Deliverable D1.4

Year 2 Report on Requirements, Architecture and Performance Evaluation

Editor SPCOM (UPC)

Contributors (TID), (IKL), (SEDS), (ELI), (UPC)

Version 2.1

Date March, 2024

Distribution PUBLIC (PU)

Towards a smart and efficient telecom infrastructure meeting current and future industry needs (TIMING-1)

(Ref. TSI-063000-2021-145)

DISCLAIMER

REVISION HISTORY

Version	*Date*	*Responsible*	*Comment*
0.0	December, 6, 2023	Olga Muñoz	Table of contents
1.0	February, 26, 2024	All authors	Complete version
1.5	March, 8, 2024	Olga Muñoz	First Integrated version
2.0	March 22, 2024	Olga Muñoz	Reviewed version
2.1	March 29, 2024	Luis Velasco	Final reviewed version

LIST OF AUTHORS

Partner ACRONYM	*Partner FULL NAME*	*Name & Surname*
UPC	*Universitat Politècnica de Catalunya*	*M. Ruiz, S. Spadaro, L. Velasco, D. Careglio, J. Comellas, F. Agraz, A. Pagès, J. Villares, M. Cabrera, O. Muñoz, J. Vidal*
TID	*Telefonica I+D*	*L.M. Contreras, M. Blanco, J. Folgueira, R. López*
IKL	*Ikerlan*	*G. Ros, R. Torrego*
SEDS	*Safran Electronics & Defense*	*J. Sánchez*
ELI	*E-lighthouse*	*J. Moreno, P. Pavón, A. Buendía*

GLOSSARY

Abreviations/Acronym	*Descriptions*
5GPPP	5G Public Private Partnership
AGV	Automated Guided Vehicles
AI	Artificial Intelligence
AP	Access Point
API	Application Programming Interface
ARM	Advanced RISC Machines
BE	Best Effort
CM	Connectivity Manager
CNC	Central Network Configuration
CSI	Channel State Information
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
DetNet	Deterministic Networks
DL	Downlink
DMA	Direct Memory Access
DT	Digital Twin
E2E	End-to-End
ENP	E-lighthouse Network Planner
ETSI	European Telecommunication Standards Institute
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
FRER	Frame Replication and Elimination for Reliability
GCL	Gate Control List
GUI	Graphical User Interface
gRPC	High Performance Remote Procedure Calls
HTTP	Hypertext Transfer Protocol
i4.0	Industry 4.0
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IIOT	Industrial Internet of Things
IP	Internet Protocol
KPI	Key Performance Indicators
LAN	Local Area Networks
LSP	Label Switched Paths
MAC	Multiple Access Control
MCS	Modulation and Coding Scheme
MEC	Multi-Access Edge Computing
MM	Monitoring Manager

MPLS	Multiprotocol Label Switching
NBI	Northbound Interface
NFV	Network Function Virtualization
NSC	Network Slice Controller
OFDMA	Orthogonal Frequency Division Multiple Access
OLS	Open Line System
OSS/BSS	Operations Support System and Business Support System
OTN	Open Transport Network
PCS/PMA	Physical Coding Sublayer/Physical Medium Attachment
PER	Packet Error Rate
PHC	PTP Hardware Clock
PTP	Precision Time Protocol
QoS	Quality of Service
QSI	Queue State Information
RAN	Radio Access Network
RB	Resource Block
RRM	Radio Resource Management
RT	Real Time
SBI	Southbound Interface
SDN	Software Defined Networks
SDP	Slice Demarcation Points
SLA	Service Level Agreement
SLO	Service Level Objectives
SNMP	Simple Network Management Protocol
STA	Station in a Wi-Fi network
STF	Short Training Field
TCP	Transmission Control Protocol
TDD	Time Division Duplexing
TDMA	Time Division Multiple Access
TE	Traffic Engineering
TFS	Teraflow SDN
TIF	Telecom Infra Project
TM	Topology Manager
TSN	Time Sensitive Networks
TSU	Time Stamping Units
UL	Uplink
URLLC	Ultra-reliable Low-latency Communications
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

EXECUTIVE SUMMARY

This document updates deliverable D1.2 with the activities carried out within WP1 during the second year of the TIMING project, providing updated specifications of the TIMING components and preliminary performance evaluations.

For the **Wi-Fi Time-Sensitive Network (TSN) nodes**, the derivable includes new requirements that arose during the design and implementation of the complete TIMING architecture, along with updated specifications to fulfil such new requirements, which involve the definition of new features and blocks. Moreover, the deliverable presents three tests to evaluate performance requirements related to superframe configuration and latency, wireless hop synchronization, and packet error rate.

Within the **Ethernet-based segment** of the TIMING TSN, the Central Network Configuration (CNC) component is an adaptation of Safran's proprietary solution that has been tailored to meet the main requirements outlined in TIMING. This deliverable includes a detailed explanation of the structure and components of the CNC system, the current status of these components, and some performance evaluation tests carried out.

The **TSN controller** serves as the intermediary between the upper-level entities, such as the Connectivity Manager (CM), and the data plane elements, specifically the wired TSN-enabled switches and wireless Access Points (APs). Previous deliverable D1.2 reported the general architecture of the SDN-TSN controller as well as the main models employed for the representation of the abstract topology and device capabilities. The current deliverable focuses on the developments and tests of the internal structure of the SDN-TSN controller.

The **TSN Connectivity Manager (CM)** is composed of two main functional blocks: the network planner, which is an extension of the capabilities provided by the E-Lighthouse Network Planner, and the core module, devoted to coordinating the actions targeted to orchestrate end-to-end (e2e) TSN flows across networks. Compared with the content in deliverable D1.2, this deliverable describes the latest version of the TSN CM architecture that can be organized into three main pillars: i) an update in the architecture, ii) restructuration of the CM NBI; and iii) enhancements in the GUI.

For the **scheduling component** of the TIMING project, this deliverable includes the design of TSN Wi-Fi windows for isochronous traffic (not reported in the previous deliverable D1.2), the design of a reinforced learning-based DL/UL splitter, and the scheduling of asynchronous traffic. Additionally, the description of four performance evaluation tests and their evaluations are included.

During the execution of the project, we realized the **need to extend the control plane architecture and add a novel** component to deal with network-wide resource allocation. Consequently, this deliverable extends the content in D1.1 and D1.2 by introducing a new *Time Sensitive Flow Scheduler Planner* (**TS-FSP**) component in the provisioning workflow and formally describing the optimization problem to be solved within the TS-FSP. Furthermore, the deliverable highlights the relation of this new module with the Digital Twin (DT), includes extensions and updates of the DT modules, particularly for traffic generation and queueing models, and provides illustrative results that validate the performance of the DT.

TABLE OF CONTENTS

EXECUTIVE SUMMARY 7
1 INTRODUCTION 1
2 DEPLOYMENT SCENARIOS, TRAFFIC CHARACTERIZATION, AND SERVICE REQUIREMENTS ... 1
3 WI-FI TSN NODES 2
3.1 Updated requirements 3
3.2 Updated specifications 3
3.3 Performance evaluation tests 4
3.3.1 Superframe configuration and latency test 4
3.3.2 Wireless hop synchronization test 5
3.3.3 PER test 6
4 ETHERNET TSN NODES 7
4.1 Structure and components 7
4.1.1 The Southbound API between the CNC and the TSN switches 8
4.1.2 The Northbound API between the CNC and the TSN Controller 9
4.1.3 The CNC Monitoring component 11
4.2 Current status 12
4.3 Testing 13
4.3.1 Test 003: SafranCNC-Monitoring_DeliverStats_1 13
4.3.2 Test 004: SafranCNC-SBI_configureVLAN_1 14
4.3.3 Test 005: SafranCNC-SBI_configureTAS_1 16
4.3.4 Test 006: SafranCNC-NBI_Latency_1 17
5 TSN CONTROLLER 19
5.1 Updates of the Architecture 20
5.2 Test and Evaluation 21
6 TSN CONNECTIVITY MANAGER 23
6.1 Updates of the Architecture 24
6.2 Updates of the TSM CM NBI 26
6.2.1 Controllers 27
6.2.2 Topologies 28
6.2.3 Provisioning 28
6.3 Updates of the ENP Front-END (GUI) 28
6.3.1 Topological Information Visualization Updates 29
6.3.2 Control Panel Upgrades 30
6.3.3 Summary Report Enhancements 31
7 SCHEDULING ALGORITHMS 32

7.1 TSN Windows Design for Isochronous Traffic 32
7.1.1 Botton-up design (assuming a compatible TSN Ethernet will be feasible) 34
7.1.2 Top-down design (design compatible with pre-defined TSN windows) 37
7.1.3 Extension to the Uplink 38
7.1.4 Preliminary Performance Evaluation 39
7.2 DL/UL splitter: updated specifications 43
7.2.1 States and Actions 43
7.2.2 RL based DL/UL splitter 45
7.2.3 Preliminary Performance Evaluation 46
7.3 Selected Algorithms for the Wireless Flow Scheduler (WFS) 49
7.3.1 TSN Windows Design for Isochronous Traffic 49
7.3.2 UL/DL Splitter for Asynchronous Traffic 50
7.3.3 Asynchronous Traffic DL Scheduling 51
7.3.4 Asynchronous Traffic UL Scheduling 51
7.3.5 Preliminary Performance Evaluation 52
7.3.6 Test 3 description 53
7.3.7 Simulation results (Test 3) 56
7.3.8 Test 4 description 62
7.3.9 Simulation results (Test 4) 62
8 DT AND KPI ESTIMATION 65
8.1 Provisioning workflow updates 65
8.2 TS-FSP Problem Definition 66
8.3 DT Extensions and Updates 67
8.3.1 Traffic generators 67
8.3.2 Queue models 68
8.3.3 KPI estimation 69
8.4 Illustrative Results 71
9 CONCLUSIONS AND OUTLOOK 74
10 REFERENCES 74

1 INTRODUCTION

This document summarizes the activities carried out during the second year of the project related to WP1, devoted to providing updated specifications and preliminary performance evaluations of the TIMING components.

