DETERMINISTIC6G

Report on the time synchronization for E2E time awareness

D2.4

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

Report on the time synchronization for E2E time awareness

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:	Report
Deliverable reference number:	D2.4
Contributing workpackages:	WP2, WP4
Dissemination level:	PUBLIC
Due date:	30-04-2025
Actual submission date:	30-04-2025
Responsible organization:	CMC
Editor(s):	Jose Costa-Requena
Version number:	1.0
Status:	Final
Short abstract:	This deliverable continues the work related to precise time synchronization within the context of sixth generation (6G) networks. The report includes the validation of hot-standby architectures using simulation in cooperation with WP4. It also explores the packet delay minimization techniques that can be applied to PTP traffic and evaluates the security framework's efficiency in cooperation with WP3. Moreover, the report analyses some options for improving time synchronization information using separate slices or allocating a different data flow with higher Quality of Service (QoS) to delivery time synchronization.
Keywords:	6G, TSN, DetNet, time synchronization, PTP, gPTP, Hot-standby

Contributor(s):	Huu Nghia Nguyen (MI) Mahin Ahmed (SAL) Lucas Haug (USTUTT) Jose Costa-Requena (CMC) Marilet De Andrade Jardim (EAB)

Reviewers:	Frank Dürr (USTUTT) Raheeb Muzaffar (SAL) Edgardo Montes de Oca (MI)

Revision History

V 0.1	Initial draft
V 0.2	Content complete
V 0.3	Version for internal WP2 review
V 0.4	Version for consortium review
V 1.0	Version reviewed and approved by PMT

Disclaimer

This work has been performed in the framework of the Horizon Europe project DETERMINISTIC6G co-funded by the EU. This information reflects the consortium's view, but the consortium is not liable for any use that may be made of any of the information contained therein. This deliverable has been submitted to the EU commission, but it has not been reviewed and it has not been accepted by the EU commission yet.

Executive summary

This deliverable continues the work performed in the project related to precise time synchronization within the context of sixth generation (6G) networks, which are expected to play a pivotal role in various industrial use cases (e.g., extended reality (XR), autonomous driving, exoskeletons, adaptive manufacturing, etc.).

This report focuses with regards to time synchronization specifically on validating hot-standby architectures using simulation in cooperation with WP4. The report explores packet delay minimization techniques that can be applied to PTP traffic and evaluates the security framework's efficiency in cooperation with WP3 by using a packet delaying attack scenario. This report also analyses some options for improving time synchronization information using separate slices or allocating a different data flow with higher Quality of Service (QoS) to deliver time synchronization.

The report highlights the importance of time synchronization as the physical and the digital worlds converge to a cyber-physical continuum. To this end, it analyses the hot-standby solution to improve the robustness of time synchronization. It also discusses the challenges posed to time synchronization from a security point of view, emphasizing the importance of resilience in time synchronization mechanisms.

The report includes an analysis of the usage of QoS flows for reducing delay to improve time synchronization. Moreover, the report includes measurements of delay variation when using QoS data sessions in a setup using off the shelf commercial components.

Contents

List of Figures

List of Tables

1 Introduction

As future networks evolve, they are expected to play a pivotal role in diverse application domains, including industrial automation, healthcare, and augmented reality [D6G-D1.1], among others requiring dependable communication.

One of the critical challenges that future communication networks must address is the precise time synchronization across devices and network elements within heterogeneous environments (mobile and stationary) and across heterogeneous communication technologies (wired and wireless). Time synchronization is fundamental to the proper functioning of various applications, particularly in industrial settings where coordination, automation, and data integrity are paramount. On one hand, precise time synchronization to achieve accurate temporal alignment of data and processes is essential for optimizing efficiency, minimizing errors, and ensuring seamless interoperability in these advanced industrial scenarios. On the other hand, it is equally imperative that time synchronization systems are resilient in the face of potential failures and fortified against security threats. Industrial environments are more sensitive than public networks to network disruptions, hardware malfunctions, and cyber-attacks, all of which can jeopardize the integrity of communication networks. Hence, a delicate balance must be struck between achieving time accuracy and implementing robust security measures to ensure reliable and secure operation of industrial processes.

1.1 Objective of the document

This report presents three major contributions. The first contribution consists of an analysis of hot-standby for establishing a resilient and continuous time synchronization framework for future 5G advanced (5G-Adv) and 6G networks. This mechanism is poised to fortify the temporal accuracy and reliability of critical industrial processes in the face of potential disruptions.

The second contribution in this report includes the description of the security framework's efficiency by using packet delaying attack scenarios.

The last part of the report includes an analysis of different options to improve the delivery of time synchronizations with QoS flows, which are presented together with measurement results.

1.2 Relation to other work packages

Within the technical work packages of the DETERMINISTIC6G project, D2.4 is part of WP2, which focuses on 6G-centric enablers for deterministic communication services. The relation of D2.4 to other work packages is shown in Figure 1-1 it takes input on the application requirements for time synchronization coming from WP1 [D6G-D1.1]. D2.2 takes input from the security mechanisms for deterministic communications presented in WP3 [D6G-D3.2]. The feasibility of the time synchronization enhancements proposed in this report are validated with simulation scenarios in WP4, which focuses on a 6G deterministic communication validation framework.

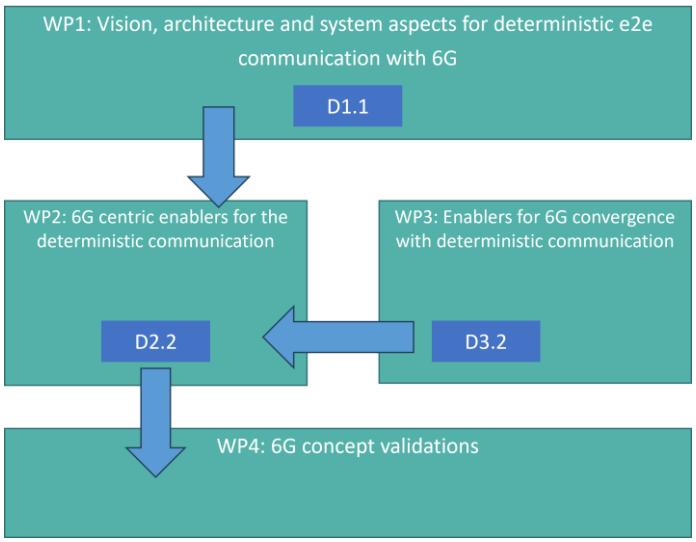


Figure 1-1: Relationship of D2.2 to other work packages and deliverables

1.3 Background concepts

For the self-contained purpose, we briefly present in this section some background concepts that are used in this deliverable.

1.3.1 Precision time protocol (PTP)

In a packet-switched network, PTP [IEEE 1588-2019] can be used for clock synchronization between the clock of one participant that serves as the main clock, known as the server clock, and the clocks of other participants, referred to as client clocks. The objective is to ensure that the client clocks maintain the same timestamp as the server clock. The synchronization process between the server and client is based on the exchange of timestamped messages between them to allow the client to calculate the clock offset, that is, the difference between its clock and the server clock.

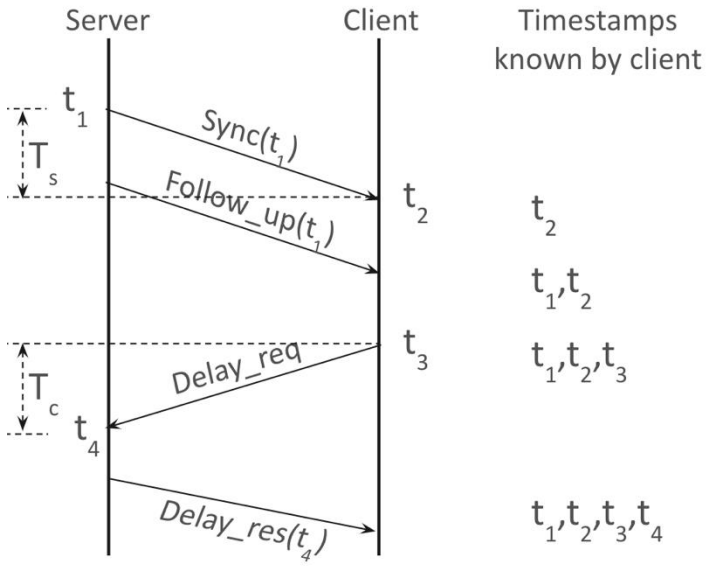


Figure 1-2. Time synchronization using 2-step PTP

Figure 1-2 presents the PTP messages to synchronize a client clock with a server one. The server sends a sync message to the client and timestamps the moment t1. Since t1 can be known exactly only once the message leaves the server, it sends a follow_up message to carry t1 to the client. The client receives the sync message at timestamp t2 according to its local clock. Since the transmission is not instantly, the difference between t2 and t1 represents the offset between client clock and the server clock in addition to the propagation delay Tdelay :

$$T_{offset} = (t_2 - t_1) - T_{delay} \quad (1)$$

To calculate the propagation delay, the client sends a `delay_req` message and timestamps the moment t3, at which the message leaves the client. The server receives the message at t4 according to its clock and send a delay_res message to carry t4 to the client. The client assumes that the communication links are symmetric and obtain the propagation delay:

$$T_{delay} = \frac{T_s + T_c}{2} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \quad (2)$$

Combining the equations above, the client calculates the offset of its clock and the server clock as given by the following equation:

$$T_{offset} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \quad (3)$$

The client then updates its clock accordingly as below:

$$Clock_{client} -= T_{offset} \quad (4)$$

1.3.2 Programable data planes & P4

Software Defined Networking (SDN) is driving a new approach to network management by decoupling the control and data planes. It reduces reliance on dedicated hardware by introducing programmable devices that handle network traffic processing and control. With SDN, network administrators can centrally manage and define network behavior using a software-based controller, enabling dynamic control over data traffic flow (data plane) and forwarding decisions based on network policies and configurations. However, traditional SDN mainly focused on programmability at the control plane, while the data plane remained limited to pre-configured policies and lacked flexible programmability.

Data plane programmability enables network owners to define data plane functionality through software running on programmable networking devices. P4 [BDG+14], a domain-specific language, is designed for programming these devices to handle packet processing. One of P4's key feature is its protocol independence, allowing networks to adapt to different protocols without hardware changes. Additionally, P4 facilitates seamless interaction between the programmable data plane and the

control plane, enabling efficient coordination between control logic and packet processing on network devices.

1.3.3 In-band network telemetry

Traditional network monitoring approaches deduce network status by either injecting and then observing the operational status of the probe packets (active monitoring) or by analyzing the real network packets via proxy reporting, from switches, or traffic mirroring, via packet capturing or sniffing (passive monitoring). In-band Network Telemetry (INT) has emerged as a new technical term [TSZ+21] for high-precision network monitoring in real-time. It refers to an automated process that enables the data plane to remotely collect, process, and report network state information without requiring intervention from the control plane. INT combines data packet forwarding with network measurement. By integrating data packet forwarding with network measurement, INT allows packets in the data plane to carry telemetry instructions and transport collected metric results, enabling real-time network visibility and analysis.

An INT system usually consists of INT-capable network devices, an INT collector, and eventually a network controller that configures the devices. An INT-capable device collects the network status by inserting metadata into packets. The device will not attach its information into a packet if the resulting packet size is greater than its Maximum Transmission Unit (MTU). An INT collector receives and extracts the INT data generated by the devices.

1.4 Structure and scope of the document

After the introduction in Section 1, Section 2 presents a solution to provide reliable time synchronization for end-to-end time awareness. This section focuses on the importance of time synchronization and how hot standby solution can deliver the required robustness. In particular redundancy provided with hot-standby enhancements to 3GPP network functions and reliable distribution of timing information are presented. Section 3 analyses the solutions proposed to secure the data transfer for time synchronization. Section 4 presents some alternatives to improve the delivery of time synchronization either to allocate a different slice or request a new flow with QoS for time synchronization messages. Finally, Section 5 concludes the deliverable with some discussions on the future work.

1.5 Terminology used in the deliverable

In this deliverable, we follow the inclusive terminology as decided by IEEE 1588 [IEEE 1588g-2022] standardization bodies to describe the time synchronization architectures. Whereby the following should be kept in mind.

- Master is replaced with timeTransmitter (tT)
- Slave is replaced with timeReceiver (tR)
- grandmaster (GM) remains grandmaster
- best master clock algorithm (BMCA) is replaced with best timeTransmitter clock algorithm (BTCA)

2 Resilient time synchronization for end-to-end time awareness

The development of 6G networks is bringing a new era of ultra-reliable, time-sensitive communication, where precise synchronization is essential for industries such as autonomous systems, industrial automation, and smart grids. At the heart of this evolution is Time-Sensitive Networking (TSN), a framework designed to ensure deterministic communication by minimizing latency and jitter. To achieve this, TSN relies on key mechanisms such as time synchronization, traffic management, and resource allocation, all of which contribute to the seamless and predictable flow of data. A critical component of TSN is the IEEE 802.1AS standard, which establishes the generic Precision Time Protocol (gPTP) to enable highly accurate time distribution across network devices. One of the most important functions within this system is the selection of a GM clock, which is responsible for providing the reference time to all other devices. Traditionally, this selection has been handled by BTCA, which determines the most suitable GM based on network conditions.

As TSN expands into wireless networks, 5G technology has already introduced a virtual TSN bridge, which includes the network functions to support synchronization and TSN communications between end stations. The transition to 6G networks builds upon this framework while maintaining backward compatibility with previous standards. However, one of the major challenges with existing time synchronization methods is the inefficiency of BTCA in responding to network disruptions. It struggles to handle transient clock faults and is slow to recover from link failures, leading to potential synchronization instability. To counter these limitations, the IEEE 802.1ASdm standard introduces a new approach that eliminates BTCA in favor of a redundant hot standby GM, ensuring a more resilient and fault-tolerant synchronization system for wired networks [IEEE 802.1ASdm].

Building on this concept, we propose an extension of the hot standby GM mechanism to 6G-TSN networks, aiming to create a more robust synchronization infrastructure. A key aspect of this approach involves the strategic placement of multiple GMs to enhance stability and minimize the risk of disruptions. We first provide an overview of important considerations for selecting the two GM in 6G-TSN networks, initially discussed in detail in D2.2 DETERMINISTIC6G use cases and architecture principles [D6G-D2.2]. We develop a time synchronization framework as presented in D4.4 DETERMINISTIC6G DetCom simulator framework 2 [D6G-D4.4] and analyze the performance for different options for the two GM locations in a 6G-TSN network.

2.1 Overview of time synchronization mechanisms in 5G-TSN networks

In 5G-TSN networks, the 5G system (5GS) functions as a time-aware virtual TSN bridge, ensuring synchronization between TSN end stations. The synchronization process relies on the IEEE 802.1AS standard, which defines the gPTP to distribute accurate time across the network [IEEE 802.1AS]. The time synchronization messages are time stamped at the ingress and egress of the 5GS, thus calculating the residence time. The time synchronization accuracy greatly depends on the correct estimation of this residence time. Three key synchronization mechanisms are standardized by 3GPP for 5G-TSN networks including downlink synchronization, uplink synchronization, and timing as a service (TaaS). The different time synchronization options are shown in Figure 2-1. The time error budget for 5G system is 900 ns [TS 22.261].

Downlink synchronization considers a TSN GM on the network side of 5GS [AMS+21]. Uplink synchronization considers a TSN GM on the device-side of the 5GS [3GPP-TS23501]. In this case, the time synchronization messages traverse the air interface between the UE and gNB twice for UE-to-UE synchronization.

