6G-D2.4]	D2.4, "Report on the Time Synchronization for E2E Time Awareness", April 2025, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D2.4-v1.0.pdf
[D6G-D3.1]	D3.1, "Report on 6G Convergence Enablers Towards Deterministic Communication Standards", December 2023, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D3.1-v1.0.pdf
[D6G-D3.2]	D3.2, "Report on the Security Solutions", December 2023, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D3.2-v1.1.pdf
[D6G-D3.3]	D3.3, "Report on Deterministic Edge Computing and Situational Awareness via Digital Twining Security Solution", June 2024
[D6G-D3.4]	D3.4, "Optimized Deterministic End-to-End Schedules for Dynamic Systems", June 2024, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D3.4-v1.0.pdf
[D6G-D3.5]	D3.4 "Report on Multi-Domain End to End Schedules", April 2025, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D3.5-v1.0.pdf
[D6G-D3.6]	D3.6, "Report on Deterministic Edge Computing and Situational Awareness via Digital Twining", April 2025, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D3.6-v1.0.pdf
[D6G-D4.1]	D4.1, "DETERMINISTIC6G DetCom Simulator Framework Release 1", December 2023,

	https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D4.1-v1.0.pdf
[D6G-D4.2]	D4.2, "Latency Measurement Framework", March 2024, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D4.2_v1.0.pdf
[D6G-D4.3]	D4.3, "Latency Measurement Data and Characterization of RAN Latency from Experimental Trials", April 2025, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D4.3-v1.0.pdf
[D6G-D4.4]	D4.4, "DETERMINISTIC6G DetCom Simulator Framework Release 2", April 2025, https://deterministic6g.eu/images/deliverables/DETERMINISTIC6G-D4.4-v1.0.pdf
[EGS+25]	S. Egger, J. Gross, J. Sachs, G. P. Sharma, C. Becker, F. Dürr, "End-to-End Reliability in Wireless IEEE 802.1Qbv Time-Sensitive Networks", IEEE/ACM International Symposium on Quality of Service (IWQoS 2025) (to appear), July 2025. DOI (preprint): 10.48550/arXiv.2502.11595.
[EXPU25]	ExPECA User Guide, https://expeca.proj.kth.se/start/, last accessed June 2025
[HDM+23]	L. Haug, F. Dürr, S. S. Mostafavi, G. P. Sharma, J. Sachs, J. Harmatos, J. Costa-Requena, and J. Ansari, "D4.1 – First DetCom Simulator Framework Release – Datasets," December 2023. DOI: 10.5281/zenodo.10405085
[IEEE1588-2019]	IEEE Standard 1588-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," June 2020. DOI: 10.1109/IEEESTD.2020.9120376
[IEEE15-8021Qbv]	IEEE Standard 802.1Qbv-2015, "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment 25: Enhancements for Scheduled Traffic", March 2016, DOI: 10.1109/IEEESTD.2016.8613095
[IEEE20-8021AS]	IEEE Standard 802.1AS-2020, "IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications", June 2020, DOI: 10.1109/IEEESTD.2020.9121845
[INE25]	INET Framework (website), https://inet.omnetpp.org/, last accessed June 2025
[LBD+20]	F. Lanotte, A. Baldoni, F. Dell' Agnello, A. Scalamogna, N. Mansi, L. Grazi, B. Chen, S. Crea, N. Vitiello, "Design and characterization of a multi-joint underactuated low-back exoskeleton for lifting tasks", 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)", October 2020, DOI: 10.1109/BioRob49111.2020.9224370
[OMN25]	OMNeT++ Discrete Event Simulator (website), https://omnetpp.org/, last accessed April 2025
[SOL+23]	T. Stüber, L. Osswald, S. Lindner, and M. Menth. "A Survey of Scheduling Algorithms for the Timeaware Shaper in Time-Sensitive Networking (TSN)", IEEE Access, 11, June 2023. DOI: 10.1109/ACCESS.2023.3286370.
[XIL12]	Xilinx, "AXI Reference Guide", UG761 (v14.3), November 15, 2012. https://docs.amd.com/v/u/en-US/ug761_axi_reference_guide, last accessed June 2025

[YPG+17]	T. Yan, A. Parri, V. Ruiz Garate, M. Cempini, R. Ronsse, N. Vitiello, “An Oscillator-Based Smooth Real-Time Estimate of Gait Phase for Wearable Robotics”, Springer Nature, Volume 41, 2017. DOI: 10.1007/s10514-016-9566-0

List of abbreviations

5G	Fifth Generation
5GS	5G System
6G	Sixth Generation
AC	Application Client
AMF	Access Mobility Function
AU	Actuation Unit
BTCA	Best time Transmitter clock algorithm
CCDF	Complementary Cumulative Distribution Function
CNC	Centralized Network Controller
CNI	Container Network Interface
CU	Centralized Unit
DetNet	Deterministic Networking
DoS	Denial of Service
DS-TT	Device Side TSN translator
DU	Distributed Unit
DVP	Delay Violation Probability
E2E	End-to-END
EDAF	End-to-End Data Analytics Framework
EAS	Edge Application Server
ECS	Edge Configuation Server
GCL	Gate Control List
GM	Grandmaster
GNSS	Global Navigation Satellite System
gNB	Next generation NodeB
gPTP	generalized Precision Time Protocol
IAT	Inter-Arrival Time
IDS	Intrusion Detection System
INT	In-band Network Telemetry
IRTT	Isochronous Round-Trip Tester
JSON	JavaScript Object Notation
HARQ	Hybrid Automated Repeat Request
HLA	High-Level Architecture
HLC	High-Level Controller
HTTP	HyperText Transfer Protocol
LLC	Low-Level Controller
MEC	Multiaccess Edge Computing
MLC	Mid-Level Controller
NEF	Network Exposure Function
NF	Network Function

NLMT	Network Latency Measurement Tool
NRF	Network Repository Function
NWDAF	Network Analytics Function
NW-TT	Network side TSN translator
OAI	OpenAirInterface
OC	Ordinaire Clock
OVS	Open Virtual Switch
P4	Programming Protocol-independent Packet Processor
PCF	Policy Control Function
PD	Packet Delay
PDC	Packet Delay Correction
PDU	Protocol Data Unit
PDV	Packet Delay Variation
PRB	Physical Resource Block
QoC	Quality of Control
RAN	Radio Access Network
RLC	Radio Link Control
RSRP	Reference Signal Receive Power
RSRQ	Reference Signal Receive Quality
PTP	Precision Time Protocol
RT	Real Time
RU	Radio Unit
SDR	Software-Defined Radio
SD-SEC	Software-Defined Security
SMD	Security Management Domain
SMF	Session Management Function
SSI	Serial Synchronous Interface
SSLA	Security Service Level Agreement
TAS	Time-Aware Shaper (IEEE 802.1Qbv)
TC	Transparent Clock
TDA	Time-Delay Attack
TDD	Time Division Duplex
TLV	Type-Length-Value
TSA	Time-Sensitive Application
TSN	Time-Sensitive Networking
TSN-AF	TSN Application Function
tR	time Receiver
TSCTSF	Time-Sensitive Communication and Time Synchronization Function
TSe	Egress timestamp

TSi	Ingress timestamp
TSN	Time-sensitive Networking
tT	time Transmitter
TT	TSN Translator
UDP	User Datagram Protocol
UP	User Plane
UPF	User Plane Function
URLLC	Ultra-Reliable Low Latency Communication