Section 2 contains an overview of typical deployment scenarios, including the expected number of devices and data rates, and a summary of the mix of traffic types that a TSN-based industrial communication network could face, along with the service requirements of the different traffic types.

The rest of the sections focus on the different TIMING components:

- The Wi-Fi TSN nodes that provide a wireless extension to the wired TSN data plane of the TIMING architecture.
- The Ethernet TSN nodes that provide the main operational settings to Safran's Ethernet-based segment of the TIMING TSN network and gather the telemetry and system status indicators.
- The TSN controller that includes specialized interfaces to the TSN nodes and the TSN connectivity manager, and is crucial to enable the connectivity manager's provisioning decisions.
- The TSN Connectivity manager (CM) that is a pivotal component in the TIMING architecture, as it is positioned at the apex and interfaces with both TSN and Metro SDN Controllers and the Digital Twin.
- The wireless scheduler that allocates Wi-Fi resources for multiple concurrent services (TSN and BE) in downlink and uplink.
- The Digital Twin (DT) that, in the event of provisioning requests, provides KPI estimations to guarantee that TS flows meet the required performance upon acceptance of these requests.

This deliverable contains updated requirements and specifications for each component. Additionally, the deliverable describes performance tests and includes preliminary results corresponding to these tests.

2 DEPLOYMENT SCENARIOS, TRAFFIC CHARACTERIZATION, AND SERVICE REQUIREMENTS

5G-ACIA has produced a whitepaper [5G-19] with traffic modelling and deployment scenarios.

In such document, a typical deployment scenario is described for a factory. The exemplary case describes the layout of a typical factory for discrete production and assembly. It comprises a production area, assembly areas, a warehouse, a commissioning space and office cubicles, spanning a total of approximately 15,000 square meters with a ceiling 30 m high.

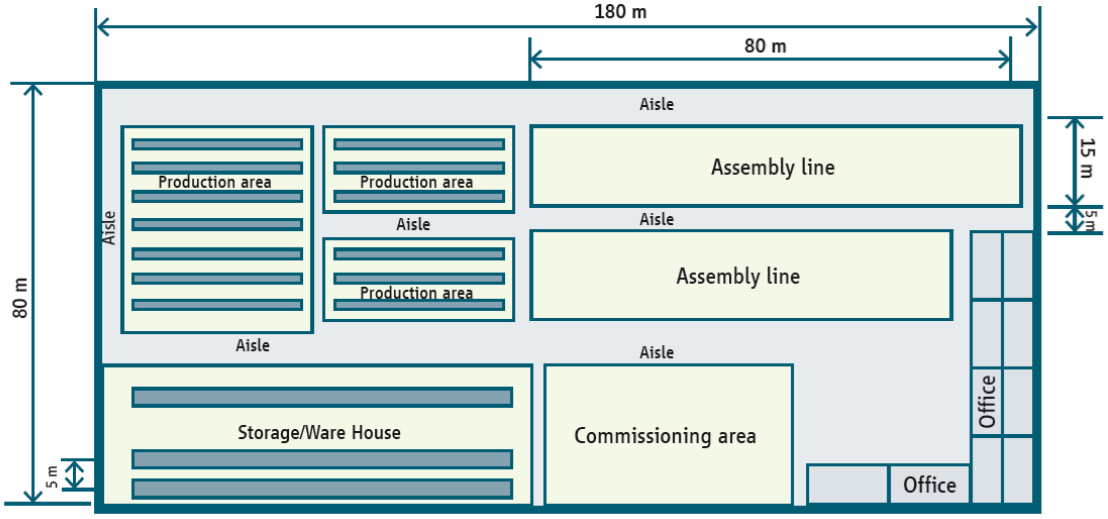

Figure 1. Layout of a typical factory [5G-19]

The whitepaper also proposes the expected device density for the use cases contained in the document, aligned with 3GPP TS 22.104 specification. Three deployment scenarios are considered:

- Small-scale scenario. In this context reflects an initial roll-out of half the number of devices (half the density) defined in TS 22.104, typical to a brown-field factory.
- Large-scale scenario. It corresponds to the wide deployment of 5G-based devices in a new or highly flexible factory with modular production cells.
- Inbound logistics scenario. It reflects the deployment of many more mobile robots and AGVs but on the other hand fewer motion-control use cases. This last scenario is the one closest to the TIMING use case.

Table 1 shows the expected number of devices and the expected data rates.

Use case	Device density [#/sqm]	# of devices	UL traffic [Mbit/s]	DL traffic [Mbit/s]
Small scale deployment scenario				
Process Monitoring	n/a	20		
Close-Loop-Control	0.002	15		
Motion Control	0.2	150		
Mobile Robot	0.001	8		
HMI	(note 1)	7		
Control-Control 100	0.003	0		
			105	**300**
Large scale deployment scenario				
Process Monitoring	n/a	40		
Close-Loop-Control	0.002	30		
Motion Control	0.2	300		
Mobile Robot	0.001	40		
HMI	n/a (note 1)	14		
Control-Control 100	0.003	2		
			550	**730**
Inbound logistics deployment scenario				
Process Monitoring	n/a	20		
Close-Loop-Control	0.002	30		
Motion Control	0.2	30		
Mobile Robot	0.001	40		
HMI	(note 1)	14		
Control-Control 100	0.003	0		
			410	**190**

Table 1. Expected number of devices and the expected data rates [5G-19]

Regarding TSN traffic characterization, 5G-ACIA [5G-21] summarizes the mix of various traffic types that a TSN-based industrial communication network could face. Service requirements range from best-effort traffic to critical real-time traffic.

Traffic types	Periodic / Sporadic	Typical period	Data delivery guarantee	Tolerance to Jitter	Tolerance to loss	Typical data size (Byte)	Criticality	Traffic priorities (VLAN PCP)	Strict priority IEEE 802.1Q	Redund-ancy IEEE 802.1CB	Time synch IEEE 802.1AS	Scheduled traffic IEEE 802.1Qbv	Frame preemption IEEE 802.1Qbu	PSFP IEEE 802.1Qci	TSN configuration IEEE 802.1Qcc
Isochronous	P	100 µs ~ 2 ms	Deadline	0	None	Fixed: 30 ~ 100	High	6	M	O	Yes	M		M(T)	M
Cyclic - Synchronous	P	500 µs ~ 1 ms	latency bound (τ)	≤ τ	None	Fixed: 50 ~ 1000	High	5	M	O	Yes	M		M(T)	M
Cyclic - Asynchronous	P	2 ms ~ 20 ms	latency bound (τ)	≤ τ	1 ~ 4 Frames	Fixed: 50 ~ 1000	High	5	M	O	No		R	M(R)	M
Events: control	S	10 ms ~ 50 ms	latency bound (τ)	n.a.	Yes	Variable: 100 ~ 200	High	4	M	O	No		O	M(R)	M
Events: alarm & operator commands	S	2 s	latency bound (τ)	n.a.	Yes	Variable: 100 ~1500	Medium	3	M	O	No		O	M(R)	M
Network control	P	50 ms ~ 1 s	throughput	Yes	Yes	Variable: 50 ~ 500	High	7	M	O	No				
Configuration & diagnostics	S	n.a.	throughput	n.a.	Yes	Variable: 500 ~ 1500	Medium	2	M				O	M(R)	M
Video	P	Frame Rate	throughput	n.a.	Yes	Variable: 1000 ~ 1500	Low	1	M	O	No		O	M(R)	M
Audio/Voice	P	Sample Rate	throughput	n.a.	Yes	Variable: 1000 ~1500	Low	1	M	O	No		O	M(R)	M
Best effort	S	n.a.	None	n.a.	Yes	Variable: 30 ~ 1500	Low	0	M				O		

Table 2. Service requirements for different traffic types [5G-21]

The following notes apply in Table 2.

- Note 1: M: mandatory, (T): time-based policing, (R): rate-based policing, O: optional, R: recommended. Various organizations have proposed diverse traffic priority values that differ from those given here
- Note 2: Time synchronization refers to synchronization of data transmission time to the network cycle for synchronized TSN operation. In addition, some applications may require time synchronization via the network.
- Note 3: For camera-assisted control applications, camera traffic can be cyclic-asynchronous. Cameras are synchronized at application level with a required synchronicity in the range of 1µs-10µs. Camera traffic may produce higher data throughputs (e. g., 1080p / 30Hz / 8-bit pixel video corresponds to 500 Mbit/s).

Furthermore, the 5G-ACIA whitepaper [5G-19] is focused on traffic modelling for industrial use cases. This characterization considers both traffic-related and network-related attributes. Regarding traffic attributes, it considers the message size, the transfer interval and the data rate. On the other hand, regarding the network attributes, it considers the radio link direction and the communication paths. This document proposes traffic models for the following use cases:

- Motion control, mobile controller
- Closed loop control, mobile I/O gateway
- Process monitoring
- Mobile robot
- Human-machine interface (HMI)
- Closed-loop control for process automation
- Control-to-control

The mobile robot can be considered as the closest use case to the one in TIMING. The characterization provided is as follows:

Attributes		Logical Link 1		Logical Link 2		Logical Link 3		Logical Link 4	
		LSEP	LTEP	LSEP	LTEP	LSEP	LTEP	LSEP	LTEP
Periodic Traffic	Transfer Interval					10,00	10,00	1,00	1,00
	Message Size					64	64	64	64
Aperiodic and Burst Traffic	Data Rate								
	average	8.000,00	10,00	500,00	1,00				
	peak	8.000,00	10,00	4.000,00	10,00				
All Traffic	Radio Link Direction	uplink	downlink	uplink	downlink	uplink	downlink	uplink	downlink
	Communication Path	P2P	P2P	P2P	P2P	P2P	P2P	P2P	P2P

Traffic volume (kbit/s)	
Sum uplink	9.050,00
Sum downlink	561,00

Table 3. Traffic characterization for the mobile robot use case.