Timing as a Service (TaaS) enables 5G networks to provide highly accurate time synchronization to external systems, including TSN-enabled industrial networks [3GPP-TS23501]. By leveraging global navigation satellite system (GNSS) time or another high-precision source, the 5G network distributes timing information to various end stations. This approach allows industries to integrate 5G networks into their existing time-sensitive infrastructures while ensuring consistency in synchronization across different systems.

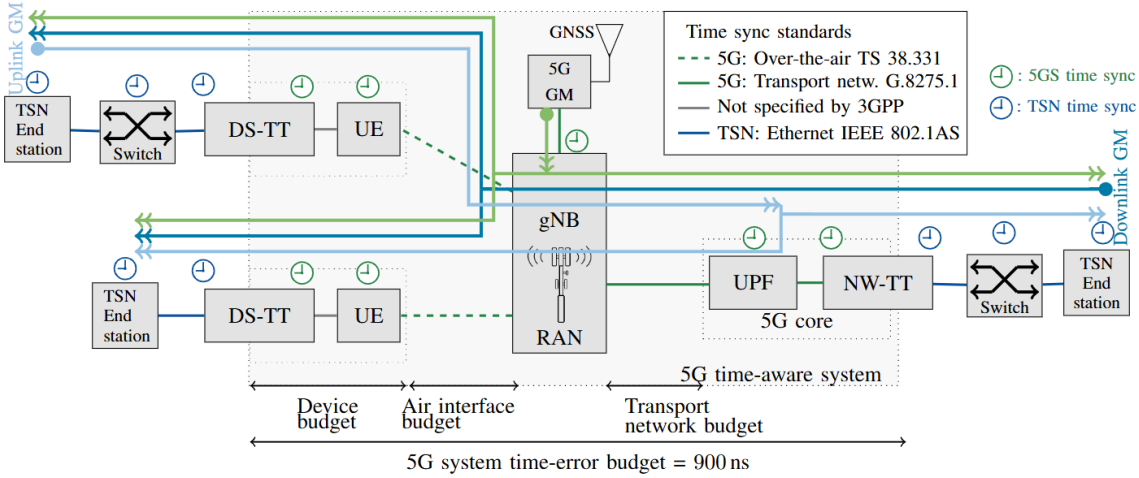


Figure 2-1: Time synchronization in 5G-TSN network where 5GS operates as a time-aware system. Time synchronization for downlink GM (dark blue), uplink GM (light blue) and 5G GM (green) are provided. The GM are represented with small colored dots.

For all the above-mentioned time synchronization schemes in a 5G-TSN network, BTCA is used to automatically select the GM clock within a network. The BTCA evaluates clocks based on criteria like the clock priorities and its quality, thus ensuring the best clock serves as a GM in the network. However, as mentioned in D2.2 [D6G-D2.2], BTCA has limitations, including susceptibility to delays in case of failures and its inability to account for transient faults in the clock quality [BBF+13]. Hence leading to an unstable time synchronization and longer delays when waiting for the selection of a new GM.

2.2 Hot standby operation

Given the limitations of the BTCA, IEEE 802.1ASdm [IEEE 802.1ASdm], the hot standby amendment to the IEEE 802.1AS standard, enhances time synchronization mechanisms by introducing redundancy and improving fault tolerance. In this case, the BTCA is no longer used, rather a static configuration of GM is done by a management entity. One of its key features is the support for a redundant hot standby grandmaster (GM), which ensures continuous synchronization even in the event of a failure in the primary GM. It requires the hot standby GM to be synchronized to the primary GM before transmitting its synchronization messages. This amendment aims to reduce downtime and improve synchronization reliability in wired TSN networks, and similar concepts can be extended to 6G-TSN networks to further enhance resilience in time-sensitive applications.

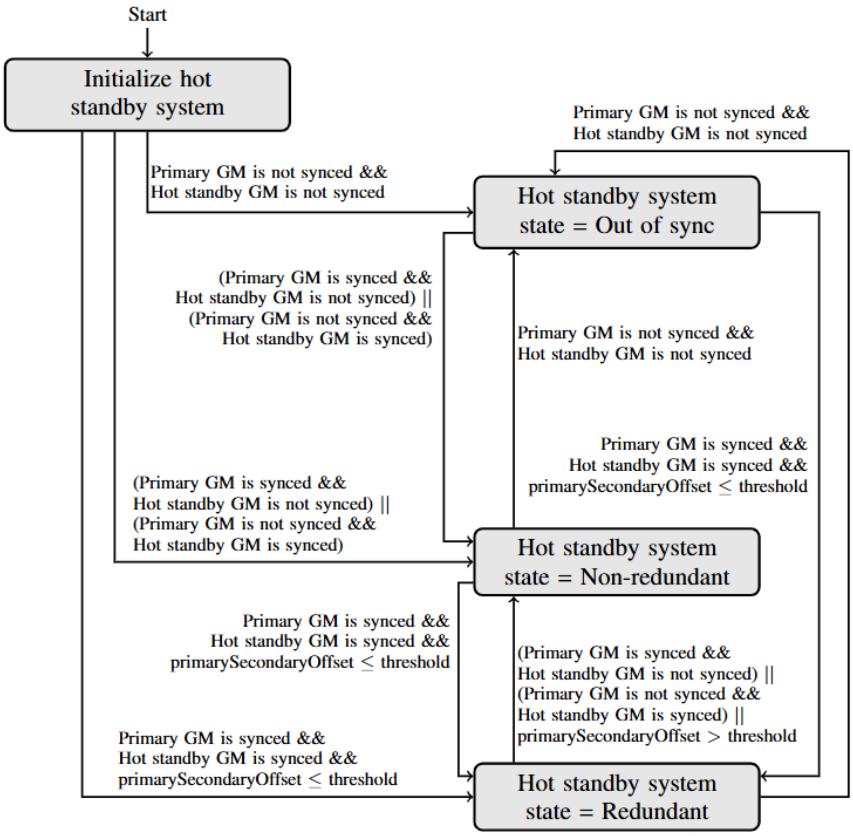


Figure 2-2: Hot standby system state machine when hot standby is enabled, abstracted from [IEEE 802.1ASdm]

In a hot standby system, each time-aware device runs two PTP instances in separate domains: a primary and a hot standby. The primary domain is used as long as both are available, but if it fails, the system switches to the hot standby. The hot standby GM stays synchronized with the primary GM to ensure consistent timing. A variable, *primarySecondaryOffset*, tracks the time difference between

them, and if it exceeds a threshold, the system shifts to a non-redundant state. The state machine for a hot standby system is depicted in Figure 2-2. Here the state “is synced“ means that the synchronization requirements of the PTP instance are fulfilled. An optional split functionality helps recover synchronization by transferring time from a synced to an out-of-sync instance, restoring redundancy (i.e., two domains are available).

2.3 Overview of considerations for the GM locations with hot standby operation in 6G-TSN networks

Both the 3GPP and IEEE 802.1 standardization bodies recognize the importance of continuous time synchronization and have incorporated support for multiple GMs and synchronization paths to enhance network resilience. However, certain limitations persist, particularly regarding fault tolerance and synchronization stability in the event of failures. Introducing redundancy in GM selection can significantly improve fault tolerance and ensure uninterrupted synchronization for critical applications. In 6G networks, adopting a solution similar to IEEE 802.1ASdm, which employs a hot standby GM, could be an effective strategy for minimizing downtime and maintaining synchronization integrity.

When considering possible options for the placement of the two GMs in the light of the standardized time synchronization mechanisms, the following should be considered:

- Using the 6G GM as one of the GMs could benefit from the higher quality (access to GNSS time) 6G clock. In the case of failure of the primary (6G) GM (e.g., loses access to GNSS time), the gNB can still support time synchronization for a certain time while being in a holdover state.
- The 6G clock is used to synchronize the other 6G elements like the gNB, UPF, and other UEs for their proper functioning. Hence, it is not desirable that the 6G GM is synchronized to an external network (non 6G clock), making it not a suitable choice for a hot standby GM.
- Similarly, for use cases involving scenarios where one of the GMs is the 6G GM, the optional split functionality for hot standby synchronizing 6G clock should be set to FALSE. This will prevent the hot standby GM from attempting to synchronize the 6G GM’s clock once the 6G GM or 6G connection is restored after a failure.
- Having the GM on the network side helps keeping the time-error low for device side TSN end-stations given the timing messages only travel once over the air interface.

2.4 Analysis of time synchronization architectures with hot standby

Using the simulator as presented in D4.4 [D6G-D4.4], we analyze the performance of time synchronization with hot standby and BTCA for two different time synchronization architectures.

2.4.1 Simulation setup

Our simulation network architecture is shown in Figure 2-3 and Figure 2-4. It consists of a converged 6G-TSN network with multiple TSN bridges and end stations. The 6G system (6GS) integrates with the

TSN network as a time-aware system. The TSN End station 1 is connected to switch A, which connects to the network side of 6GS via the NW-TT. The TSN End station 2 is connected to switch B, which connects to the device side of the 6GS via DS-TT 1. The TSN End station 3 connects to the device side of the 6GS via DS-TT 2. These DS-TTs are configured with the uplink and downlink delays using the PD-Wireless-5G-1 dataset [HDM+23]. We later refer to TSN end stations as stations. The 6G clocks of the TSN translators are configured to drift apart a maximum of ±450 ns, ensuring the total time error budget of 900 ns between two TSN translators, as standardized by 3GPP [3GPP-TS23501], is met. For TSN clocks we configure a random drift rate. This random drift rate is based on the time deviation (TDEV) requirements for local clocks from the IEEE 802.1AS standard [IEEE 802.1AS] but simplified to a maximum TDEV of 5 ns in an observation interval of 1 s and a maximum frequency offset of 100 ppm. The gPTP components in all network devices are configured using the recommended values in the IEEE 802.1AS standard. This means, the sync interval is configured to be 125 ms, the interval between announce messages is configured to be 1 s. The sync and announce timeouts are configured to be three times the respective interval value. We simulate two different BTCA implementations. One, BTCA where the Announce messages are sent immediately to neighboring nodes if BTCA execution results in the selection of a new GM. And an alternative implementation, BTCA2 where the PTP instance waits until the end of the current announce interval to send out the Announce messages.

Simulation settings are presented in Table 1 below.

Simulation parameter	**Value**
Time deviation for TSN clocks	5 ns
Sync interval	125 ms
Announce interval	1 s
Maximum Freq. offset of TSN clocks	100 ppm
Time error budget TSN translators	± 400 ns

Table 1: Simulation Parameters

2.4.2 Simulation scenarios

In the following, we describe our three simulation scenarios, which are all based on the network described above. The intended primary GM is the NW-TT in all our simulations. In case the NW-TT is selected as the GM, it synchronizes its local TSN clock to its local 6G clock as described in our simulation setup.

In the first scenario we consider the 6G GM as the primary GM and a network side TSN end-station as a hot standby GM. The two-time domains are depicted as blue and green in Figure 2-3. We configure the port states of gPTP manually in the hot standby scenario, but the same outcome can be achieved by setting the priority1 values per domain and letting BTCA select the GM of every domain. The primary domain (domain 0) spreads across the whole network with the NW-TT (using the 6G clock) configured as the GM. The hot standby domain (domain 1) has the TSN End station 1 set as the GM and also synchronizes all TSN devices in the network.

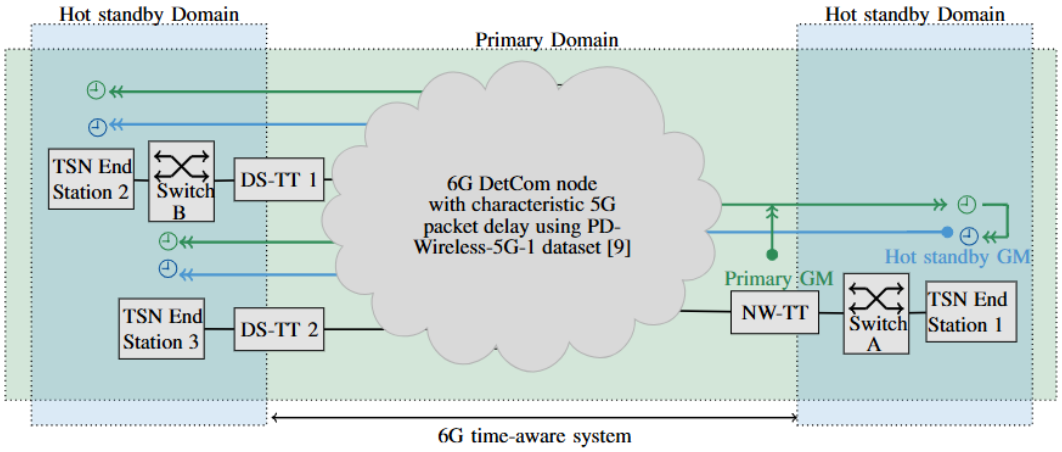


Figure 2-3: 6G GM as primary GM and network-side TSN end station as hot standby GM.

The second scenario (as shown in Figure 2-4) considers two different hot standby GMs for different sections of the network. We consider the 6G GM as the primary GM (green time domain). One hot standby GM at a TSN end station on the network side of the 6GS (purple time domain) and a second hot standby GM on a TSN end station connected to the device side of the 6GS (blue time domain).

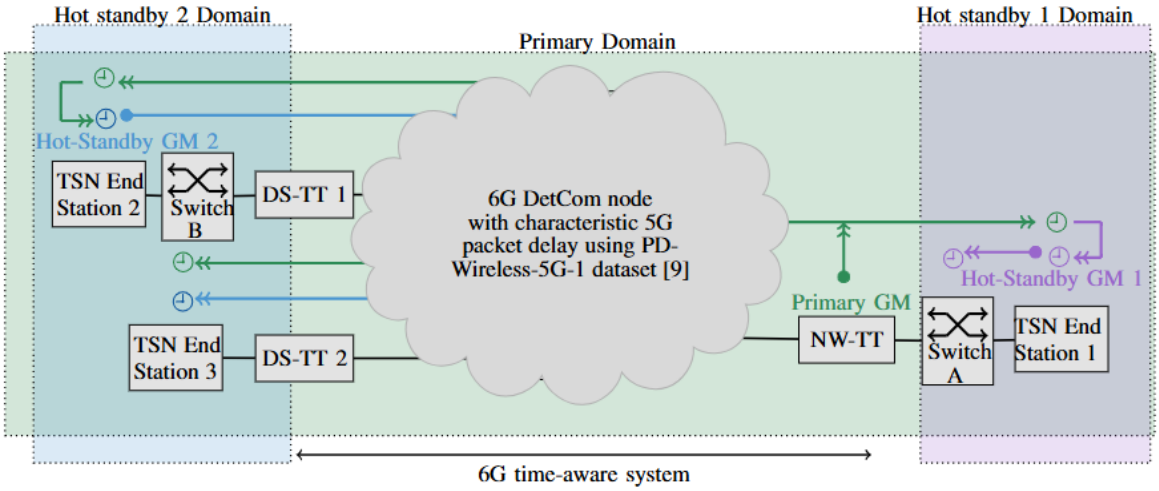


Figure 2-4: 6G GM as primary GM and one hot standby GM on device-side TSN end station and another hot standby GM on network-side TSN end station.

The third scenario uses BTCA to determine the best GM. In our setup we assume that TSN End Stations 1 and 2 are equipped with the most accurate clocks, which is indicated by a clockAccuracy value that is lower than the value of the remaining devices in the network. Therefore, the BTCA will select these devices as the GMs of the corresponding network domains. In our setup we assume that TSN End stations 1 and 2 are equipped with the most accurate clocks, which is indicated by a clockAccuracy value that is lower than the value of the remaining devices in the network. Therefore, the BTCA will select these devices as the GMs of the corresponding network domains.

Failure cases

Within the scenarios specified above, we analyze three failure cases:

1. The NW-TT loses access to 6G timing. Hence, the NW-TT becomes GM-incapable and stops transmitting sync messages as the GM. In the simulation, this failure occurs from 10 s to 50 s.
2. The 6G network availability is compromised. In other words, the network connection between the NW-TT and DS-TTs is lost. In the simulation this failure occurs between 70 s and 110 s.
3. The connection between NW-TT and switch A is lost. In the simulation this failure occurs between 130 s and 170 s.

2.4.3 Simulation results

In this section, we analyze the scenarios and failure cases introduced above based on different metrics, namely the influence of the 6G time-aware system, the adaptability to failures and the out-of-sync time.

Influence of the 6G time – aware system:

First, we evaluate the influence of the 6G system on the synchronization accuracy of the TSN system in 6G-TSN networks for hot standby Scenario 1 in all three failure cases. Figure 2-5 shows the clock drift of the 6G clocks located in the TSN translators and the drift of the TSN clocks located in stations 1 and 2. A clock offset of 0 is the ideal time. As can be seen, the total drift error for the 6GS remains within the 900 ns budget. During intervals without failure, e.g., between 50 s and 70 s, the clock drift of the TSN clocks tightly follows the clock drift of the 6G clocks. This indicates that a main source for the TSN clocks drifting apart is the inaccurate synchronization of the 6G system. Since the net effect of the failure cases is partly obfuscated by this drift of the 6G clocks, all subsequent simulations are based on an ideal 6G clock, i.e., free from any drift or offset.

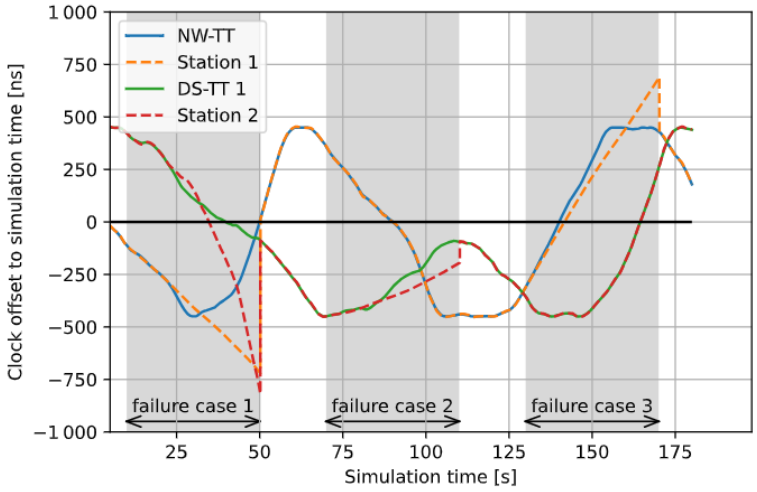


Figure 2-5: Clock drift in scenario 1 (BTCA) with drifting 6G clocks.

Adaptability to failures:

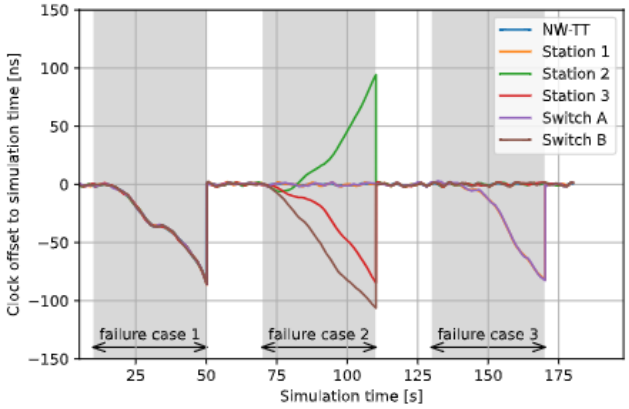
Next, we compare the adaptability in the different time synchronization scenarios across all three failure cases. Our analysis focuses on the clock offset of TSN nodes within the 6G-TSN network relative to the ideal clock under these scenarios, with results presented in Figure 2-6. Since we consider idealized 6G clocks for this analysis and the NW-TT is the primary GM, the NW-TT always has a zero clock drift.

In the first failure case, where the NW-TT becomes GM-incapable, a key distinction between the two hot standby scenarios becomes evident. In hot standby scenario 1, all devices seamlessly switch to the hot standby domain, with station 1 assuming the role of hot standby GM. Here, all TSN nodes are perfectly synchronized with each other. In contrast, in hot standby scenario 2, the network splits into two separately synchronized parts. This occurs because the device side synchronizes to hot standby GM 2 (station 2), while the network side synchronizes to hot standby GM 1 (station 1). As a result, the two sections of the network drift apart from each other, despite all network links remaining intact and theoretically capable of maintaining synchronization. In the BTCA scenario, station 1 is selected as the GM, thus resulting in a similar behavior of the TSN nodes as in hot standby scenario 1.

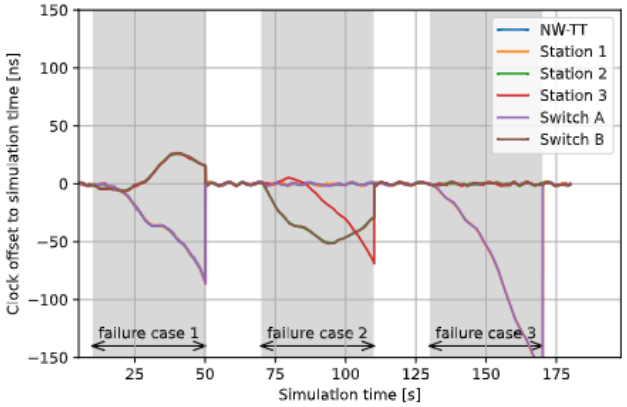
In the second failure case, where the connection between the NW-TT and DS-TTs is lost, a network split into multiple segments is unavoidable due to the lack of link redundancy. In both hot standby scenarios, the network side TSN nodes remain synchronized to the NW-TT as the primary GM. However, the device side TSN nodes respond differently depending on the hot standby scenario. In hot standby scenario 1, the device side TSN nodes lack an alternative synchronization source, causing their clocks to drift randomly apart. In contrast, in hot standby scenario 2, station 2 and switch B remain synchronized by the hot standby Domain 2 and only station 3 drifts apart, as the link between the DS-TT 1 and DSTT 2 is broken and there is no available hot standby GM to synchronize to. The BTCA scenario behaves similar to the hot standby scenario 2, eventually selecting station 2 as the new GM to keep switch B synchronized.

In the third failure case, where the link between the NW-TT and switch A is broken, we observe a similar behavior for all three scenarios. The device side TSN nodes remain synchronized to the NW-TT. In both hot standby scenarios, station 1 is the hot standby GM for the network side TSN nodes. Similarly, in the BTCA scenario, station 1 is eventually selected as the GM by the BTCA algorithm.

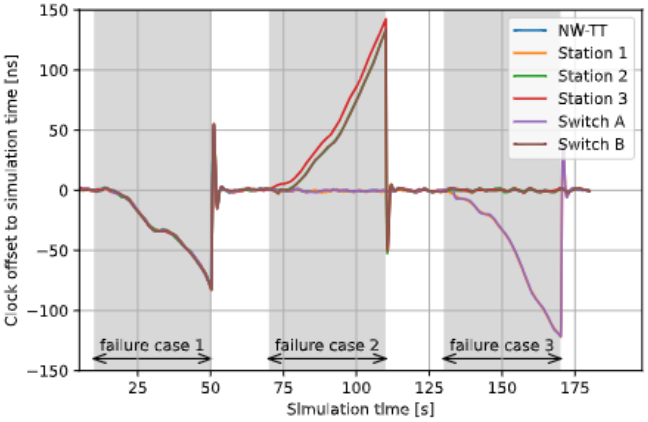
In summary, when comparing BTCA and hot standby, we observe that in the BTCA scenario, in any failure case, there is only one group of clocks drifting away in the same direction from the ideal GM at the NW-TT per disconnected network part. This shows that BTCA is able to find exactly one new GM for all interconnected parts of the network as long as there is at least one GM-capable device in each separate part of the network. Hot standby with a static configuration, on the other hand, leads to parts of the network being without a GM if the hot standby domain is separated as well. Having different hot standby domains can additionally lead to devices synchronizing to two different GMs even though the network topology would still allow a single GM to be present.



(a) Hot standby scenario 1



(b) Hot standby scenario 2



(c) BTCA scenario

Figure 2-6: Drift of TSN clocks in a setup with ideal 6G clocks

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 and the resulting clock offset for switch A in hot standby scenario 1 against the two BTCA implementations in failure case 3. In this failure case, in hot standby scenario 1, the primary domain is no longer available, and the switch A switches to the hot standby domain. In contrast, in the BTCA implementations, BTCA needs to select new GM. The out-of-sync duration is defined as the time between a syncReceiptTimeout and a new Sync message.

In order to also analyze the impact of multiple hops when BTCA searches for a new GM, we generate results for 10 and 50 intermediate hops between switch A and station 1. The following results are based on 100 simulations for each simulation scenario. In Figure 2-7, we present the out-of-sync time between switch A and the newly selected GM station 1. Additionally, Figure 2-8 shows the cumulative distribution of the maximum offset between switch A and station 1.

As expected, in the hot standby scenario, the out-of-sync time is minimal, as devices are able to switch to the hot standby domain almost immediately after the sync timeout. The short out-of-sync time also leads to a negligible maximum offset between switch A and station 1. Similarly, the effect of the number of hops is negligible, as the sync timeout occurs independently on every device and no new GM needs to be selected.

The BTCA implementations, on the other hand, take longer to adapt to network failures compared to hot standby scenario 1. This longer out-of-sync time also leads to a greater clock offset. There are two main reasons why this happens. First, the Announce timeout is greater than the Sync timeout. Thus, the time until BTCA starts to react to the failure in the network is greater than in the hot standby scenario. The second reason is that BTCA needs time to exchange Announce messages in order to select a new GM, while the hot standby GM is already preconfigured in the hot standby scenario. This effect especially becomes clear in BTCA2, which is highly affected by the number of intermediate hops. If Announce messages are not sent immediately, every additional hop introduces a longer out-of-sync time and thus, a greater maximum offset. In summary, our results confirm the assumption that BTCA reacts slower to network failures, and thus hot standby outperforms BTCA with regards to out-of-sync time and, as a consequence, also in the clock offset.

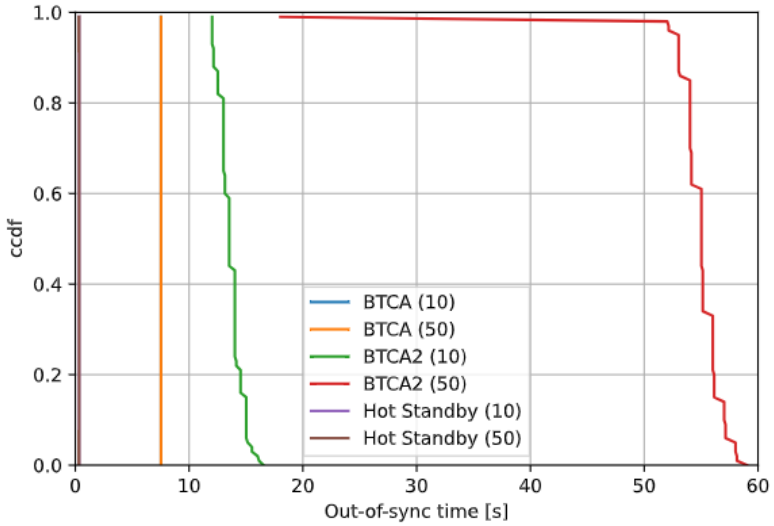


Figure 2-7: Complementary cumulative distribution of the out-of-sync time of switch A

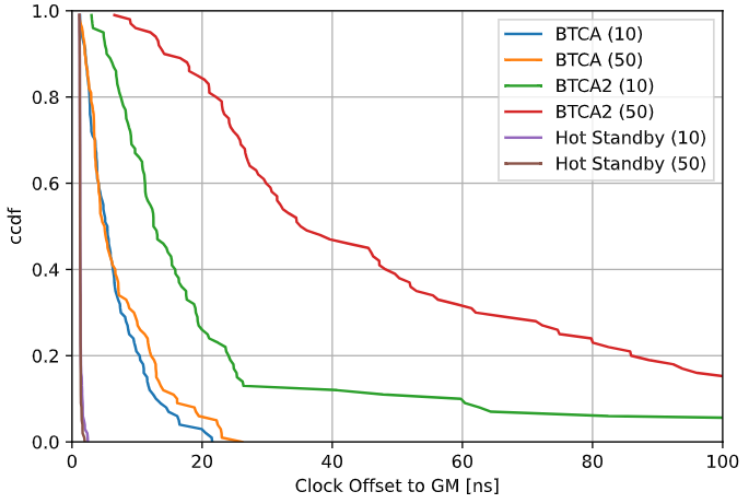


Figure 2-8: Cumulative distribution of the maximum offset between switch A and new GM.

2.5 Key takeaways

The scenarios with a hot standby GM outperform BTCA in terms of clock drift and out-of-sync time in case of failures. However, our analysis indicates that in certain failure cases, parts of the network may be left without a GM when using static GM configuration (hot standby). In contrast, BTCA is able to dynamically select a new GM in the event of failure, albeit with some delay. This leads us to conclude that the decision to use a static GM configuration with hot standby or a dynamic GM configuration with BTCA should be based on the network topology and the presence of other redundancy measures, such as link redundancy. For instance, in a network with multiple redundant links, a static GM configuration with hot standby might be more effective. Conversely, in a network with fewer redundancy measures, a dynamic GM configuration could provide better resilience by dynamically selecting new GM in case of failure.

3 Security and time synchronization reliability

3.1 Introduction

5G and the upcoming 6G networks are proposed to support various applications requiring high reliability and low latency, including industrial automation, smart cities, etc. In response, their integration with emerging deterministic networking technologies such as TSN and Deterministic Networking (DetNet) has been recently introduced to ensure deterministic communication with low latency and minimal jitter. These technologies rely on time synchronization between the nodes in the network to coordinate packet scheduling, traffic shaping, and time-aware resource allocation, which are essential for guaranteeing reliable and predictable network performance.

PTP is a widely adopted standard for time synchronization in packet-switched networking systems. It can provide sub-nanosecond accuracy. Despite its importance, PTP was initially designed without built-in security mechanisms. TSN and DetNet are usually deployed in a closed and controlled network that protects the network from any external security threat. However, their integration in 6G deterministic communication which consists of different heterogeneous devices and technologies, the exposure of this attack vector to various security threats is increasing. In the worst case, due to its heterogeneous nature, an infrastructure component can even host an attacker, or be compromised by an attacker, making the comprehensive secure distribution of timing information in the network against a malware infiltration a very tedious task [AS21].

Although Annex P of IEEE 1588-2019, also known as Profile D, has introduced several security features to protect clock synchronization to deal with attacks such as spoofing and message tampering, interruption of message exchange, and replay of messages, these mechanisms remain ineffective against Time-Delay Attacks (TDA). Unlike traditional cyberattacks that alter packet content, a TDA does not manipulate PTP messages but only delays the packet, for example, an attacker intercepts a PTP message and holds it for a given interval before relaying it to its destination. The attacker can use various delay attack strategies such as maliciously added delay can be constant, jittered, or slowly wandering.