3 WI-FI TSN NODES

Wi-Fi TSN nodes provide a wireless extension to the wired TSN data plane of the TIMING architecture. As already presented in previous deliverables, this Wi-Fi TSN extension is made of one Access Point (AP) and one or several nodes/stations (STA) which implement the following HW/SW architecture:

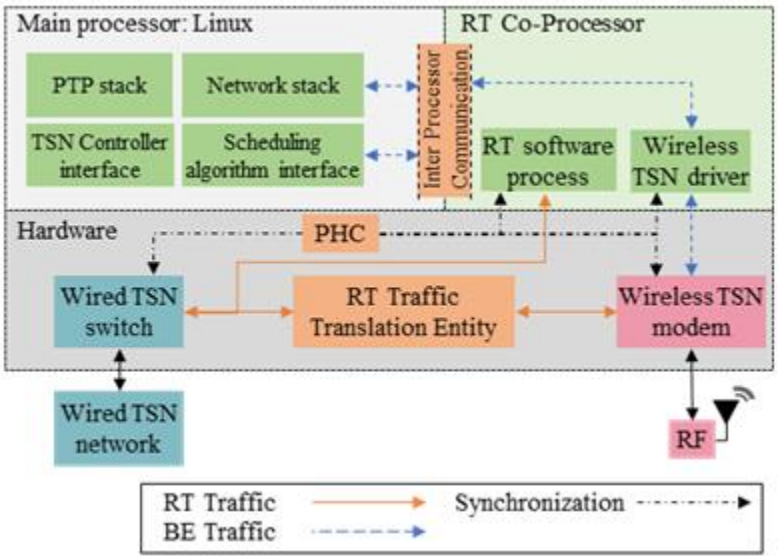

Figure 2. Block diagram of the Wi-Fi TSN node HW/SW architecture

- **Hardware for the Wi-Fi TSN Node (VHDL):** includes the real-time physical communications interfaces (Ethernet TSN and Wi-Fi TSN), a PTP Hardware clock, and a Real Time (RT) traffic translation entity.
- **The real-time co-processor:** hosts the Wi-Fi TSN modem driver in charge of configuring/controlling the Wi-Fi TSN modem and the RT software program in charge of configuring/controlling the RT traffic translation entity and managing the best effort (BE) traffic.
- **The main processor:** runs a Linux operating system and provides: the primary control application, hardware and control access drivers, a TCP/IP network stack, a PTP stack, an interface/API for external TSN controller communication, and an interface/ API for wireless scheduling methods.

3.1 UPDATED REQUIREMENTS

Table 4 sums up the requirements for the Wi-Fi TSN nodes already gathered in deliverable D1.2:

KPI	**Value**
Data Rate	• 64 Mbps (theoretical maximum data rate), • 10 to 16 Mbps (real expected data rate) for a BW of 20 MHz and a MCS 4
Synchronization	• < 100 ns in the Wireless hop
End-to-end latency	• 100-200 μs at the wireless hop (RT traffic) • unbounded but expected to be handled below 20 ms (BE traffic)
Reliability	• < 10^{-3} for low interference conditions

Table 4. Performance requirements for the Wi-Fi TSN nodes

As an update of these requirements, during the design and implementation of the complete TIMING architecture the following new requirements have arisen:

- **PTP profiles:** the PTP stack executing into the main processor must be configurable into several profiles (default PTP profile, power profile, telecom profile, 802.11AS profiles… etc.) Therefore, the PTP stack must support several configurations such as: E2E/P2P, unicast/multicast, L2(ethernet)/L4(UDP).
- **Wireless TSN reconfiguration time for scheduling algorithms:** the time it takes to reconfigure the Wireless TSN superframe, propagate the information to all the devices in the wireless segment and resume normal operation must be reduced as much as possible. In order to get the most out of the scheduling algorithms, the fastest they can apply a new configuration, the better the performance will be. Ideally a new configuration should be applied every 1-2 TSN superframes, but a top performance of 5-10 TSN superframes is expected.
- **Statistics:** the Wi-Fi TSN AP must provide the TSN controller and the scheduling algorithms with the statistics they require. The following statistics are currently foreseen: current TSN configuration, SNR and CIR with every STA, average size of aggregated DL and UL queues, and average number of empty slots in the last N frames in the DL and UL.
- **BE slot configuration:** the configuration of the BE slots in the wireless TSN superframe (position, size, UL/DL… etc.) must be provided by the scheduling algorithms. The Wi-Fi TSN AP must receive from the scheduling algorithm and apply to the devices of the wireless TSN segment the configuration of the BE traffic slots.

3.2 UPDATED SPECIFICATIONS

I order to fulfil the new requirements described above, the following change of specifications of several blocks in the architecture of the Wi-Fi TSN nodes has been necessary:

- **Real-time co-processor:**
  - Wireless TSN driver

        - Statistics collection task (new feature): A new feature has to be added into the wireless TSN driver so that it collects the necessary statistics from the wireless TSN modem and sends them to the main processor (to the statistics collection block)
    - RT software process
        - Wireless TSN configuration propagation task (new feature): A new feature has to be added into the RT software process so that it propagates the wireless TSN superframe configuration into all the devices of the wireless TSN segment via the beacon frame that is sent at the start of each superframe.
- **Main processor:**
    - PTP stack:
        - The PTP stack: the PTP stack and its configuration files must be modified accordingly in order to support several PTP profiles (default PTP profile, power profile, telecom profile, 802.11AS profiles… etc.) and the configurations they require (E2E/P2P, unicast/multicast, L2(ethernet)/L4(UDP).
    - Statistics collection block (new block)
        - A new block has to be added into the main processor which will be in charge of receiving the statistics collected by the wireless TSN driver and sending them via the control API to the TSN controller or to the scheduling algorithms when required.
    - Scheduling algorithms (new block)
        - In order to reduce the reconfiguration time of the wireless TSN superframe as much as possible, the scheduling algorithms designed by the UPC team will be integrated into the main processor of the Wi-Fi TSN AP.
    - Control API (new block)
        - The former TSN controller interface and the scheduling algorithm interface will be replaced by an API in order to ease their use. This API will be used both for consulting statistics and for configuring the Wi-Fi TSN nodes.

3.3 PERFORMANCE EVALUATION TESTS

In order to evaluate the performance requirements depicted above several tests have been carried out.

3.3.1 Superframe configuration and latency test

3.3.1.1 Description

In this test several uplink and downlink flows have been configured in the HW and the cycle time and latencies have been measured. The measurement setup comprises a four-port oscilloscope with 4 GHz BW used to measure the exchanged frames, one AP, and five STAs. The RF output port of the AP is connected to a splitter whose outputs are connected to an antenna and the scope. The AP was connected to the first input of the scope, two STAs were connected to the

second input of the scope through splitters, and the two remaining STAs were connected to the third input of the scope. Table 5 summarizes the configured network and data flows.

RT Downlink Subframes			RT Uplink Frames		
Subframe type	Payload length [B]	MCS	Subframe type	Payload length [B]	MCS
Beacon	14	BPSK ½	-	-	-
DL STA 1	10	QPSK ½	UL STA 1	17	QPSK ½
DL STA 2	16	QPSK ½	UL STA 2	23	QPSK ½
DL STA 3	9	BPSK ½	UL STA 3	11	BPSK ½
DL STA 4	35	16-QAM ½	UL STA 4	10	16-QAM ½
DL STA 5	50	64-QAM ¾	UL STA 5	24	64-QAM ¾

Table 5. Superframe Structure with TRT = 200 µs, TS = 500 µs

3.3.1.2 Results

Figure 3 shows the over-the-air capture of the transmitted/received superframe. It can be shown how the different flows (both RT flows and BE flows) are organized into the superframe.

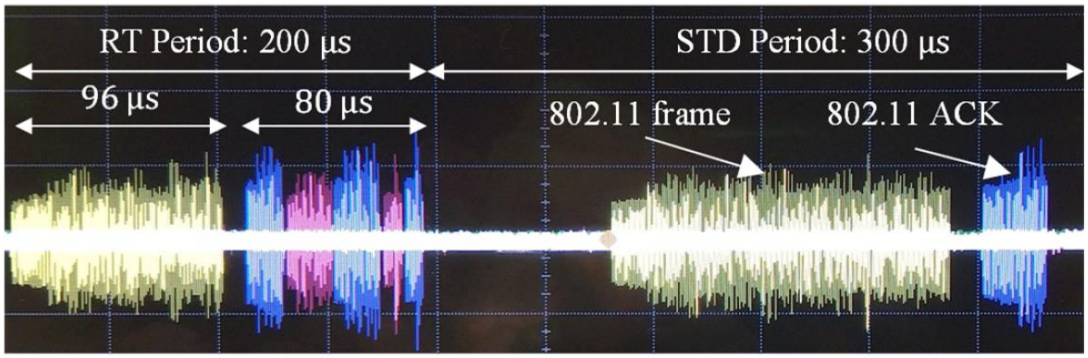

Figure 3. Superframe capture with RT and BE periods, TS = 500 µs

3.3.2 Wireless hop synchronization test

3.3.2.1 Description

In this test the synchronization precision of the PTP stack running between the Wi-Fi TSN AP and the Wi-Fi TSN STAs has been measured. As depicted in Figure 2, the Wi-Fi TSN nodes include a PHC that holds the system time. This PHC is the one being synchronized by PTP. Besides, the output of the PHC is introduced into a PPS gen block that generates a rising edge every time that the PHC output reaches 0 ns, hence it is possible to measure the synchronization precision comparing the PPSs of the different devices.

The experimental testbed (see Figure 4) comprises 4 main elements: two Wi-Fi TSN nodes, an Anite Propsim F8 Radio channel emulator used to test different wireless channels and a Tektronix MSO 2040B oscilloscope used to compare the PPS signals.