As TDA can be detected only via monitoring, we propose to use In-band Network Telemetry (INT) to monitor PTP frames in real time. The INT metadata are embedded inside Type-Length-Value (TLV) extensions of PTP frames. This integration does not cause any incompatibility in the network devices, such as PTP server or client. Using PTPv2 extensions for INT has significant advantages, as it allows for high-precision latency measurements, real-time network optimization, and proactive security monitoring, such as detecting time-delay attacks on PTP synchronization.

To evaluate the proposed approach, we implemented an open-source emulation to emulate precision time synchronization and to perform TDA attack & detection. As far as we know, this is the first programmable Transparent Clock implemented using P4 and it can be enabled or disabled its INT capability at runtime to encapsulate monitoring data into PTP frames. This emulation can provide a building block for the integration of time synchronization process into Software Defined Networking and programmability networking.

3.2 Threat Model

This section describes the attack and the possible vulnerabilities are analyzed.

3.2.1 Time-delay attack vulnerability

We can see in Equation (1) that the computation of the offset between server clock and client clock is based on the propagation delay Tdelay. In turn, Tdelay is calculated by assuming that the communication link is symmetric [IEEE 1588-2019]. However, this assumption introduces a vulnerability that attackers can exploit.

By analyzing network traffic, intercepting PTP messages and artificially delaying either sync or delay_req messages, an attacker can manipulate Tdelay , causing the client to compute an incorrect clock offset and update its clock inaccurately, as shown in Figure 3-1. It is worth noting that an identical delay introduced into both sync and delay_req messages has no impact on the behavior of the protocol. Therefore, a TDA can be performed either on sync or delay_req messages, or on both of them with different delays.

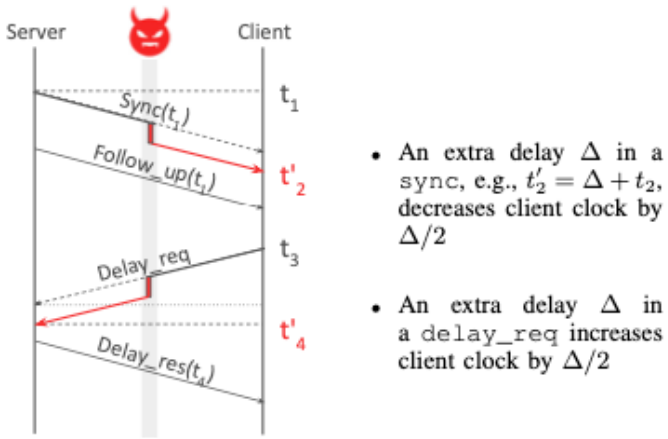


Figure 3-1: Time-Delay Attack in PTP messages

If an attacker delays a PTP sync message by Δ unit of time, then the message will arrive at the client at t'2 = t2 + Δ. In such a case, the client calculates the new clock offset as follows:

$$T'_{offset} = \frac{(t_2 + \Delta - t_1) - (t_4 - t_3)}{2} = \frac{\Delta}{2} + \frac{(t_2 - t_1) - (t_4 - t_3)}{2} = \frac{\Delta}{2} + T_{offset} \qquad (5)$$

From the equation above, we can conclude that if an attacker delays a sync message by Δ then the attacker can decrease the client clock by Δ/2. Similarly, if an attacker delays a delay_req message by Δ then the client clock will be accelerated by Δ/2.

A PTP message passed through a network device may experience different delays for different directions, such as uplink and downlink. To compensate for such an asymmetric delay variation of PTP messages, PTP version 2 introduces a correctionField that is a 64-bit field in PTP messages that accumulates the residence time of a message when it passes through multiple PTP-aware devices,

called Transparent Clocks. Precisely, when a sync message passes through a Transparent Clock (TC), the TC calculates the residence time of the message, such as, $t_{egress} - t_{ingress}$, and adds this residence time to correctionField of its corresponding follow_up message.

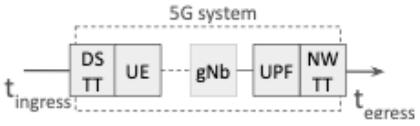


Figure 3-2: Time synchronization over a 5G system acting as a PTP-aware node

The client will then exclude the residence time from its Tdelay. Thus, the client can achieve more precise and scalable synchronization, especially in networks with TCs. This is the case when PTP messages pass through a 5G system which acts as a virtual TC, see Figure 3-2.

TDA can unintentionally occur in a TC when the correctionField does not reflect correctly the residence time of its message. Indeed, a vital prerequisite for correctly calculating the residence time in the virtual TC is that the $t_{ingress}$ and t_{egress} are measured by a single reference clock. In other words, it requires that the clocks of the device side TSN translator (DS-TT) and the network side TSN translator (NW-TT) are perfectly synchronized. Otherwise, a TDA can occur in such a TC.

3.2.2 Attacker model

In this work, we adopt a Man-In-The-Middle (MITM) adversary model [IW20] in which the attacker is in a position between the targeted PTP client and its associated server. The attacker can be either an external entity or an insider with malicious intents [Miz14]. An external attacker can gain such a strong position, for instance, by deploying low-level network attacks, such as ARP poisoning in order to redirect the traffic toward itself. An internal attacker can exploit, for example, a permission escalation vulnerability of the device firmware to gain control over a network device.

A MITM attacker can gain full control over the communication channel, such as dropping messages, inserting fake messages and changing fields of passing messages. However, since cryptography-based mechanisms, such as, MACsec or IPsec protocols, can be used for securing the communication between the server and its clients, inserting fake messages and changing fields of passing messages is detectable and therefore not an option. Instead, the attacker is supposed to aim at compromising the time synchronization process of the system using TDAs.

Nevertheless, the attacker is expected to have the expertise and the tools to be able to intercept and hold selected PTP messages to introduce asymmetric communication delays, leading to an incorrect calculation of Tdelay as the malicious delay introduced by the attacker is not taken into account by correctionField, thus it is not compensated by the client.

We focus on detection TDAs in End-to-End (E2E) PTP time synchronization process.

3.2.3 Time-delay Attack Detection

Our proposed detection approach relies on three signatures of TDAs.

- Attack must be continuous to effectively manipulate the client clock. PTP operates by regularly exchanging sync and delay_req messages, to adjust client clock and maintain accurate timing. If an attacker delays these messages only once or sporadically, subsequent synchronization cycles will correct the manipulated offset.
- Attack does not influence the server clock that serves as the reference clock. The server clock does not rely on incoming synchronization messages to adjust its time, but it relies on a trusted and highly accurate time source such as GPS signals. Consequently, the timestamps generated by the server clock, for example, preciseOriginTimestamp in follow_up messages, or receiveTimestamp in delay_resp messages, remain intact.
- In an E2E time synchronization scenario, an attack outside of a TC does not influence this transparent clock as it does not adjust its own clock based on PTP messages. Although, it is worth noting that a TC can be compromised to perform a TDA, this TDA does not affect other TCs in the network.

The first signature requires our detection framework to continue monitoring and detect TDA that usually occurs multiple times. The second and third signatures give us reliable timestamps generated by the server and TCs to be used to detect TDA as the timestamps are immutable from the attack.

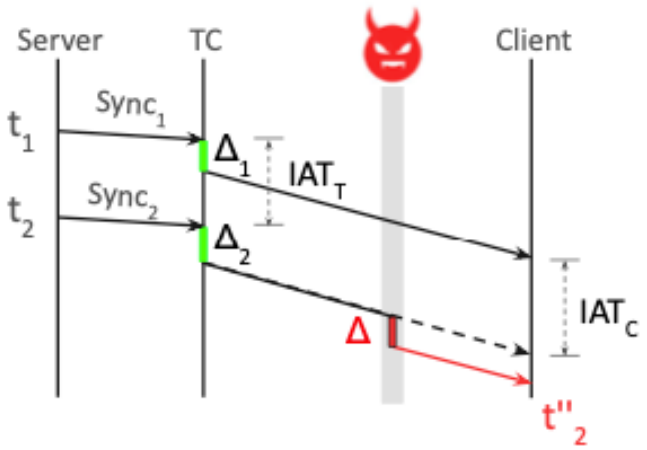


Figure 3-3: Detection of TDA via IAT Variation in E2E Time synchronization

To represent our detection, let us examine a simple E2E time synchronization scenario including a server clock, a transparent clock, and a client clock as shown in Figure 3-3. In a normal condition, the two sync messages should experience the same transmission delay when being sent from server to TC and then to client, thus we have the following equations in which Δi represents the residence time of the message i at TC:

$$IAT_t = t_2 - t_1 \qquad (6)$$

$$IAT_c = (t_2 - \Delta_2) - (t_1 - \Delta_1) \qquad (7)$$

When a TDA is executed to introduce an extra delay Δ in the second sync message, we get the new inter-arrival time IAT_c of these messages at the client:

$$IAT_c' = IAT_c + \Delta \quad (8)$$

We can deduce the value of Δ:

$$\Delta = IAT_c' - (t_2 - \Delta_2) + (t_1 - \Delta_1) \quad (9)$$

As ti and Δi are carried to the client in the preciseOriginTimestamp and correctionField of follow_up message respectively, we can obtain Δ at the client.

Given the accuracy requirement of the time synchronization process being T that represents the maximum deviation of client clock and server clock, then we see that the time synchronization process can tolerate a TDA within the delay of $2 * T$. Therefore, we obtain the trigger conditions for TDA detection as below in which we use a small parameter σ to set the sensitivity of the detection:

$$\Delta \geq (2 * T + \sigma) \quad (10)$$

3.3 Emulation framework implementation

3.3.1 Programmable transparent clocks

A TC enhances clock synchronization by measuring and compensating for the delay that packets experience inside network devices. Unlike ordinary clocks, TCs do not synchronize themselves but instead update the correction field in PTP packets to account for their internal processing time, known as residence time. Although P4-enabled hardware switches, such as Tofino and NetFPGA, are PTP-aware switches [KJC19], [MBB+23], [FOZZ+24], this is not the case for the virtual P4 switch, BMv2. The hardware switches allow P4 programs to request for PTP delay updates at egress MAC for TC when enabling update_delay_on_tx but this is not fully programmable by the P4 programs. We present in this section our BMv2 extension to support PTP and our TC implementation using P4. Implementing a TC using P4 allows for flexible, software-defined processing of PTP packets directly in the programmable data plane without requiring changes at the control plane level.

The key function of a P4-based TC is to accurately measure the residence time and adjust PTP messages accordingly to ensure precise time synchronization across the network. To achieve this in P4, we extend BMv2 to timestamp packets upon ingress and egress ports. Indeed, the current version of BMv2 provides ingress_global_timestamp and egress_global_timestamp meta variables to timestamp the moment when the packet started processing the P4 ingress and egress pipelines, respectively. However, it is important to note that the P4 ingress pipelines starts only after the packet arrives at its ingress port, and the egress pipelines completes before the packet is sent out to its egress port. As a result, ingress_global_timestamp does not accurately reflect the exact time the packet arrives at the input port, and egress_global_timestamp does not reflect the precise moment it leaves

the output port. This means that these values cannot be used to calculate the residence time of the packet when it passes through the P4 switch. We extended BMv2 to provide 2 additional timestamp values which represent the moment when a packet arrives or leaves the input or output port. The difference of these 2 timestamp values represents the residence time of the packet.

Once obtaining the residence time as the difference between these two timestamps above of a sync or delay_req packet, we need to store this value, which is then later added to the correctionField of its corresponding follow_up or delay_res packet, respectively. These pairs of packets, e.g., sync and follow_up, are correlated using the 3-tuple (clockId, portId, sequenceId).

3.3.2 Realtime monitoring via INT

PTP is primarily designed for time synchronization, but its extension fields can be leveraged for In-band Network Telemetry. To implement INT within PTPv2, our P4-based TCs encapsulate custom telemetry data in PTPv2 packet extension TLV fields. This approach ensures minimal overhead without introducing additional probe packets and leverages existing network infrastructure for efficient telemetry.

The structure of an extension TLV field to carry INT data is shown in Listing 1. We detail below each element:

- tlvType is configurable and must be a unique value which is different for existing TLV types as in Table 2.
- fieldLength is 26. Each TC inserts a PTP extension whose size is 26 bytes in total.
- switchID is a unique number to identify the TC.
- ingressTstamp and egressTstamp are the ingress and egress timestamps of the packet.
- correctionField is the value of the correction field before being adjusted by the TC.

```
header ptp_tlv_int_t {
    bit<16> tlvType;
    bit<16> fieldLength;
    bit<16> switchId;
    bit<64> ingressTstamp;
    bit<64> egressTstamp;
    bit<48> correctionNs;
}
```

Listing 1. Structure of TLV in P4

It is worth noting that the INT feature of our P4-based TC can be enabled or disabled at runtime.

Table 2: Existing TLV types in PTPv2

Value	Description
0x0001	Management TLV
0x0002	Organization Extension TLV
0x0003	Request Unicast Transmission TLV
0x0004	Grant Unicast Transmission TLV
0x0005	Cancel Unicast Transmission TLV
0x0006	Acknowledge Cancel Unicast TLV

3.3.3 Time-delay attack localization

By verifying the condition in Equation (10) at the client, a TDA can be detected in the network, but it is impossible to locate it. This is caused by the fact that the correctionField accumulates the residence time of all TCs on its path. The granularity of this correctionField is carried in INT data of the same packet. As each TLV extension provides correctionField and timestamps of the packet before arriving at a TC, we can easily determine if there exists a TDA between the TC and the server by applying Equation (10).

3.4 Experimental evaluation

3.4.1 Testbed setup

Figure 3-4 represents our testbed for evaluating the proposed monitoring framework, as well as its detection and localization of TDAs. The testbed consists of a Dell Laptop and a Raspberry Pi 3. In the laptop, which has 14 CPU cores and 32 GB of RAM, we use Mininet to emulate a virtual network having a PTP server, which is implemented by LinuxPTP [CM10] and a set of our P4-based transparent clocks. The PTP client, implemented also by LinuxPTP, is hosted in the Raspberry Pi, which is connected to the Laptop via an Ethernet cable.

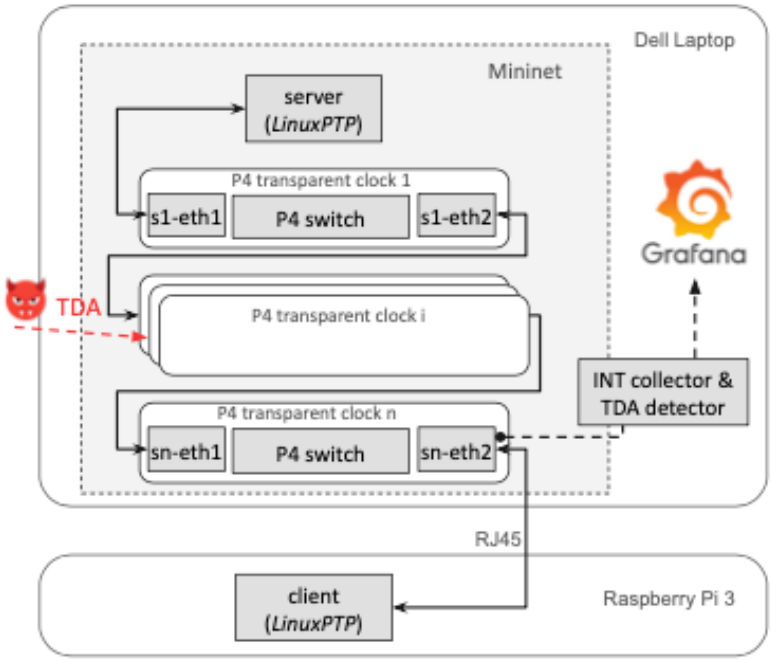


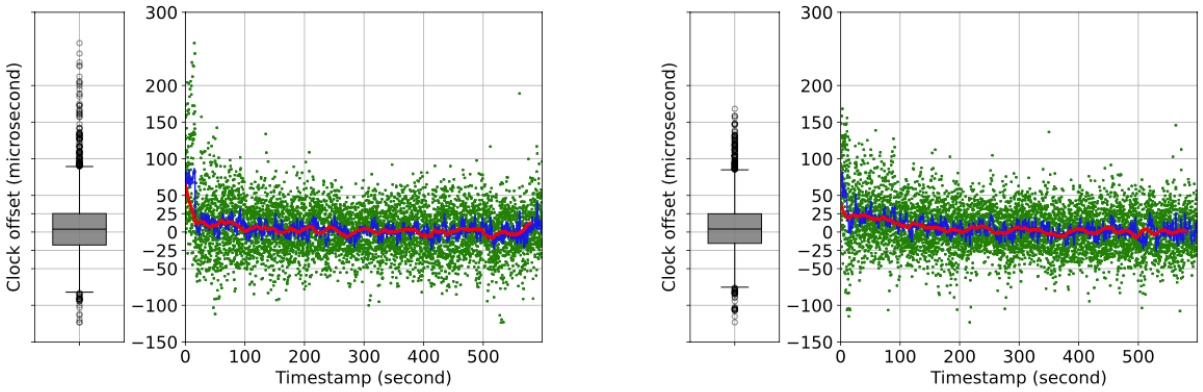
Figure 3-4: Testbed setup

It is worth noting that, in our testbed, all PTP clocks (server, client, and transparent) use Linux software timestamping although LinuxPTP supports PTP Hardware Clocks (PHC). This is a trade-off. Hardware timestamping gives best time sync accuracy [OBAU18], but it also requires specific hardware for a clock.

We put an INT collector which sniffs network traffic at the TC just before the client. The collector collects INT data generated by the TCs to monitor, detect, and localize TDAs. The result is then graphically represented by the Grafana analytics and visualization software.

3.4.2 Accuracy of P4-based transparent clocks

In this section, we present an evaluation of our P4-based transparent clocks. The objective is not to evaluate LinuxPTP performance but the correctness of our TCs. We use the testbed presented in Figure 3-4 where we introduce 10 TCs between the client and server. Each test runs for 600 seconds and is repeated 10 times to ensure reliability. We used the default LinuxPTP configuration, logSyncInterval=0, which configures the server to send a sync message and its follow_up each second, and logMinDelayReqInterval=0, which configures the mean time interval of 1 second between Delay_Req messages sent by the client.



(a) P4-based TCs (b) LinuxPTP-based TCs

Figure 3-5: Clock offset of the client during PTP time synchronization across 10 transparent clocks (TCs)

To compare the performance, we conduct the same set of tests after replacing the P4-based TCs by the LinuxPTP-based TCs. Figure 3-5a and Figure 3-5b illustrate the clock offset between the client and server as reported by LinuxPTP when using P4-based TCs and LinuxPTP-based TCs, respectively.

The left side of each figure is a boxplot summarizing the distribution of clock offset values while the right side gives a detailed view of individual measurements over time. In the detailed view, each green dot corresponds to an individual offset measurement at a specific moment. The blue line shows the average clock offset over all test executions. To improve readability and reduce noise, the red line provides a smoothed version of the average, achieved using Python's NumPy convolve function with window size is 20. The chart initially shows a high clock offset, which quickly stabilizes after approximately 30 seconds. This rapid change is due to the initial system clock difference between the client and server. Following this, the client requires around 120 seconds to fully reach a steady state.

From these results, we observe that after the calibration phase, all clock offset values remain below 200 µs, aligning with the expected accuracy of LinuxPTP when using software timestamping. This confirms that our P4-based transparent clocks perform on par with LinuxPTP-based TCs and, most importantly, do not introduce any additional degradation to time synchronization performance.

3.4.3 TDA detection & localization

The TDA detection is passed through 3 phases, calibrating, learning, and monitoring, as described below.

- Calibrating phase takes place firstly when the client begins synchronizing with the server. During this phase, the clock offset values are relatively large and exhibit significant variation. Gradually, the clock offset decreases and stabilizes over time. Once this phase is complete, the client achieves synchronization with the server.
- Learning phase is executed once the client is synchronized with the server. During this phase, we assume that the time synchronization network works normally, without any TDA. The objective of this phase is to determine the maximum clock offset, thus, the accuracy supported by the network in normal condition. The accuracy must satisfy the given accuracy requirement T. We also determine the threshold ΔT which represents the maximum variation of IAT calculated by Equation (9). Thus this ΔT must not trigger the condition in Equation (10), i.e., we have $\Delta T \leq (2 * T + \sigma)$.
- Monitoring phase is performed to monitor the time synchronization network to detect and localize TDAs. We constantly calculate the new values of Δ. An alert is triggered if (i) it is greater than the threshold ΔT determined in the learning phrase above and (ii) it can cause a clock offset bigger than the given requirement, i.e., $\Delta > (2 * T + \sigma)$.

We show in Figure 3-6 a Wireshark representation of a delay_res message which has 3 TLV extensions to carry on INT data of 3 switches.

Each TLV includes a switch ID, Ingress and Egress timestamps, and a Correction value, representing timing data from three switches along the network path. This figure visually illustrates how INT data is integrated into PTP messages to enhance network visibility and enable precise delay tracking across multiple hops.

Figure 3-7 shows graphically our framework, which monitors, detects, and localizes TDAs at each PTP packet, the sync messages on the left side and the delay_req messages on the right side. The two charts on the first row represent the variation of IAT of messages at the server and each TC. The IATs are calculated by Equation (7). We can see that the interval between two consecutive sync messages is almost 1 second but this is not the case for the delay_req although the client is configured to send a delay_req each second.

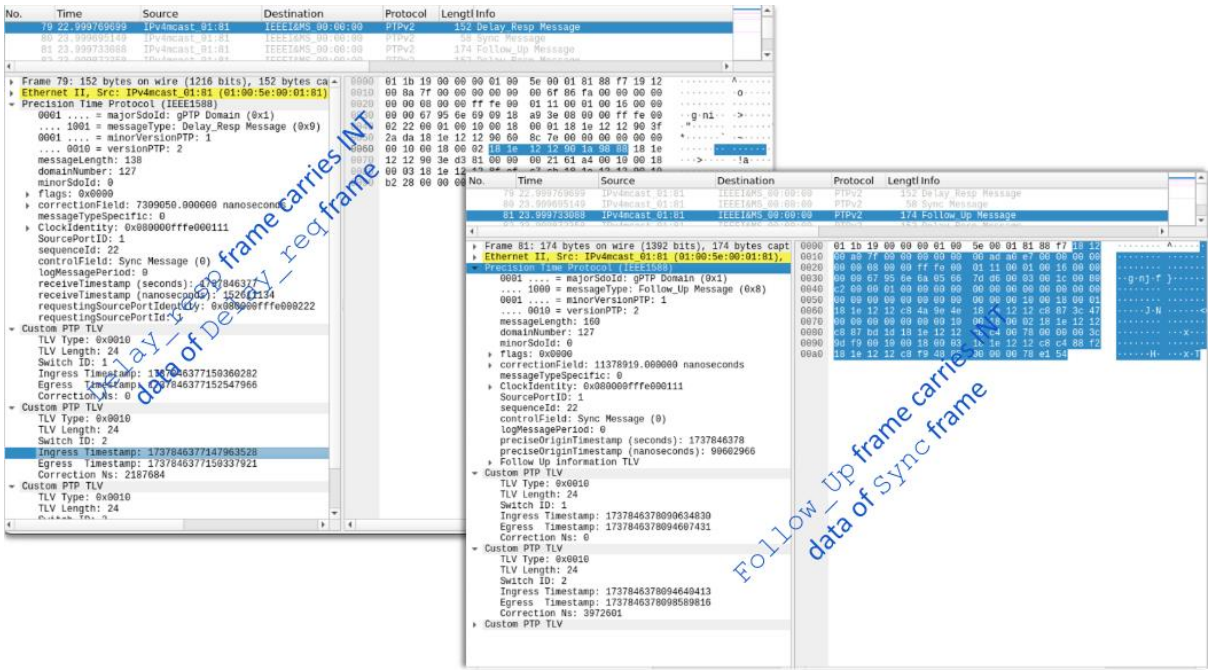


Figure 3-6: INT embedded in TLV extensions

The charts on the second row represent the difference of these IATs of each TC with the ones of the server. They represent the values calculated by Equation (9). The charts on the third row indicate whether there exists TDA on the network and its position is limited between the server and the impact TCs shown on the charts. The last row contains two charts that indicate the arrival time of sync packets, on the left side, and the depart time of delay_req packets, on the right side, at each transparent clock. Although these clocks may not be synchronized, the charts still can give an idea about the delays of packet propagating between them. It is worth noting that the TDA detection does not rely on the absolute timestamps but on their variation.

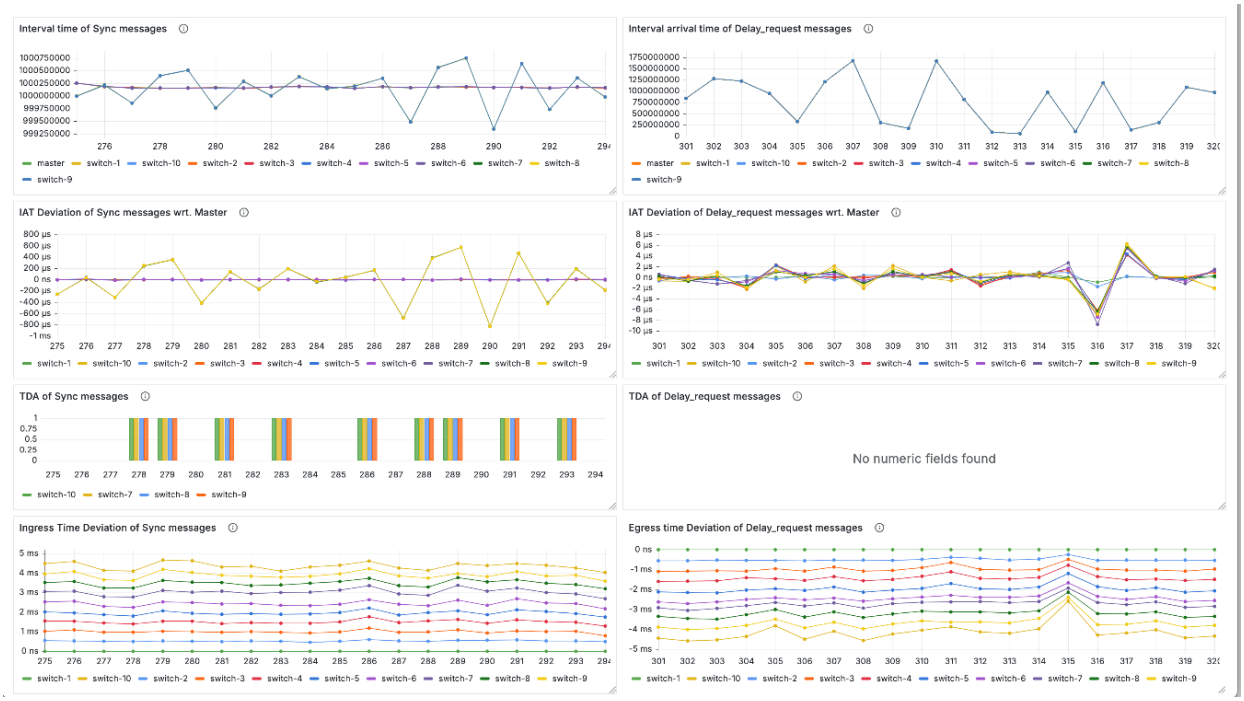


Figure 3-7: PTP monitoring, TDA detection, and localization

3.5 Key takeaways

We proposed a P4-based transparent clock implementation to enhance time synchronization in Precision Time Protocol (PTP) networks while maintaining compatibility with existing LinuxPTP solutions.

Our approach leverages Inband Network Telemetry (INT) to monitor, detect, and localize time-delay attacks (TDAs) in real-time without introducing additional probe traffic.

Experimental evaluations demonstrated that our P4-based transparent clocks achieve accuracy comparable to LinuxPTP-based implementations, with clock offset values remaining below 200 µs.

The security framework we designed includes a learning phase to establish a detection threshold, ensuring that anomalies in synchronization performance trigger alerts only when deviations exceed expected accuracy.

Our work contributes toward integrating time synchronization into software-defined networking (SDN) and network programmability, paving the way for more secure and flexible synchronization mechanisms.

4 Network delay reduction for improved time synchronization performance

4.1 Network delay reduction

The standard IEEE 802.1AS requires a network delay lower than 10 ms [IEEE 802.1AS] (considering also UE-to-UE communication). This requires that 5G should guarantee a maximum delay of 5 ms to always cover for the UE-to-UE communications. The 5G network provides different options to reduce network delay and improve the reliability of data session such as having a separate network slice or assigning QoS flow for PTP traffic. When the UE connects to the 5G core it establishes a packet data unit (PDU) session with a default flow. This primary flow can only be assigned best effort service without QoS. Therefore, to improve reliability, the first option consists of creating a secondary data flow within an established PDU session. This QoS flow with a 5QI can guarantee lower delay budget. Another option is to utilize network slicing to separate RAN, Transport and core resources for selected devices. This section provides an overview of those solutions and includes some performance results.

One generic approach which can be applied to any group of devices consists of network slicing. 3GPP does not specify the usage of network slicing for time synchronization but it is a generic solution that can be used to separate a group of devices that have to be synchronized. Network slicing [3GPP-TS23501, section 5.15] allows to group devices under a slice and separate the traffic from other best effort traffic allocated to the rest of devices without strict synchronization requirements.

Figure 4-1 shows a commercial deployment of network slices for grouping and separating the traffic of devices that require access to public Internet from other groups that require higher reliability and lower latency for time synchronization.