For the initial tests a Small Office, Rayleigh wireless channel (WLAN channel A from [VIN04]) has been used.

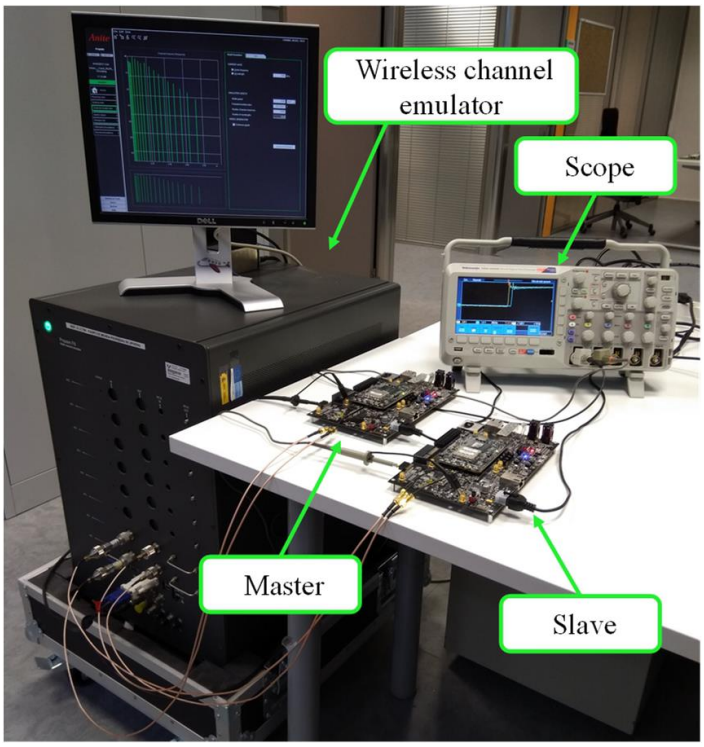

Figure 4. Synchronization precision measurement setup

3.3.2.2 Results

Figure 5 shows the obtained synchronization precision. As can be seen the expected +- 100ns precision is obtained.

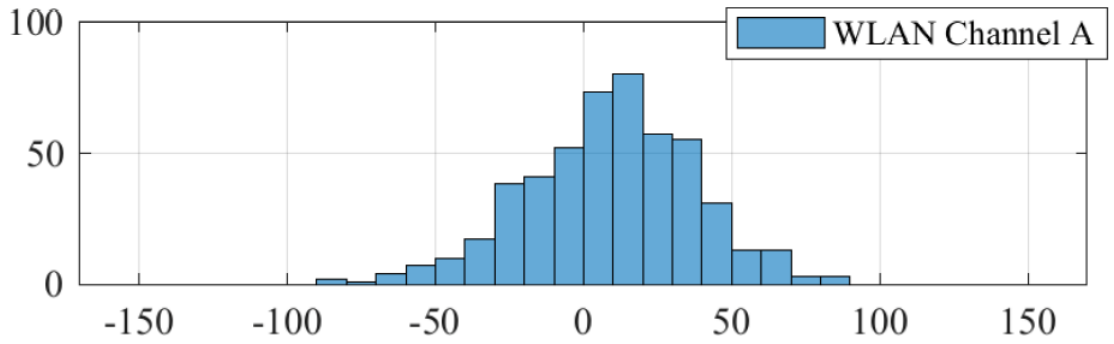

Figure 5. Synchronization precision results

3.3.3 PER test

3.3.3.1 Description

The performance of the system for DL and UL periods has been assessed in terms of raw PER (without retransmissions), and PER with 1 retransmission. The retransmissions are performed with a lower MCS. In the case of 64-QAM ¾, the retransmission uses 16-QAM ½, in the case of 16-QAM ½, the retransmission uses QPSK ½, and in the case of QPSK ½, the retransmission uses BPSK ½.

3.3.3.2 Results

The results obtained show that high reliability (PER < 10^{-7}) is feasible without and with retransmissions under the condition of high Rx power. PER < 10^{-7} is especially feasible with low-order modulations (BPSK and QPSK) since these modulations offer higher protection against noise. Regarding the PER results with retransmission, the retransmission scheme seems to obtain little improvements with BPSK 1/2 and QPSK 1/2 (1-2 dB), unlike 16 QAM with retransmissions, which has a gain of 5 - 6 dB compared to 16 QAM without retransmissions.

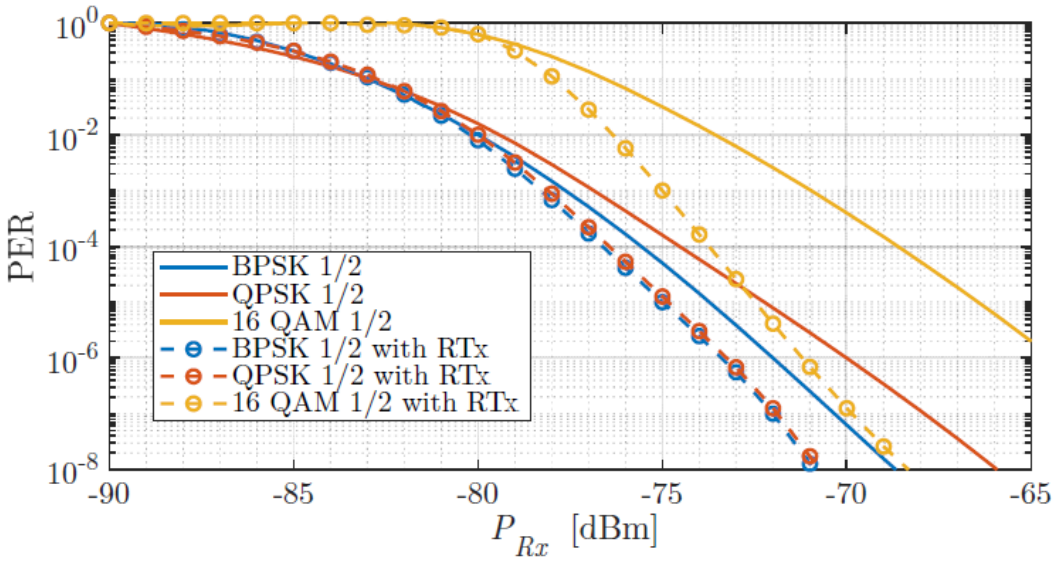

Figure 6. Packet error rate results

4 ETHERNET TSN NODES

This component is responsible for providing the main operational settings to Safran's Ethernet-based segment of the TSN network for TIMING, as well as for gathering the main telemetry and system status indicators that will be fed to third-party tools to calculate optimized operational settings for our network segment. The Central Network Configuration (CNC) component is an adaptation of a proprietary solution of Safran that has been tailored to meet the main requirements set forth in TIMING and is meant to operate as a centralized point of configuration and monitoring for our segment of the demonstrator network for TIMING.

In the sections below, we are going to explain the structure and components that compose the CNC system, the current status of these components, and some performance evaluation tests carried out.

4.1 STRUCTURE AND COMPONENTS

This section explains and breaks down all the components constituting the CNC system, aiming to deliver an in-depth understanding of each element's role and significance within the framework of CNC machinery. In Figure 7, there are numbers in parentheses representing each component that will be utilized as a reference throughout the explanation.

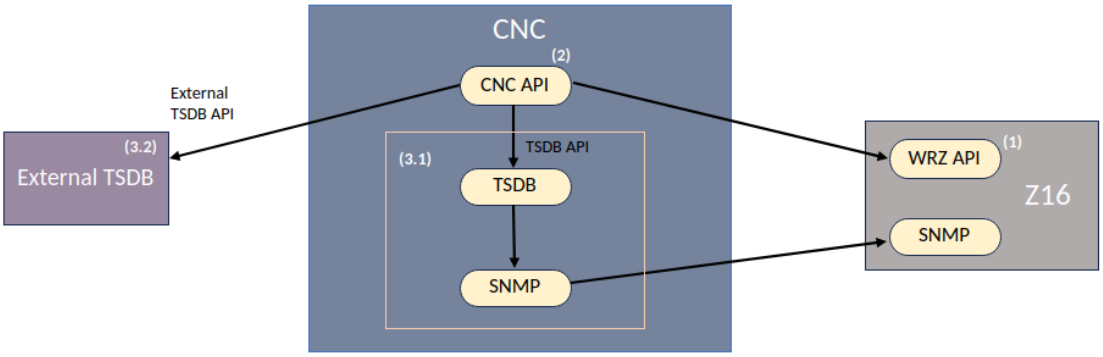

Figure 7. Ethernet TSN Nodes Structure

4.1.1 The Southbound API between the CNC and the TSN switches

The WRZ-API stands as a specialized interface facilitating communication between the Northbound API and Safran's TSN switches. This API, integrated within the equipment, plays a pivotal role in transmitting network configurations from the CNC to the respective equipment.

The network configurations are initially sent from the TSN Controller to the CNC. Subsequently, the CNC relays the configuration requests to the respective Z16 node in accordance with the received instructions for network setup.

We will now delve into the endpoints of the WRZ API that we have developed for our segment of the demonstrator network for TIMING. There exist two distinct types of endpoints: those associated with configuring the TSN network and those involved in generating reports.

4.1.1.1 Configuration endpoints

These endpoints are related to the configuration of the TSN network and are categorized into four types: VLAN configuration, TAS configuration, CAM configuration, and QCI configuration. Each of these types encompasses an endpoint through which new configurations can be applied, as detailed in Table 6.

VLAN	POST	/v1/tsn/vlan
TAS	POST	/v1/tsn/tas
CAM	POST	/v1/tsn/cam
QCI	POST	/v1/tsn/qci

Table 6. WRZ API Configuration endpoints

4.1.1.2 Generating reports endpoints

These endpoints aim to generate data about packet traffic within the Z16. The data extracted has the structure defined in Figure 8 (EPOCH time, timestamps in nanoseconds and packet sequence number), it is stored in the device itself, and can be downloaded. This data is going to be used later, with the Northbound API, to calculate latency and packets lost reports. Z16 devices are equipped with a latency probe enabling this functionality. The endpoints developed are in Table 7.

Reports	GET	/v1/tsn/reports	Get name and path of all reports generated
	POST		Generate a new report given TSN network parameters, report name and duration of the report
	GET	/v1/tsn/reports/{name}	Download a report previously generated, given the name

Table 7. WRZ API Reports endpoints

```
-----------New Capture Session with VID(1) PCP(0) Port (5)-------------
TS UTC (s),TS (ns),Sequence Number
1699870220,800349216,15182
1699870220,900474144,15183
1699870221,611904,15184
1699870221,100729536,15185
1699870221,200892000,15186
1699870368,820972992,16660
1699870368,921107232,16661
1699870369,21244032,16662
```

Figure 8. Reports structure example

4.1.2 The Northbound API between the CNC and the TSN Controller

The Northbound API between the CNC and the TSN Controller implements a generic REST API for receiving the optimized operational settings calculated at UPC's TSN Controller.

The endpoints developed for the CNC can be divided into several types: those for network equipment management, network configuration, latency and packet loss test generation, and network topology visualization.

4.1.2.1 Network equipment management

These are the endpoints responsible for managing the settings of the Z16 nodes stored within the CNC's database. Through them, we can add, modify, delete and list devices. Once the devices are added, we can monitor them to check their status (whether they are active or not), if they are scraping data, and their synchronization mode. Through other endpoints of the CNC API, we will also be able to send requests to these devices to configure the network, view the network topology, and generate reports. The endpoint paths can be seen in Table 8.

Devices	GET	/v1/device	Get all the devices in the database
	POST		Add a new device
	PATCH		Modify a device, given the name
	DELETE		Delete a device, given the name

Table 8. CNC API Devices endpoints

4.1.2.2 Network configuration

These endpoints are used to configure the TSN network. The Z16 devices allow four types of modules to configure the network: VLAN, TAS, CAM, and QCI. For each of them, we will have endpoints to create rules, and other endpoints to apply those rules we have created. Rules are how we refer to specific TSN configurations.

The workflow would then be as follows: using the endpoints */v1/net-config/<configuration module>/rule*, we can create, list, modify, and delete the rules we need. Afterwards, with the endpoints */v1/net-config/<configuration module>*, we will select the rules we want to apply and the devices to which we want to apply them.

The complete list of endpoints can be found in Table 9.

Add rules	VLAN	GET	/v1/net-config/vlan/rule
		POST	
		PATCH	
		DELETE	
	TAS	GET	/v1/net-config/tas/rule
		POST	
		PATCH	
		DELETE	
	CAM	GET	/v1/net-config/cam/rule
		POST	
		PATCH	
		DELETE	
	QCI	GET	/v1/net-config/qci/rule
		POST	
		PATCH	
		DELETE	
Apply configuration	VLAN	GET	/v1/net-config/vlan
		POST	
		DELETE	
	TAS	GET	/v1/net-config/tas
		POST	
		DELETE	
	CAM	GET	/v1/net-config/cam
		POST	
		DELETE	
	QCI	GET	/v1/net-config/cam
		POST	
		DELETE	

Table 9. CNC API Configuration endpoints

4.1.2.3 Latency and packet loss test generation

These endpoints are used to generate reports on packet traffic in the TSN network. Data is generated by a userland application that runs on the Z16 and, during the specified time, gathers results provided by the latency probe (EPOCH time, packet timestamps, and packet identifiers) and saves them locally as CSV files. After the specified duration, the CNC requests the generated CSV files from the devices (with the structure shown in Figure 8), and using these CSV files, it calculates the total network latency and the lost packets. The endpoint paths can be seen in Table 10.

Reports	GET	/v1/net-analyzer	Get all the reports generated
	POST		Start a new report
	DELETE		Delete a report, given the name
	GET	/v1/net-analyzer/downloader	Download a zip with all the files generated in a report, given the report name
	GET	/v1/net-analyzer/latency	Returns a csv with the latency calculated, given the report name
	GET	/v1/net-analyzer/packets-lost	Returns a csv with the packets-lost calculated, given the report name

Table 10. CNC API Reports endpoints

4.1.2.4 Network topology visualization

This endpoint returns a dictionary containing information about the connections of the devices. For each device, it provides details about which interfaces are connected to other devices and specifies the name, interface, and address of the connected device. With this information, the network's topology can be reconstructed.

Topology	GET	/v1/net-analyzer/topology	Returns a dictionary with informacion about the network topology

Table 11. CNC API Topology endpoints

4.1.2.5 Export Metrics Settings

This endpoint allows for obtaining a list of monitored devices, as well as adding or removing devices. Additionally, it enables the configuration of credentials required for exporting data to an external time-series database.

Export Metrics Settings	POST	/v1/export-metrics-settings	Configure external TSDB credentials and enable/disable export metrics
	GET	/v1/export-metrics-settings/targets	Get targets which are being monitored
	POST		Modify which targets are being monitored
	POST	/v1/export-metrics-settings/reload-config	Reload configuration (must be done to apply changes)

Table 12. Export Metrics Settings endpoints

4.1.3 The CNC Monitoring component

The CNC Monitoring component implements the main elements responsible for aggregating and storing the main monitoring sources of all the nodes of Safran's Ethernet-based TSN solution.

Similar to the structural framework of monitoring architectures, CNC Monitoring is divided into three components, as shown in Figure 9. Firstly, the data collection phase facilitated by SNMP Exporter, enabling the extraction of desired metrics from the devices. Secondly, the employment of a dedicated time-series database (TSDB), exemplified by VictoriaMetrics in this context, serves as a repository for storing the data acquired via SNMP Exporter. Lastly, the visualization platform, which encompasses options such as VictoriaMetrics, InfluxDB, or alternative platforms as per the user's preference.

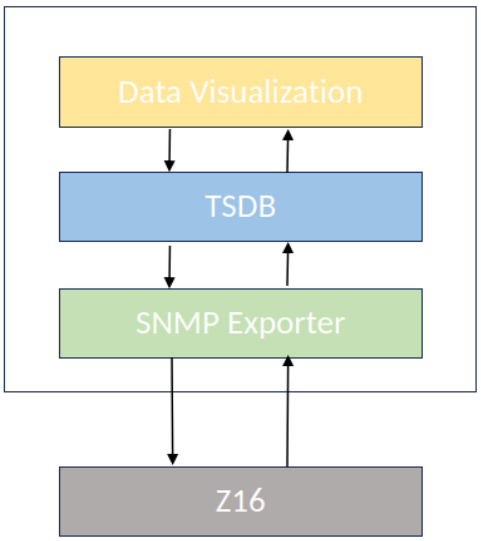

Figure 9. CNC Monitoring component structure

4.1.3.1 Data Collection and TSDB

To collect data from the devices, we use the SNMP Exporter program. SNMP sends requests to the monitored devices for metrics that we have previously defined, and the devices return the values of these metrics. Once we retrieve this data, it is stored in a time series database (TSDB), in this case, VictoriaMetrics. From there, we can visualize the data using various programs, such as Grafana, or VictoriaMetrics' own interface.

The configuration of which devices are being monitored can be done through the CNC API, as explained in the section above.

4.1.3.2 External TSDB

We can also export the data stored in VictoriaMetrics to another time series database, in this case InfluxDB. To do this we must configure CNC with our InfluxDB credentials, and select that the devices export the data they collect. This can be done though the CNC API. In this way we can survey the status of the Ethernet-based TSN segment of the network from external components, such as the system optimization modules supplied by UPC.

4.2 CURRENT STATUS

Some of the main functionalities of Safran's CNC have already been adapted to the expected use case for TIMING. Nonetheless, not all functionalities are yet available at this stage, and their corresponding status is summarized in the points below.

Component	Subcomponent	Status
The Southbound API between the CNC and the TSN switches	Configuration	Done
	Reports	Done
The Northbound API between the CNC and the TSN Controller	Network equipment management	Done
	Network configuration	In Progress
	Latency and packet loss test generation	Done
	Network topology visualization	Done
	Export Metrics Settings	In Progress
The CNC Monitoring component	Data Collection	Done
	External TSDB	Done

Table 13. Current status

4.3 TESTING

4.3.1 Test 003: SafranCNC-Monitoring_DeliverStats_1

As explained in D1.3, this test verifies the operation of the CNC monitoring component by ensuring that all the relevant TSN-related statistics and telemetry data from the network are fetched from Safran's TSN switches.

In order to test it, we monitored some devices in a TSN network, and ensured that the metric 'cpu_usage' was fetched from Safran's TSN switches. The metric values are going to be visualized with VictoriaMetrics interface, as we can see in Figure 10.

Figure 10. Metric 'cpu_usage' being monitored

4.3.2 Test 004: SafranCNC-SBI_configureVLAN_1

As explained in D1.3, this test verifies the operation of the WRZ-API implementing the SBI component of Safran's CNC by issuing VLAN configuration requests against one of Safran's TSN switches.

The configuration to apply is:

- WR0: VID: 1; PCP: 2; MAC_DST: 0xcafebabe0102; MAC_ADDR: 0xaabbccddeeff; HAS_DEST:1; DEST:1.
- WR1: VID: 1; PCP: 2; MAC_DST: 0xcafebabe0102; MAC_ADDR: 0xaabbccddeeff.

In Figure 11 we can see the request being done to the WRZ API to apply this configuration and the successful response.