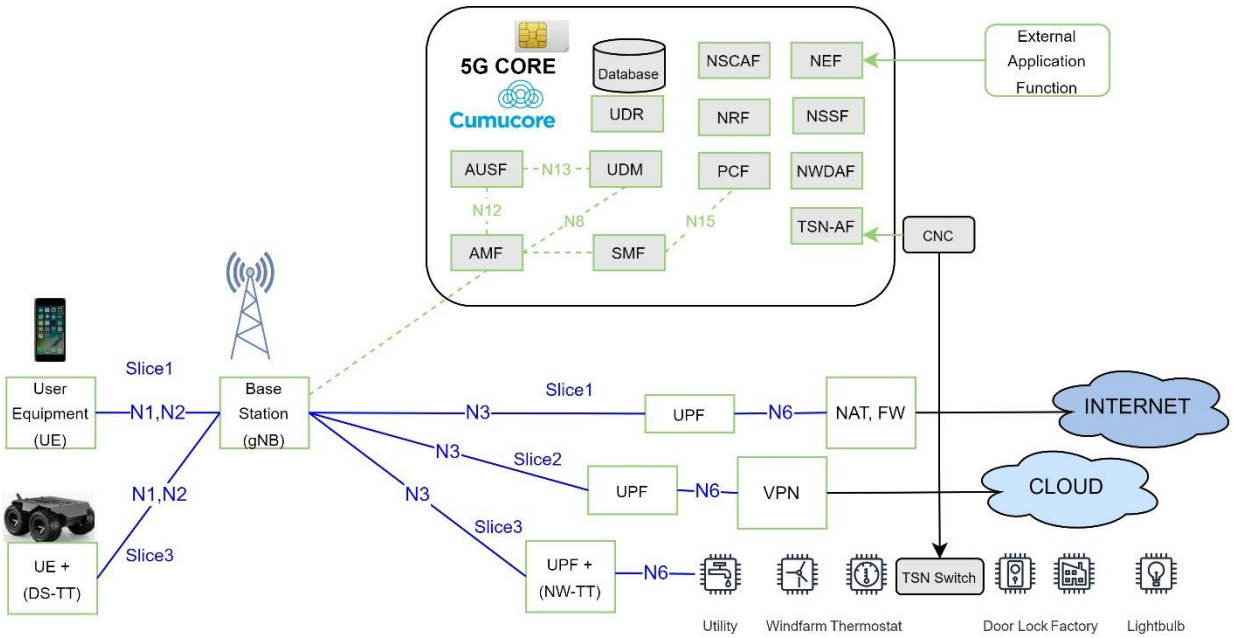


Figure 4-1: Network architecture with slices for Internet, Cloud and LAN Data Networks.

Another solution that is specified in 3GPP could be used to reduce delay budget for device communication that require accurate time synchronization. The solution consists of requesting a

second flow with QoS. The devices when registered to 5G networks will be allocated a data session with best effort or Non-Guaranteed Bit Rate. After the registration and default session is established, a secondary flow with higher QoS can be requested for some devices. The secondary flow can be requested by external application functions through Network Exposure Function (NEF) as shown in Figure 4-1.

In this section we present performance results obtained from devices that have been allocated a secondary flow with a standard 5G QoS flow identifier (5QI) = 83 instead of best effort 5QI=9.

After the UE has registered and successfully established a PDU session, the data transfer starts from the UE to the Data Network (DN). A successful connection between UE and DN requires two unidirectional generalized packet radio system (GPRS) tunneling protocol for user plane (GTP-U) tunnels for each UE between the gNB and the UPF. To establish the first tunnel to the gNB, the session setup messages sent from the gNB to the 5GS include the gNB's IP address and GTP-U Tunnel Endpoint (TEID) assigned by the gNB. The UPF will use that Tunnel ID (TEID) for sending the Downlink (DL) traffic to the UE via gNB. In the next session setup messages, the 5GS includes the UPF's IP address and GTP-U Tunnel Endpoint assigned by the UPF. The gNB will use this TEID to encapsulate the data that is sent Uplink (UL) from the UE to the DN via the UPF.

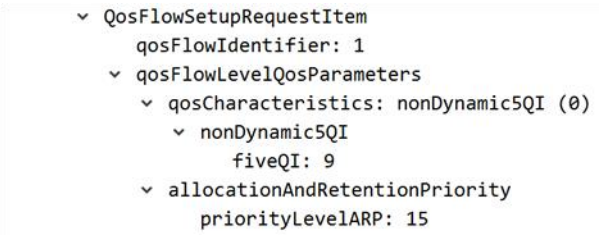


Figure 4-2: Wireshark message of mobile device session setup

In those messages as shown in Figure 4-2, the 5GS also includes QoS parameters that have been assigned in the UE profile stored in the User Data Management (UDM). Those parameters include the 5G Quality Indicator (5QI) to be used for the Packet Data Unit (PDU) session, Allocation and Retention Priority (ARP) i.e., priority level, the Guaranteed Flow Bit (GBR) rate i.e., the UE Aggregate Maximum Bir Rate (UE-AMBR) and the Quality Flow Identifier (QFI) that is included in the GTP headers to identify different flows from the same UE but different QoS.

In order to increase the reliability of sending time synchronization packets over the 5G network, a secondary flow can be assigned to the devices that will be synchronized. The assignment of QoS Flow Identifiers (QFI) to traffic flows ensures that service requirements are met based on the PTP needs. In this additional flow the 5QI would be increased to provide a GBR and delay budget.

This section presents the QoS monitoring (e.g., Packet Delay Budget (PDB), data rate, etc.) and the impact of different types of devices connected generating additional load for the overall system performance. In TS 22.261 [TS 22.261], the 3rd Generation Partnership Project (3GPP) defines service and operational requirements for various use cases in 5G systems. In this study, we focus on QoS monitoring and KPI evaluations for time synchronization that requires low delay and high reliability

for accurate synchronization. The PTP messages require low latency of ≤ 5ms E2E high reliability (99.9999%) to achieve a low synchronization offset (i.e. 1µs).

In order to obtain results closer to an industrial environment, the scope of our experiments includes a wider range of user equipment (UE) i.e., mobiles devices and routers (four in total). Adding different type of devices provides higher diversity to see the impact of the terminals in the UE-to-UE network delay. The measurement is done between two UEs but the others will still be connected to generate background traffic. We employ a specific TDD structure in contrast with experiments measuring the performance with a single device with good data transfer. Industry use cases produce more data for control actions that will interfere with the transfer of time synchronization messages. Furthermore, we compare two 5G QoS models defined by QoS Flows, based on different characteristics e.g., throughput, jitter, and Round-Trip Time (RTT). These measurements provide valuable insights about the alignment of 5G QoS parameters with the communication needs of Industry 4.0.

4.2 Experiment setup for enhanced QoS flow

The experimental framework involved the utilization of four distinct UE devices: Jetson Nano 2GB, Raspberry Pi 5, Nokia XR21, and a laptop. While the Nokia phone had direct connection to the network through SIM, other devices relied on Ethernet PDU using 5G-based routers. The experiment was structured into three phases, using the iperf3 tool as the application server to measure the end-to-end traffic performance.

At first, throughput measurements were collected with all devices connected simultaneously over one-hour period. The second phase targeted jitter measurements using UDP streams on three devices. Due to varying network conditions, the dataset involves between 6 and 24 hours of data collection. In the final step, RTT measurements were gathered using ICMP ping. This collection was conducted only with the Jetson Nano device, while other devices generated traffic to create network load, thereby assessing its impact on overall network latency.

All three phases were conducted for both Non-Guaranteed Bit Rate (Non-GBR) and assigning second flow with GBR 5QI configurations to analyze the variations and differences in performance relative to the requirements for delivering time synchronization. This comprehensive approach serves to evaluate the efficiency of QoS-based traffic and in fulfilling KPI requirements in use cases of URLLC for PTP packets.

In the experiment we used a commercial gNB with configuration in following Table 3.

Table 3: gNB configuration parameters

Features	Values
Band	N77
Frequency	3.9GHz
Carrier Bandwidth	100MHz
Transmit Power Range	40dBm per channel
Modulation scheme	256QAM
TDD DL/UL Configuration	2.5ms single periodicity DSUUU
SCS	30kHz

4.3 Enhanced QoS flow for network delay reduction

In 5G, QoS management plays a critical role in ensuring an efficient handling of diverse services and traffic types. One of the core mechanisms used to manage QoS is the 5QI to categorize traffic based on its requirements for latency, service type, throughput, etc. The 5QI is applied within PDU sessions, where QoS flows are mapped to Data Radio Bearers (DRBs) in the Radio Access Network (RAN). This mapping enhances the delivery of differentiated services to UEs based on pre-configured or dynamically updated QoS profiles. It describes forwarding treatments that packets should receive to improve reliability.

In our experiment we utilize the QoS parameters shown in the following Table 4.

Table 4: 5QI used for the performance measurements setup.

Type	5QI	PDB[1]	Packet Error rate	Maximum Data Burst	Services[3]
Non-GBR	9	300ms	10−6	N/A	Video, TCP-based
GBR[2]	83	10ms	10−4	1354 bytes	Discrete Automation

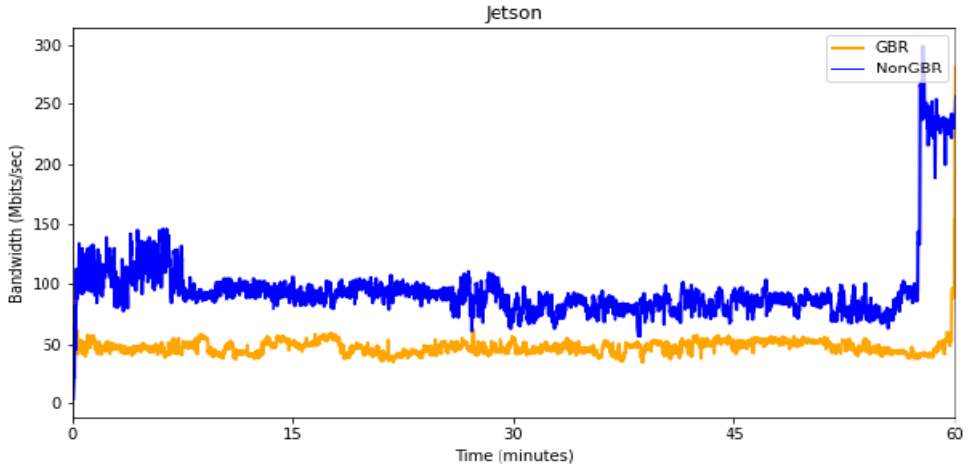
[1]PDB is the maximum allowable delay between the UE and the N6 termination point of UPF, which is our edge application. The defined value is irrespective of the direction of communication (i.e. DL/UL). It includes both Core Network (CN) PDB and RAN PDB obtained by substracting the CN PDB to the given PDB. For Non-GBR 5QI a CN PDB of 10 ms is applied, while for delay-critical GBR, the CN PDB is 1 ms. For GBR 5QI 83 that provides 10ms PDB is used because is the only one supported in commercial gNB used in the setup.

[2]Delay-critical GBR.

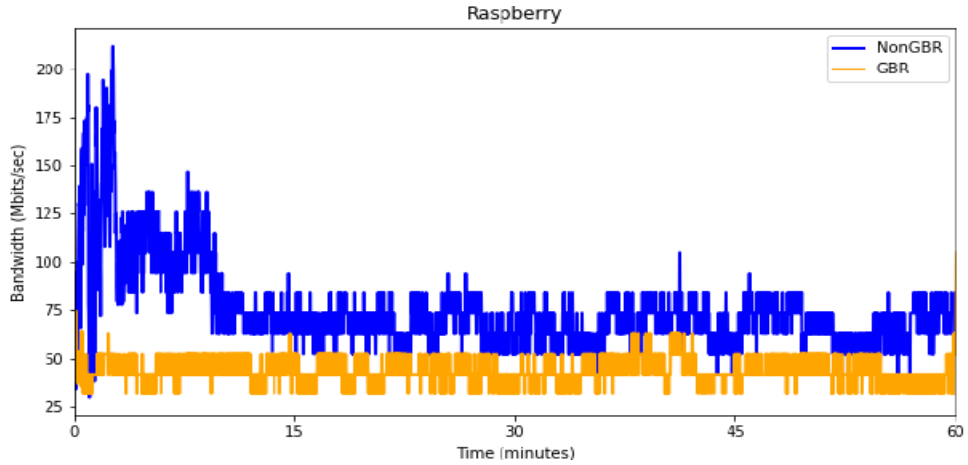
[3]Buffered streaming, www, e-mail, chat, FTP, p2p file sharing, progressive video, etc.

4.4 Enhanced QoS measurements results

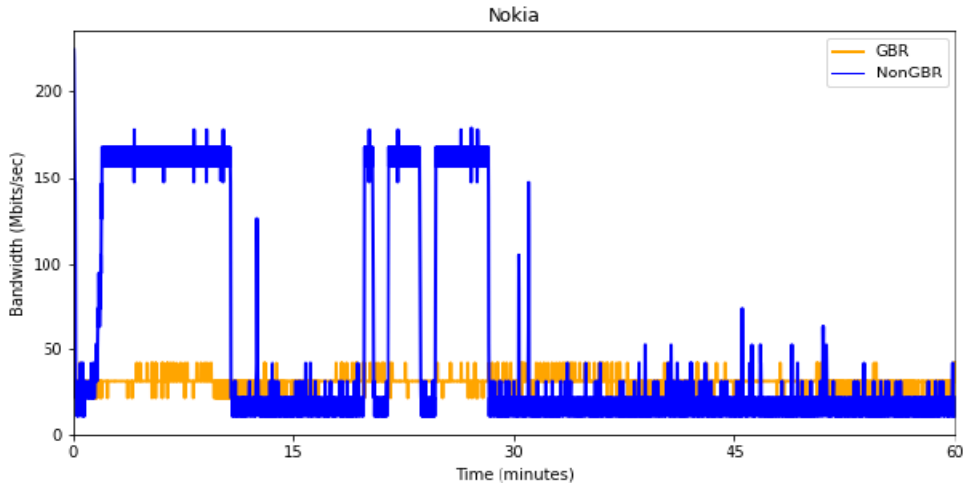
The measurements presented in Figure 4-3, the best-effort delivery, represented by the Non-GBR flow, demonstrates proactive resource allocation to UEs based on network conditions. It accounts for the observed fluctuations in the Non-GBR graphs. In contrast, with a GBR flow, the network ensures a balanced distribution of resources among UEs, as they receive a consistent and similar forwarding treatment. Minor variations are also noticed at the start and end of the measurements, with some UEs exceeding peak data rates of 250 Mbps. These variations correspond to slight delays or offsets, from few seconds to minutes, in the initiation or termination of data transmission between the UEs.



a)



b)



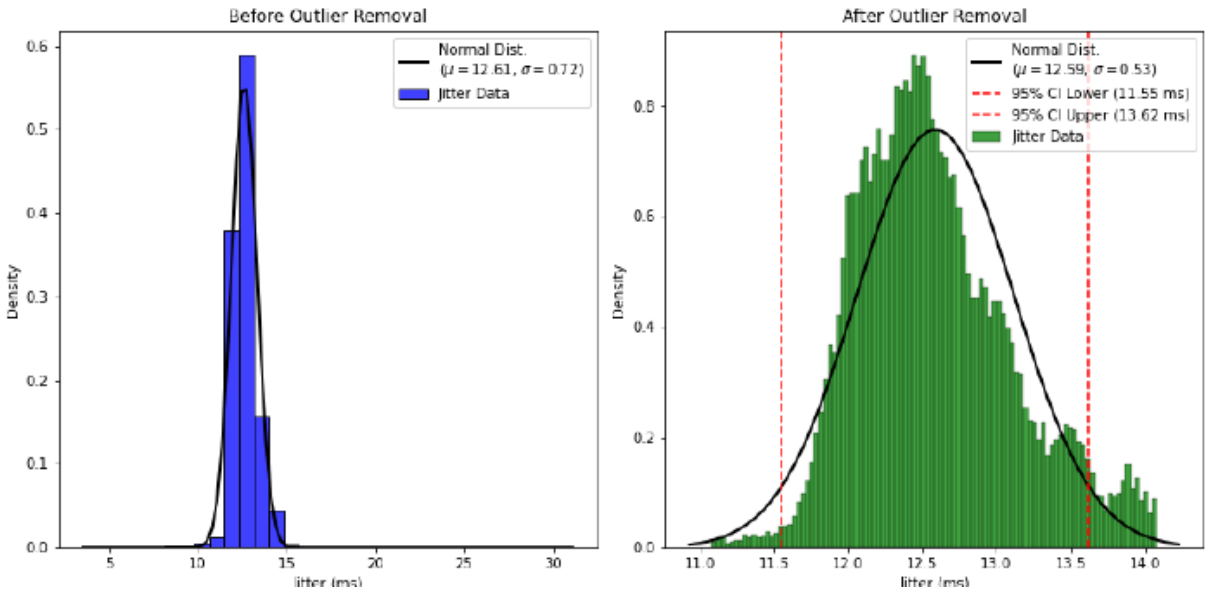
c)

Figure 4-3: Performance measurements for different devices with Non-Guaranteed Bit Rate (NonGBR) with 5Q1=9 and Guaranteed bit rate (GBR) 5QI=83 flows.

The results in Figure 4-3 show that second flow with GBR provides more reliable data transmission, which would ensure delivery of PTP packets across the 5G network. The GBR would provide sufficient throughput for PTP traffic with more reliable packet delivery. The measurements are performed to show the difference in UE to UE traffic depending on the type of device which would impact the packet reliability. The flow with QoS delivers a GBR with constant throughput and lower packet drops.

The other factor to consider is the delay variation since PTP has 10 ms requirement for the end-to-end delay. The packet delay is variation of packet arrival times over a network. In ideal network conditions, data packets delay variation is low. High variation in packet delay measurements might indicate network congestion or instability that might exceed the 10 ms limit for PTP packet delay. Iperf uses the interarrival estimation of jitter as in RFC 1889 to measure the jitter. It is calculated for UDP-based traffic with Real-time Transport Protocol (RTP) when the user device sends data packets every 1 s interval at a constant bit rate. Our goal in this experiment is to identify any variability of measurements between best-effort versus guaranteed service for the edge application.

Figure 4-4 shows the results packet delay where we can see approximated normal distributions of the packet delay when using a default Non-GBR flow. We present the original measurements without any pre-processing along with the results after outlier removal. Before outlier removal, we observe large and inconsistent fluctuations with spike up to four times the mean value. Next Figure 4-4 show the results for laptop, Nokia mobile and Raspberry Pi 5.0.



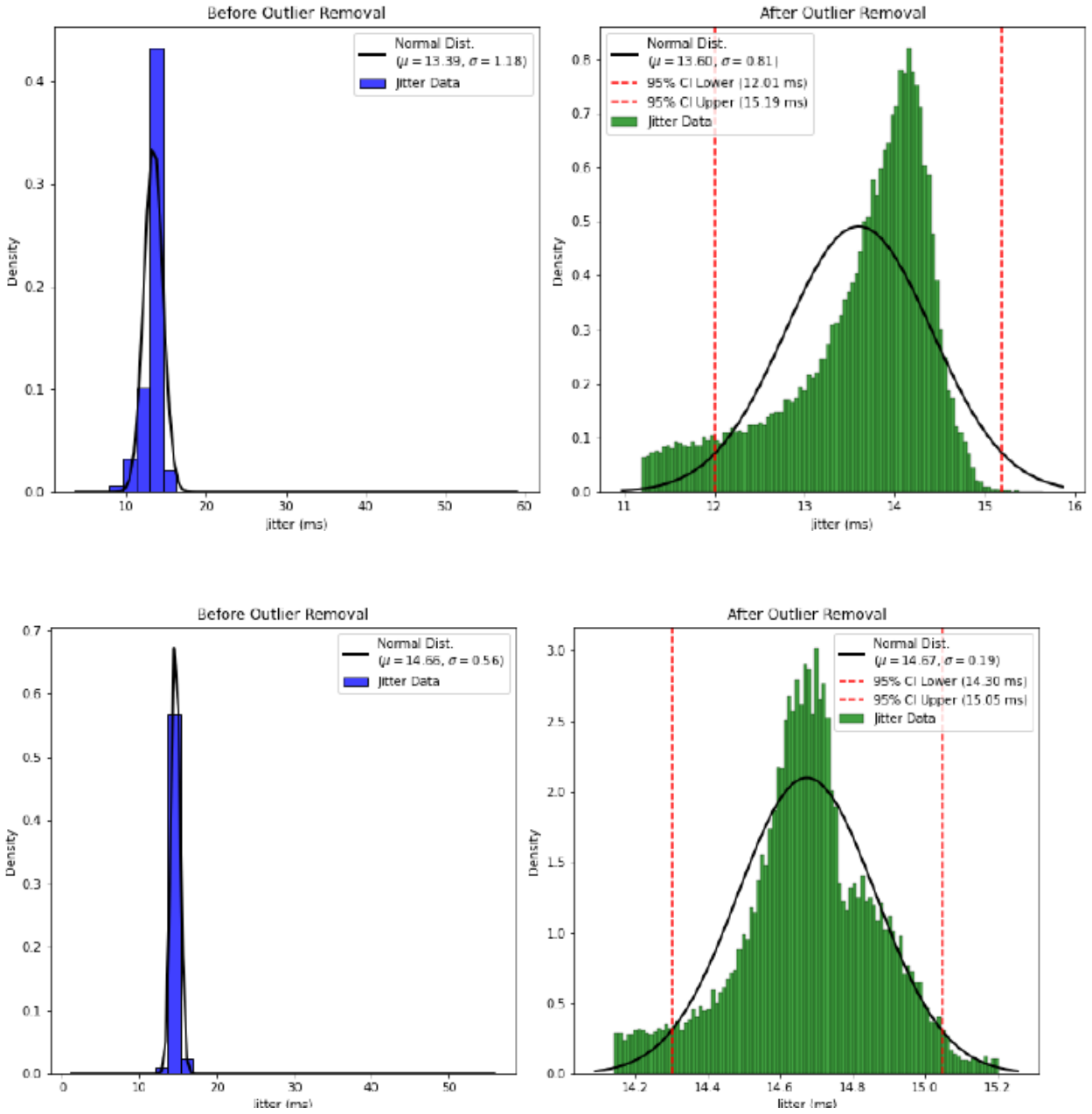
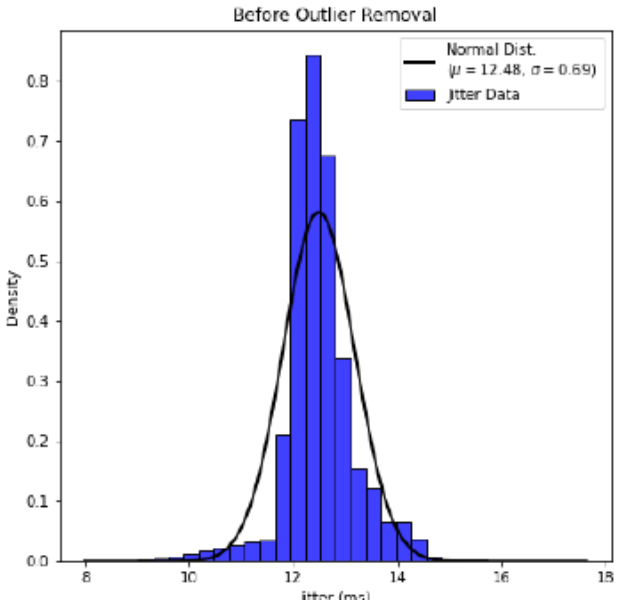
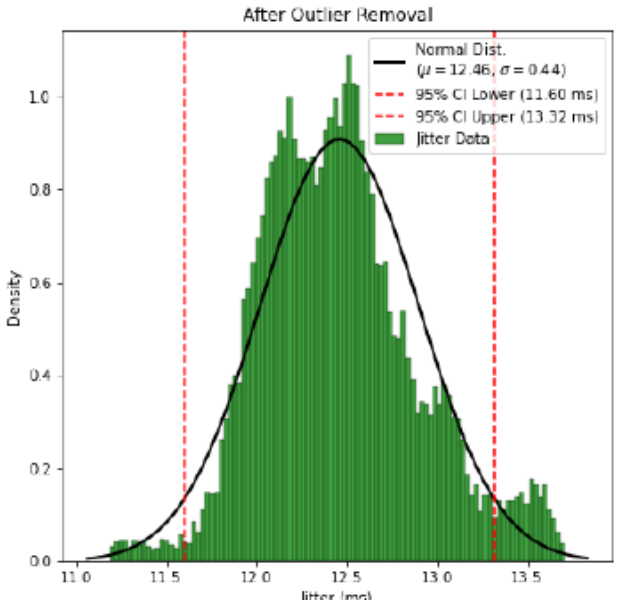
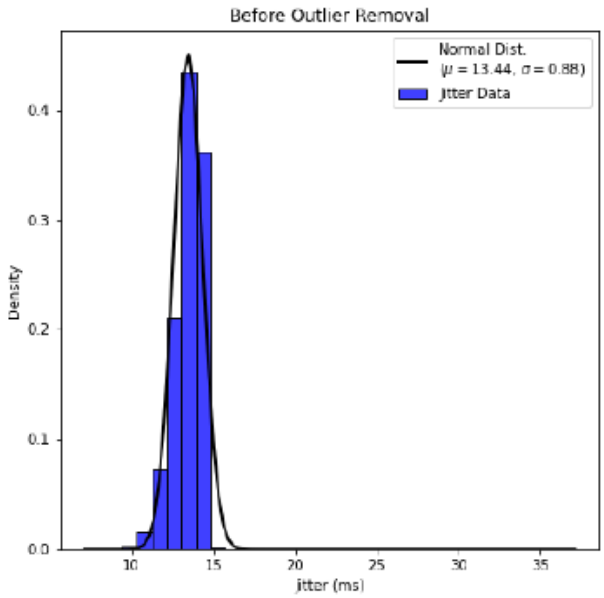


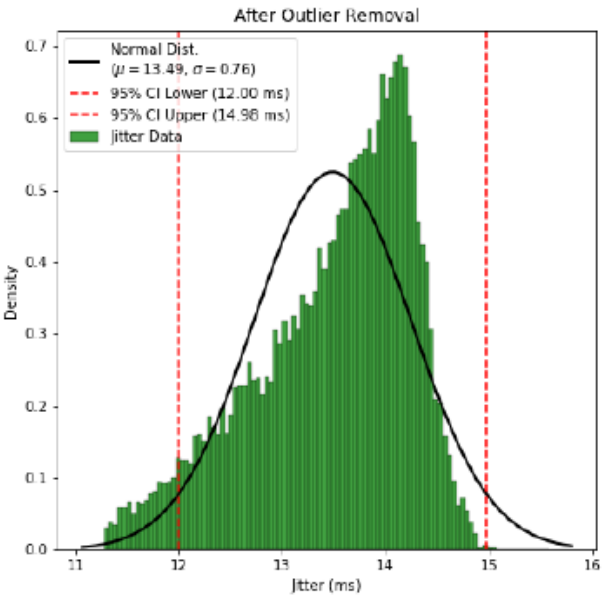
Figure 4-4: Delay distribution measured in mobile, laptop and rPI devices for non-GBR flow.

Figure 4-5 presents the fitted normal distribution of delay values measured using GBR flow. Prior to outlier removal, we observed that GBR resulted in smaller jitter deviations, average of 2ms, in contrast with Non-GBR. After removing outliers, the jitter variation under GBR became even more distinct, showing smaller fluctuations by only a few milliseconds compared to best-effort delivery.


Before Outlier Removal
Normal Dist.
($\mu = 12.48$, $\sigma = 0.69$)
Jitter Data
Density
jitter (ms)


After Outlier Removal
Normal Dist.
($\mu = 12.46$, $\sigma = 0.44$)
95% CI Lower (11.60 ms)
95% CI Upper (13.32 ms)
Jitter Data
Density
jitter (ms)


Before Outlier Removal
Normal Dist.
($\mu = 13.44$, $\sigma = 0.88$)
Jitter Data
Density
jitter (ms)


After Outlier Removal
Normal Dist.
($\mu = 13.49$, $\sigma = 0.76$)
95% CI Lower (12.00 ms)
95% CI Upper (14.98 ms)
Jitter Data
Density
Jitter (ms)

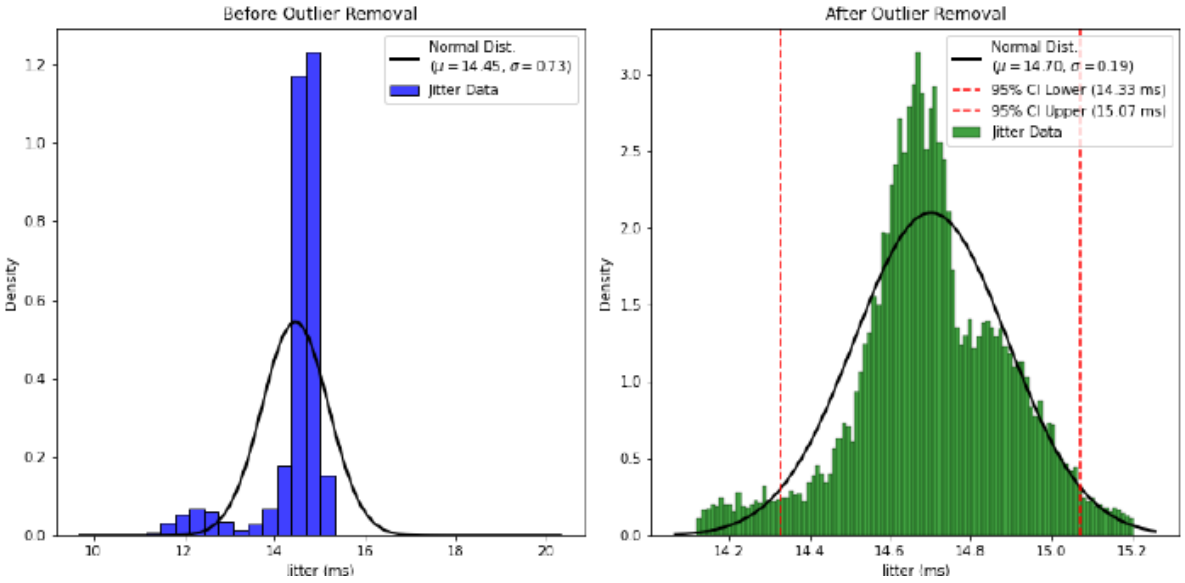


Figure 4-5: Jitter distribution measured in mobile, laptop and rPI devices for GBR flow.

The comparison of the jitter for default and secondary flow with GBR does not show major differences. Consistently with jitter results, there is minimal to no difference in latency between the QoS flows, as shown in Figure 4-5. The measurements were consistently taken using Jetson Nano device, while other UEs generated traffic load toward the edge server. The mean delay variation difference between GBR and non-GBR is ≤ 1 ms, slightly favoring GBR.

4.4.1 Key takeaways

In this subsection we presented the results when allocating a new flow with higher QoS to improve the reliability of the time synchronization messages.

The result shows there is an improvement with more constant throughput when using GBR compared to more stochastic with random peaks when using Non-GBR. However, the measurements show that there is only negligible improvement to the overall packet delay.

Therefore, the main takeaway is that adding flows with higher QoS would be relevant for data transfer that require deterministic throughput and as such will improve the reliability of transferring time synchronization information.

4.5 PTP measurements with enhanced QoS

In this section we measure the performance of time synchronization using the enhanced QoS based on second flow with GBR 5QI used in previous measurements.

Figure 4-6 shows the setup for measuring the time sync offset between the Grand Master clock located in the fixed network and the clock on the User Equipment (UE) with the 5G modem. The figure shows the UE with the modem integrated in a development board. The UE establishes Ethernet PDU session

and is connected to a TSN switch (i.e., Kontron 2) to send and receive the time synchronization messages over the 5G system. On the fixed network the 5G core includes the User Plane Function that supports 5GLAN and Ethernet PDU and is connected to another TSN switch (i.e., Kontron 1) that will send the Ethernet PTP messages to the UPF.