Curl

```
curl -X 'POST' \
  'http://10.22.18.9:8201/v1/tsn/vlan?config_id=55' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "wr0": [
    {
      "addr": "0xcafebabe0102;",
      "vlan_id": 1,
      "vlan_prio": 2,
      "filter_en": 0,
      "mac_addr": "0xaabbccddeeff",
      "proto": 0,
      "ip": 0,
      "vlan_id_cfg": 0,
      "port": 0,
      "vlan_prio_cfg": 0,
      "dscp": 0,
      "dest": 1,
      "has_dest": 1,
      "ds": 0,
      "redundant": 0,
      "red_dest": 0,
      "red_handle": 0
    }
  ],
  "wr1": [
    {
      "addr": "0xcafebabe0102;",
      "vlan_id": 1,
      "vlan_prio": 2,
      "filter_en": 0,
      "mac_addr": "0xaabbccddeeff",
      "proto": 0,
      "ip": 0,
      "vlan_id_cfg": 0,
      "port": 0,
      "vlan_prio_cfg": 0,
      "dscp": 0,
      "dest": 0,
      "has_dest": 0,
      "ds": 0,
      "redundant": 0,
      "red_dest": 0,
      "red_handle": 0
    }
  ]
}'
```

Request URL

```
http://10.22.18.9:8201/v1/tsn/vlan?config_id=55
```

Server response

Code	Details
200	Response body

```
{
  "msg": "Success",
  "data": []
}
```

Figure 11. Apply VLAN configuration through WRZ API

4.3.3 Test 005: SafranCNC-SBI_configureTAS_1

As explained in D1.3, this test verifies the operation of the WRZ-API implementing the SBI component of Safran's CNC by issuing TAS configuration requests against one of Safran's TSN switches. The configuration to apply is:

- Nº entries in the GCL: 3
- BaseTime: <Current Network Epoch> + Additional offset to make up for configuration time
- GCL Specification:
    1. Slot #0: Duration 1ms, Gate Settings: 0x9 (Q3 & Q0 open)
    2. Slot #1: Duration 2ms, Gate Settings: 0xa (Q3 & Q1 open)
    3. Slot #2: Duration 1ms, Gate Settings: 0X0 (All gates closed)

In Figure 12 we can see the request being done to the WRZ API to apply this configuration and the successful response.

Curl

```
curl -X 'POST' \
  'http://10.22.18.9:8201/v1/tsn/tas?config_id=55' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "wr0": {
    "rules": [
      {
        "interval_time": 62500,
        "gate_cfg": 9
      },
      {
        "interval_time": 125000,
        "gate_cfg": 10
      },
      {
        "interval_time": 62500,
        "gate_cfg": 0
      }
    ],
    "tv_sec": 1703238797,
    "tv_nsec": 0,
    "tick_granularity": 0,
    "framePreemptionStatusTable": 0
  }
}'
```

Request URL

```
http://10.22.18.9:8201/v1/tsn/tas?config_id=55
```

Server response

Code	Details
200	Response body

```
{
  "msg": "Success",
  "data": []
}
```

Response headers

Figure 12. Apply TASconfiguration through WRZ API

4.3.4 Test 006: SafranCNC-NBI_Latency_1

Test #006	
Test Id	SafranCNC-NBI_Latency_1
Description	
This test verifies the operation of the CNC-API implementing the NBI component of Safran's CNC by generating a latency report.	
Preconditions	
The test assumes the TSN network has been already configured and there is packet traffic between two Z16 nodes.	
Step/s of the operation/functionality under test 1. Generate a report with arguments: a. VLAN Id: 1 b. VLAN Priority: 0 c. Sequence Offset: 22 d. Mask: ffff e. Sequence Length: 16 f. Duration: 2 min g. Name: 22-dic h. Devices: i. Device 1: 1. Host: 10.22.18.9 2. Iface: 1 3. Frer: 0 ii. Device 2: 1. Host: 10.22.18.10 2. Iface: 1 3. Frer: 0 2. If the request was correctly formatted, a new report should be created with Status In Progress. After the report duration time has passed we can obtain the latency report calculated.	
Expected Results	
The execution of the test should verify that latency reports can be generated using CNC-API.	
Output	
A successful operation of the test should be noted by returning an HTTP 200 Success execution code over the CNC-API.	

In Figure 13 we can see the request to start a new report made through the CNC API.

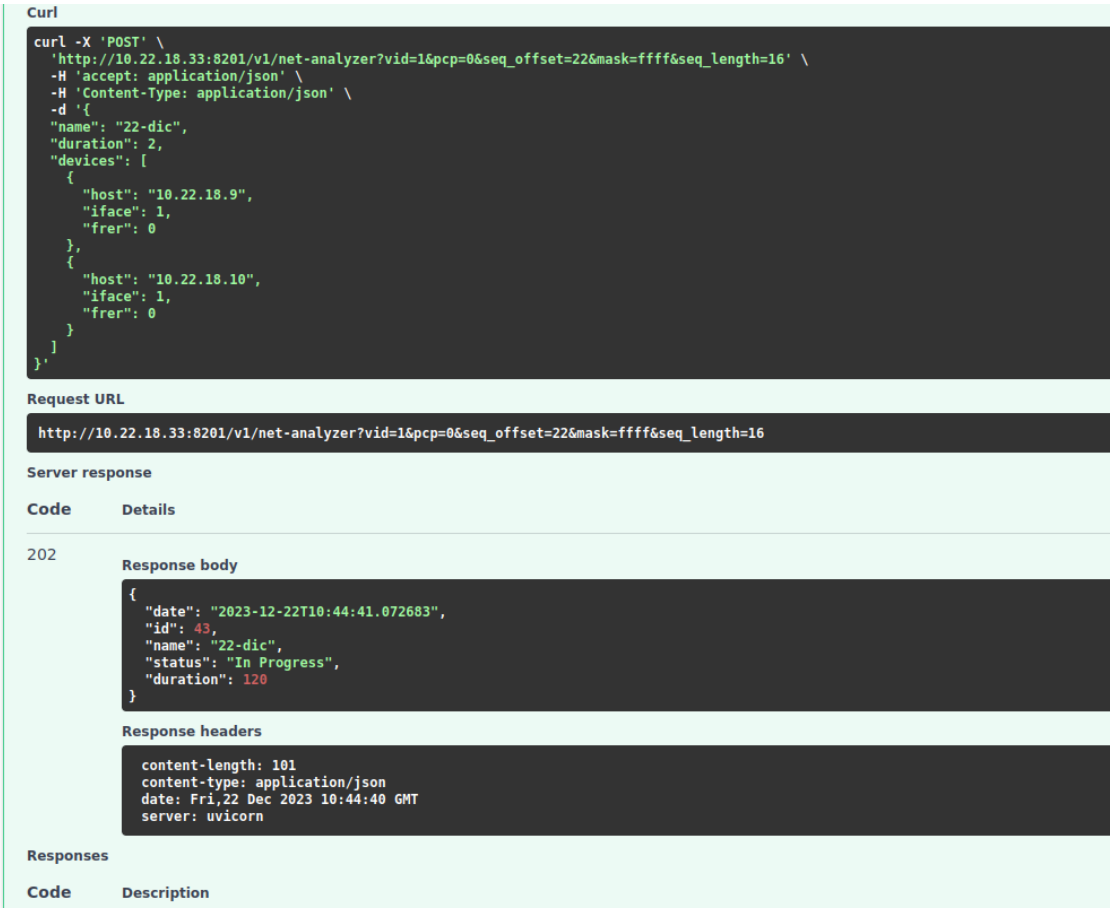

Figure 13. Start a new report through CNC API

In Figure 14 we can see the request to get the latency report calculated with Successful code 200.

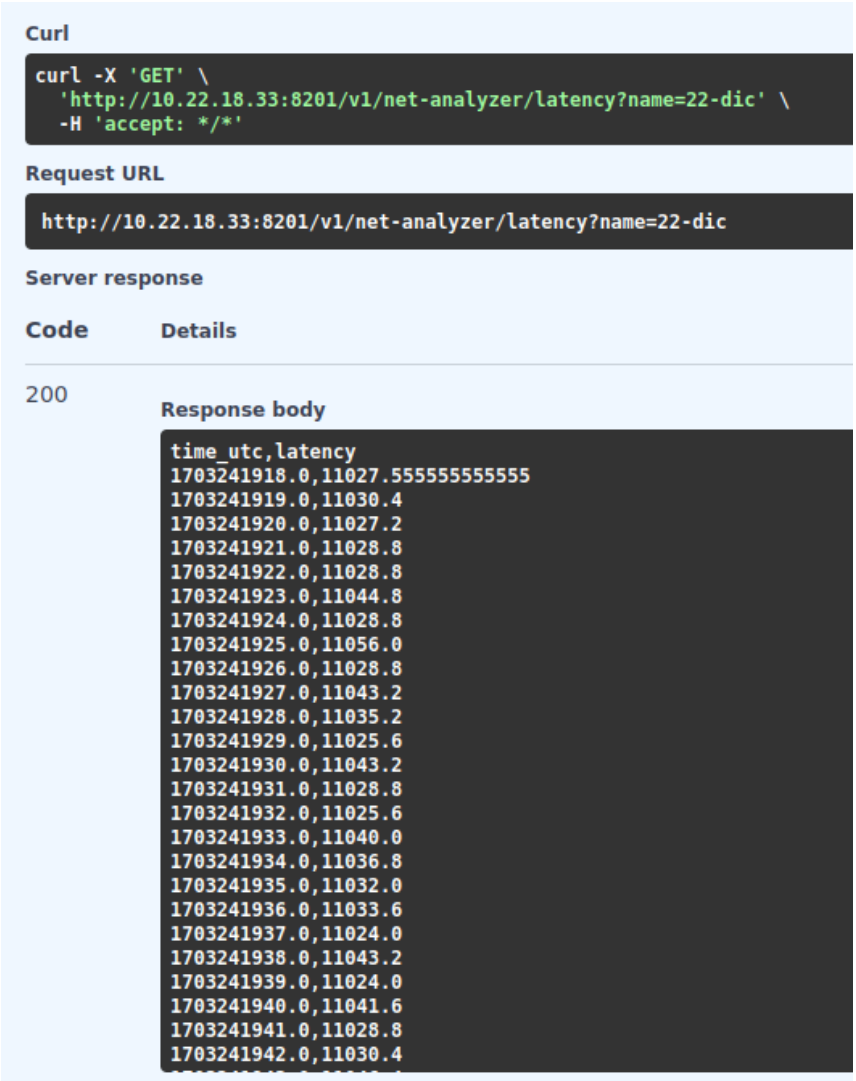

Figure 14. Get latency report file calculated through CNC-API

5 TSN CONTROLLER

The Software Defined Networking (SDN) - Time-Sensitive Networking (TSN) controller serves as the intermediary between the upper-level entities, such as the Connectivity Manager (CM), and the data plane elements, specifically the wired TSN-enabled switches and wireless Access Points (APs), for the configuration of data flows according to the specifications set by the end user/applications requirements. In addition, it also is a crucial element to enable the provisioning decisions taken at the CM, since it exposes the abstract topology of the data plane as well as the high-level capabilities of the multiple network nodes.

Previous deliverable D1.2 reported the general architecture of the SDN-TSN controller as well as the main models employed for the representation of the abstract topology and devices capabilities. The latest developments have focused on two main fronts: i) extension of the NBI for communicating with the CM, and SBI for communicating with the wired CNC and wireless APs for configuration purposes; ii) extensions to the internal logic and structure for handling service provisioning requests. The developments of the extended interfaces will be reported in

D1.2 of SP2, as well as the validation tests performed, due to the focus on the integration aspects of the implementation. In below, the current deliverable focuses on the developments and tests of the internal structure of the SDN-TSN controller.

5.1 UPDATES OF THE ARCHITECTURE

A provisioning request, in essence, contains the details of the endpoints to be interconnected and the set of physical nodes and corresponding termination points that need to be configured to materialize the flow, as well as a set of configuration parameters according to the flow requirements. Upon reception of a configuration request, the SDN-TSN controller needs to identify the nodes and termination points stated at the request and configure them in a specific way depending on their type (wired or wireless). The new developments and extensions are focused on this part of the functionalities, following the full vertical layered structure reported in D1.2. The parts that have been extended or modified are:

- Provisioning manager/service: it is the module within the SDN-TSN controller responsible for exposing the endpoints that enable the CRUD (Create, Read, Update, Delete) operations of flows within the TSN-enabled wired and wireless technological domains. The current version has been focused on developing all the logic for receiving and treating flow creation requests. Upon reception of the request, the provisioning service engages with the topology manager/service for retrieving the object that contains the details of the abstracted view of the capabilities for all of the nodes present in the path specified at the request. Within the capabilities, a specific field indicates the type of the node (e.g., wired switch, wireless AP). Thanks to this information, the provisioning manager is able to distinguish how the particular node and termination point should be configured and forward the configuration details to the appropriate module within the SDN-TSN controller responsible for the communication with the data plane.
- Southbound interface (SBI): one of the major modifications done in the current version is how the SBI is structured. In order to facilitate the coexistence of multiple data planes as well as configuration protocols, the concept of adapter has been defined. An adapter is a module that acts as a wrapper for the configuration of the underlying data planes. It offers an abstracted technology-agnostic view of the configuration endpoints of the data plane hardware to the provisioning service, or other modules within the SDN-TSN controller. On the lower part of each adapter, it implements the specific configuration protocol and operations that are dependent of the hardware that needs to be configured. Such re-structuring allows for a more flexible way of communicating with the remote hardware, facilitating the extensions of new operations or communication protocols. The current implementation of the SDN-TSN controller implements two adapters, one for the wired domain and another for the wireless one, implementing at their turn in the technology specific part of the adapter a REST and an SSH client, respectively, for communicating with the wired CNC or modify remotely the configuration files of the APs.

As a summary, Figure 15 depicts a simplified workflow of the provisioning operation for a flow establishment, highlighting the extended modules and their role on the whole sequence. The workflow focuses on the operations at the SDN-TSN controller level. First, the request for flow

establishment arrives at the NBI of the controller and is forwarded to the Provisioning Service (Step 1). This module then parses the contents of the body of the request which, among other details, contain the information about the physical nodes and termination points that need to be configured (Step 2). In order to understand the type of node, and which of the several adapters needs to interact with for its configuration, the Provisioning Service retrieves the details of the topology and node characteristics from the Topology Service (Step 3). Once this is done, the Provisioning Service engages with either the wired (step 4.a) or wireless (step 4.b) adapter. Once the configuration request is forwarded, the adapters treat the configuration details, construct the appropriate configuration message and employ the suitable communication protocol to remotely configure the specified nodes in the provisioning request (Steps 5.a and 5.b). This last part is done at the technology dependent part of the adapters. Once all the configurations are successfully applied, the Provisioning Service stores the details of the configuration plus a flow identifier at the Inventory via the Inventory Service (Step 6). This last step is necessary to have a relationship of the applied configurations and their associated flows, in cases they need to be modified or deleted.

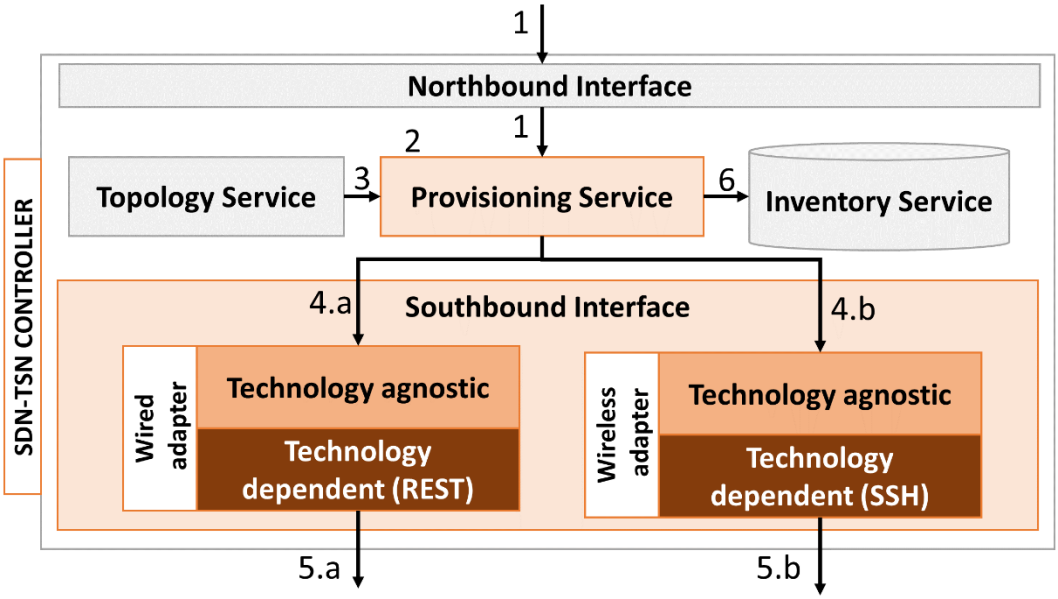

Figure 15. SDN-TSN controller updated architecture and flow provisioning workflow.

5.2 Test and Evaluation

In order to evaluate the correct performance of the extensions, a series of test have been executed. Particularly, we tested that the Provisioning Service is able to properly parse the flow establishment requests and then engages with either the wired or wireless adapter depending on the type of switch to be configured, as well as the low-level operations performed at the adapters in order to send the configuration commands. The tests focused on the internal operation of the SDN-TSN controller, hence, we will be not reporting the messages exchanges nor the validation of the NBI and SBI.

First, we report the topology assumed for the test. Figure 16 illustrates a schematic of it. Given this topology, the Topology Service exposes a JSON representation of all the data structures and properties of both abstracted nodes and links, which is the information that is employed by the

Provisioning Service to understand which of the adapters should be contacted for the node configuration.

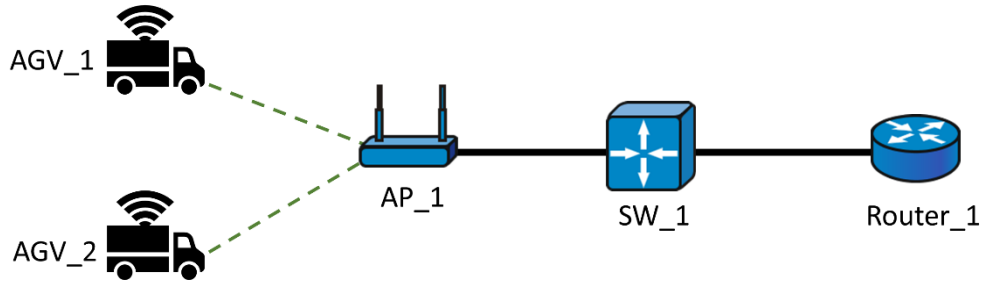

Figure 16. Test topology.

Given this topology, the test consisted in sending through an external REST client a flow configuration request to the NBI of the SDN-TSN controller, in order to start the configuration process. Said request would be the same as received from the CM during the provisioning of flows answering to application requirements. Figure 17 top depicts the part of the body of the request that contains the path that needs to be configured, which is the relevant part for this evaluation; the rest of the details and the evaluation of the interfaces per se will be reported in other deliverables. Upon reception of the request, the Provisioning Service correctly parses the path and identifies the type of the nodes, to then further engage with the adapters. Figure Z bottom showcases an extract of the SDN-TSN controller log, in which it can be seen how the multiple adapters perform the necessary operations to configure the different elements at the nodes. For instance, for the wireless AP, it is necessary to configure both the beacon flow as well as the data flow, in that case, per port. As for the wired switch, it is necessary to create both VLAN and TAS configuration rules per port, which then are finally applied to materialize the desired configuration. As said, the test focuses solely on the internal logic of the controller, so, for the particular execution, the adapters do not engage with the remote endpoints available for the configuration of the physical hardware. Nevertheless, this showcases how the controller is able to correctly handle provisioning request and split the path to be configured according to the types of the nodes expressed in it. Finally, one important aspect relates to the time required for the configuration of the equipment once a provisioning request is received. This time has three major components, namely: i) the NBI communication delay; ii) the SD-TSN controller processing of the request; iii) the SBI communication delay. Out of the three, the sole component that depends exclusively on the implementation of the controller is the processing time, since the communication delays depend strongly on the location of the remote equipment as well as the status and configuration of the management/control network between. Thus, a measure of performance for the SDN-TSN controller is the time that it requires to process the request, split it properly and prepare the configuration messages to be sent through the adapters. We executed several repetitions of the provisioning operation so as to extract the average processing time, which results in around 1-5 ms.