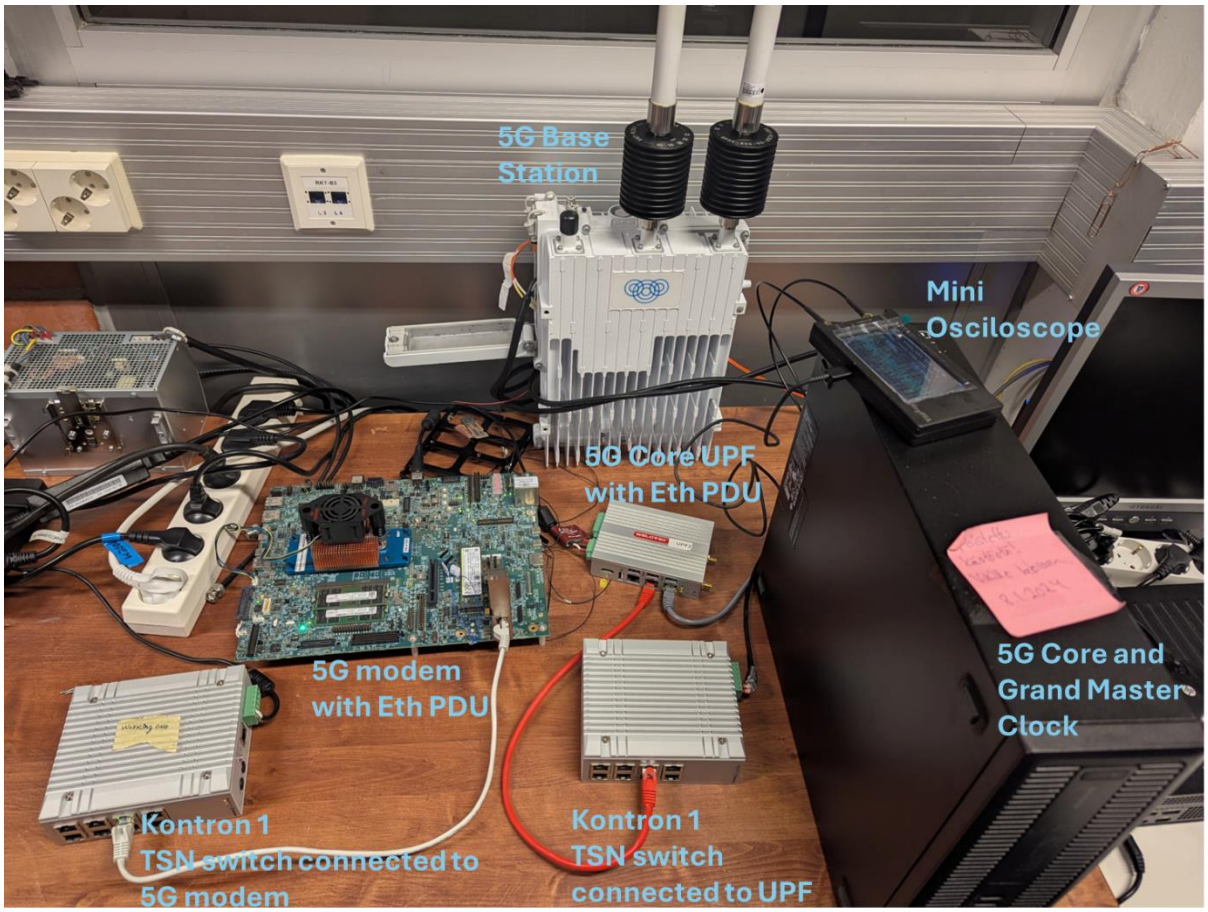


Figure 4-6: Setup with commercial off the shelf equipment for time synchronization measurements.

In this setup the objective is to measure the level of synchronization that can be achieved by using 5G system as transparent Ethernet switch with enhanced QoS where the PTP messages are send from fixed TSN network to 5G mobile devices connected to TSN devices.

Figure 4-7 shows the results of measuring the Pulse Per Signal (PPS) from the Grand Master on the fixed network and the PPS from the 5G mobile to see the synchronization offset when PTP messages are sent across the 5G system using enhanced PDU session. The results show that an offset of 41.3us is achieved which is good considering the gNB does not support 5QI with lower delay budget and the SIB9 is not used.

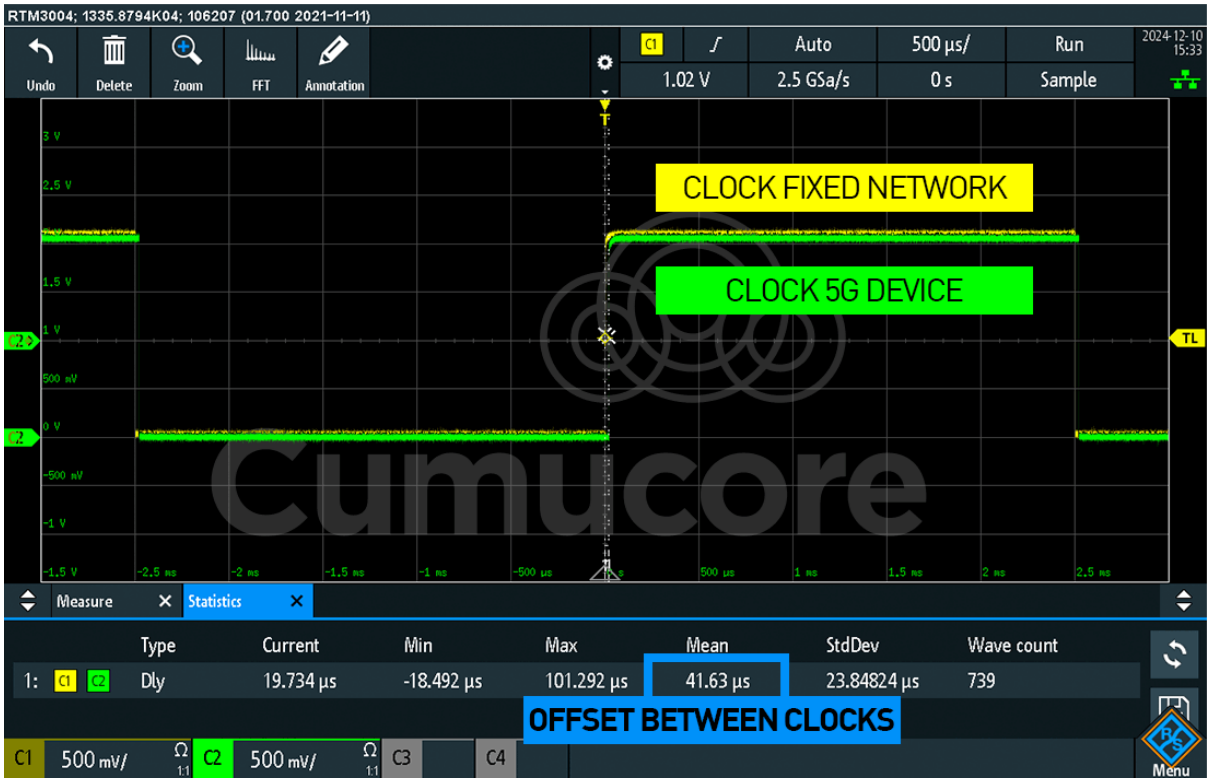


Figure 4-7: View of Rhode & Swartz oscilloscope measurement of PPS output between GM in fixed network and system clock on 5G modem.

4.5.1 Key takeaways

This section provides a real deployment using commercial equipment with additional flows for increasing the QoS and measuring the impact on time synchronization.

The experiment shows that time offset of 41 us is acceptable when time synchronization is delivered across the 5G system as data plane with higher QoS. However, the achieved offset is not sufficient for industrial environments that requires 1 us time offset. In this setup the 5G core includes the NW-TT that is adding the required time information, but the base station is not delivering the required reference time information, i.e., SIB9 that will allow the DS-TT to improve the time that would be required for the time synchronization to improve the accuracy.

5 Conclusion and future work

This deliverable analyzes multiple options for increasing the reliability of time synchronization, which is critical for TSN traffic scheduling.

The first approach consists of deploying hot-standby to increase time synchronization reliability. Therefore, in this deliverable we validate the improvements of hot-standby architectures using simulations in cooperation with WP4. Moreover, we analyze the techniques to minimize packet delay that can be applied to PTP traffic.

The second solution relies in securing the delivery of time synchronization and protecting it against attacks. The report proposes and evaluates in-band security framework's efficiency in cooperation with WP3. The deliverable evaluates the usage of the security framework to improve the resiliency of time synchronization delivery by using packet delaying attack scenario.

Finally, this report also discusses some options for improving time synchronization information using separate slices or allocating a different data flow with higher QoS to deliver time synchronization. A real testbed has been used to measure the time accuracy based on additional flows with enhanced QoS.

Moreover, future work would require the measurements of fully compliant 3GPP architecture proposed to reach 1 us time offset based on the necessary network functions and support from base station time reference.

References

[3GPP-TS23501]	3GPP, "TS 23.501: Technical specification group services and system aspects, system architecture for the 5G system (5GS)," Release-18, v18.0.0, 202
[AMS+21]	M. K. Atiq, R. Muzaffar, Ó. Seijo, I. Val, and H.-P. Bernhard, "When IEEE 802.11 and 5G meet time-sensitive networking," IEEE Open J. of the Ind. Electron. Soc., vol. 3, pp. 14–36, 2021.
[AS21]	Waleed Alghamdi and Michael Schukat. Precision time protocol attack strategies and their resistance to existing security extensions. Cybersecurity, 4(1), 2021. Publisher: Cybersecurity.
[BBF+13]	A. Bondavalli, F. Brancati, A. Flammini, and S. Rinaldi, "Master failure detection protocol in internal synchronization environment," IEEE Trans. on Instrum. and Meas., vol. 62, no. 1, pp. 4–12, 2013
[BDG+14]	Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming Protocol-Independent Packet Processors. Computer Communication Review, 44(3):87–95, 2014.
[CM10]	R. Cochran and C. Marinescu, "Design and implementation of a PTP clock infrastructure for the Linux kernel," in Proc. of ISPCS, pp. 116–121, IEEE, Sept. 2010.
[D6G-D1.1]	DETERMINISTIC6G deliverable D1.1, "DETERMINISTIC6G use cases and architecture principles", Jun. 2023
[D6G-D2.2]	DETERMINISTIC6G deliverable D2.2, "First Report on the time synchronization for E2E time awareness", Dec. 2023
[D6G-D3.2]	DETERMINISTIC6G deliverable D3.2, "Report on the Security solutions (Software)", Dec. 2023
[D6G-D4.4]	DETERMINISTIC6G deliverable D4.4, "Digest on Second 6GDetCom Simulator & Emulator Release", April 2024
[FOZZ+24]	David Franco, Eder Ollora Zaballa, Mingyuan Zang, Asier Atutxa, Jorge Sasiain, Aleksander Pruski, Elisa Rojas, Marivi Higuero, and Eduardo Jacob. A comprehensive latency profiling study of the Tofino P4 programmable ASIC-based hardware. Computer Communications, 218:14–30, March 2024.
[HDM+23]	L. Haug, F. Dürr, S. S. Mostafavi, G. P. Sharma, J. Sachs, J. Harmatos, J. Costa-Requena, and J. Ansari, "Deterministic6g/deterministic6g_data: D4.1 - first detcom simulator framework release (datasets)," Dec. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.10405085
[IEEE 1588-2019]	EEE Standard 1588-2019. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. Technical report, 2020. (Revision of IEEE Std 1588-2008).
[IEEE 1588g-2022]	IEEE, "IEEE Std 1588g-2022: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Amendment 2: Master-Slave Optional Alternative Terminology," 2022
[IEEE 802.1AS]	IEEE, "IEEE Std 802.1AS-2020: IEEE standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications," 2020

[IEEE 802.1ASdm]	IEEE, “IEEE Std 802.1ASdm-2024: IEEE standard for local and metropolitan area networks–timing and synchronization for time-sensitive applications amendment 3: Hot standby and clock drift error reduction,” 2024.
[IW20]	Eyal Itkin and Avishai Wool. A Security Analysis and Revised Security Extension for the Precision Time Protocol. IEEE Transactions on Dependable and Secure Computing, 17(1):22– 34, January 2020.
[KJC19]	Pravein Govindan Kannan, Raj Joshi, and Mun Choon Chan. Precise Time-synchronization in the Data-Plane using Pro- grammable Switching ASICs. SOSR 2019 - Proceedings of the 2019 ACM Symposium on SDN Research, (2):8–20, 2019. ISBN: 9781450367103.
[MBB+23]	Sadok Mehdi Mazigh, Marcel Beausencourt, Max Julius Bode, and Thomas Scheffler. Using P4-INT on Tofino for measuring device performance characteristics in a network lab. In Proc. of WueWoWAS2023, pages 1–4, 2023.
[Miz14]	T. Mizrahi. RFC 7384 - Security Requirements of Time Protocols in Packet Switched Networks. Technical report, 2014. ISSN 2070-1721.
[OBAU18]	L. Ong Boon, K. Anil, and S. Usman, "A comparative analysis of Precision Time Protocol in native, virtual machines and container-based environments for consolidating automotive workloads," tech. rep., 2018. IEEE-SA Ethernet & IP @ Automotive Technology Day.
[TS 22.261]	3GPP “5G; Service requirements for the 5G system” version 17.11.0 Release 17
[TSZ+21]	Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. In-band Network Telemetry: A Survey. Computer Networks, 186(December), 2021.

List of abbreviations

Acronym	Explanation
5G	Fifth generation
5G-Adv	5G Advanced
5GC	5G core
5GS	5G system
6G	Sixth generation
AF	Application function
AI	Artificial intelligence
AR	Augmented reality
AVB	Audio video bridging
BTCA	Best timeTransmitter clock algorithm
CRR	Cumulative rate ratio
DetNet	Deterministic networking
DL	downlink
DoS	Denial-of-service
DS-TT	Device side TSN translator
E2E	End-to-end
GM	Grandmaster
gNB	Next generation NodeB
GNSS	Global navigation satellite system
GPS	Global positioning system
gPTP	Generic precision time protocol
IAT	Inter-Arrival Time
INT	In-band Network Telemetry
IoT	Internet of things
LAN	Local area networks
MIMO	multiple-input multiple-output
MITM	Man-In-The-Middle
ML	Machine learning
MTU	Maximum Transmission Unit
NRR	Neighbor rate ratio
NTP	Network time protocol
NW-TT	Network side TSN translator
P4	Programming Protocol-independent Packet Processors
PDU	Protocol data unit
PHC	PTP Hardware Clock

PMIC	Port management information
PSFP	Per-stream filtering and policing
PTP	Precision time protocol
PTP-tele	PTP telecom
QoS	Quality of service
RAN	Radio access network
RRC	Radio resource control
SDN	Software Defined Networking
SIBs	System information blocks
TC	Transparent Clock
TDA	Time-Delay Attack
TLV	Type-Length-Value
TSN	Time sensitive networking
tR	timeReceiver
TSCTSF	Time sensitive communications time sensitive function
TSe	Egress timestamp
TSi	Ingress timestamp
tT	timeTransmitter
UE	User equipment
UL	Uplink
UMIC	User management information
UPF	User plane function
UTC	Coordinated universal time
UU	Radio interface
VR	Virtual reality
XR	Extended reality