Figure 17. Path details contained in the provisioning request (top); configuration of wired and wireless nodes at the SDN-TSN controller (bottom).

6 TSN CONNECTIVITY MANAGER

The TSN Connectivity Manager (CM) serves as a pivotal component in the TIMING architecture, positioned at the apex and interfacing with both TSN and Metro SDN Controllers as well as the Digital Twin (DT). Figure 18 depicts the latest version of its architecture. Thus, the TSN Connectivity Manger is composed by two main functional blocks, ENP, which is an extension of the capabilities provided by the E-Lighthouse Network Planner (ENP), a renowned optimization and planning software, and the core module. The TSN CM's core function is to coordinate the actions targeted to orchestrate end-to-end (e2e) TSN flows across networks based on data collected from these controllers.

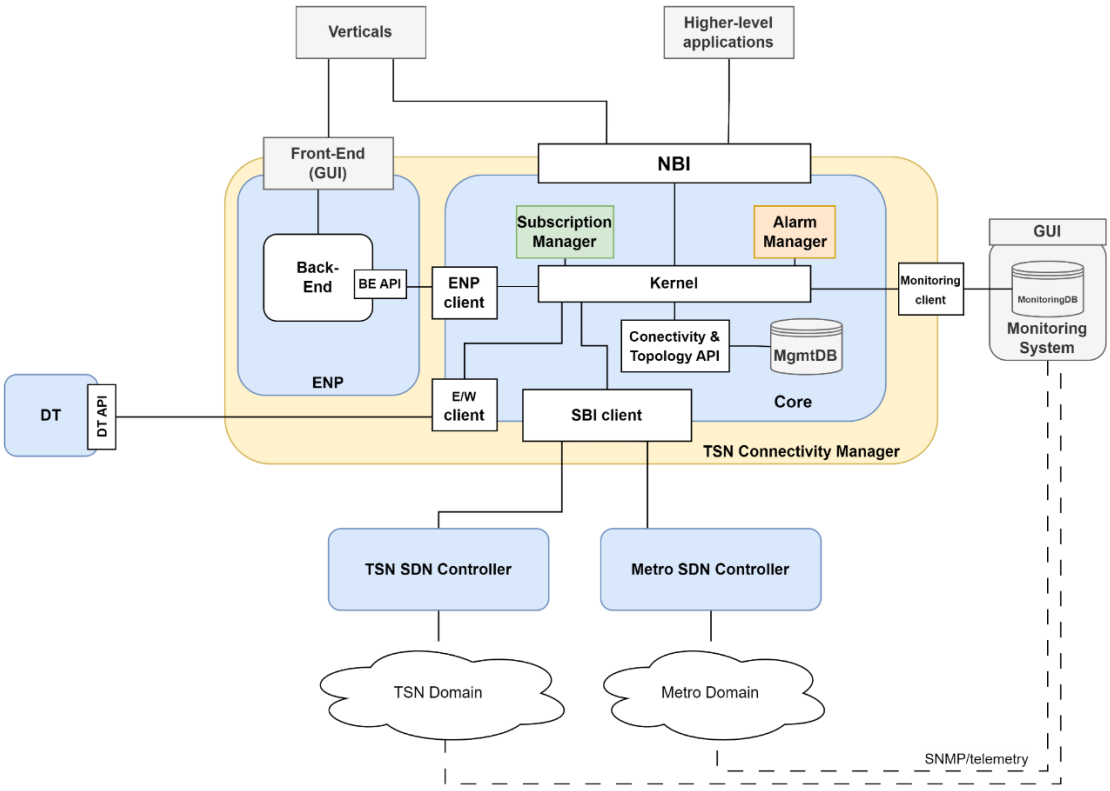

Figure 18. TSN Connectivity Manager Architecture

The evolution related to the TSN Connectivity Manager compared with the presented in deliverable D1.2 can be organized in three main pillars: i) update in the architecture, ii) restructuration of the CM NBI; and iii) enhancements in the GUI.

6.1 UPDATES OF THE ARCHITECTURE

The new architecture presented in Figure 18, solves some gaps from the original version of the TSN CM's architecture, like the absence of ENP module and its interface with the core module, and some rearrangement of the certain submodules in the core one. Thus, under this architectural scope, the functional decomposition of the two main CM's modules is summarized as follows:

- **E-lighthouse Network Planner**: is a commercial network planner tool with an advanced multi-layer and multi-domain network model focused on providing optimized network configuration to the controllers in service provisioning operations [ENP]. It is composed by two main submodules:
    - *Front-End (GUI)*: This interface allows users to interact with the TSN Connectivity Manager, providing commands and receiving visual feedback. This will be the user/vertical entry point in latter TIMING's demonstrations.
    - *Back-End*: It interprets the commands from the GUI and translates them into actions that can be executed within the network. This is where the intelligence of the tool is implemented. Exports an API to enable the core component to manage it.

- **Core:** is the central hub for processing and forwarding operations within the TSN Connectivity Manager. It interprets data, manages workflows, and ensures that network operations align with the predefined policies and quality of service requirements. Moreover, it is structured as follows:
    - *Kernel*: The central part of the TSN Connectivity Manager that handles the main processing tasks
    - *Connectivity & Topology API*: manage the topologies and active flows imported from the controllers to be stored in the MgmtDB.
    - *MgmtDB*: its role is to store, organize and serve the topological data from the different domains and other management information of the CM.
    - *Subscription Manager*: Manages network subscription requests, ensuring that network resources are allocated according to the needs of applications.
    - *Alarm Manager*: Monitors the network for any issues and alerts the network administrators to potential problems.

Concerning the connectivity, TSN CM directly connecting to all TSN and Metro SDN Controllers and the DT. It utilizes three main Application Programming Interfaces (APIs): the Southbound Interface (SBI), Northbound Interface (NBI), and the East/West Interface (E/WBI).

Through the SBI interface, the TSN CM can read underlying topological information, manage provisioning, and suggest configurations to be applied to the actual TSN and Metro networks based on optimization results.

The TSN CM's NBI is crucial for exposing all topology information and active TSN e2e flows to higher-level or external applications, allowing for graphical representation through the Graphical User Interface (GUI).

Also, the East/West interface (E/WBI) in the TIMING architecture serves a function for the operation of the TSN Connectivity Manager and its integration with the Digital Twin. The E/WBI enables the TSN CM to interconnect with the DT to simulate actual network conditions (KPI estimation) in a digital environment.

The Monitoring database (MonitoringDB) is another essential element, storing time series information related to the monitoring of the network devices, to be used for KPI validation purposes in latter stages of the project.

To sum up, compared to the architecture presented in D1.2, the more evident update is the presence of the ENP module, with its front-end providing more intuitive and user-friendly experience, with visual elements that allow network operators to interact with the TSN CM module effectively. Also, the ENP back-end complements the core module in its orchestration and control capabilities, now able to handle more complex operations and integrations within the TIMING architecture. It provides a more dynamic and flexible approach to managing TSN e2e flows based on the deployed ones. Moreover, it is remarkable to highlight the CM's clients (SBI, E/WBI and monitoring) are implemented in the core component, facilitating the operations of the core's kernel. Finally, a new internal interface is defined to enable the communication between the core and ENP.

6.2 UPDATES OF THE TSM CM NBI

The Northbound Interface (NBI) of the CM plays a crucial role in this orchestration, acting as a conduit for communication between the CM and other modules or higher-level applications or verticals. With the advent of the new NBI, there is an evolution in the way network operations are managed, providing a more flexible and dynamic interface.

The updated NBI structure introduces an enhanced modular approach that segregates different functionalities into distinct endpoints. This structure is presented in Figure 19 and consists of three main categories: controllers, flows, and topologies, each serving a unique purpose in the network management ecosystem.

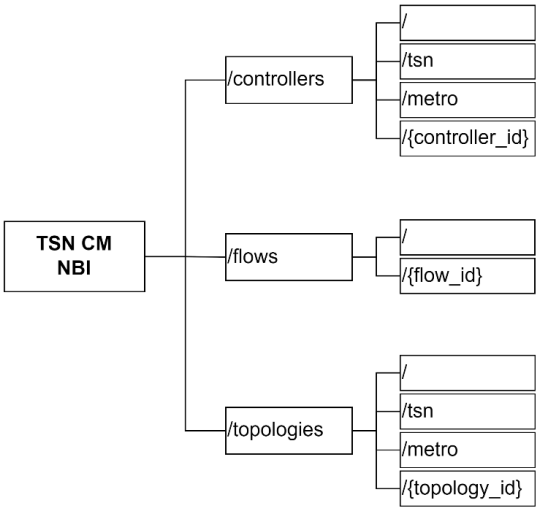

Figure 19. TSN Connectivity Manager NBI endpoint's structure

Furthermore, in Figure 20 depicted the implementation of the NBI, delineates the functionalities and operations that can be performed on various network components like flows, controllers, and topologies.

In this NBI implementation, the clear division of functionalities into flows, controllers, and topologies indicates a structured approach to network management, with RESTful operations (GET, POST, DELETE) enabling standard CRUD (Create, Read, Update, Delete) interactions with the network components. This structure supports scalability and flexibility in network operations, allowing for easy integration with different network management tools and systems.

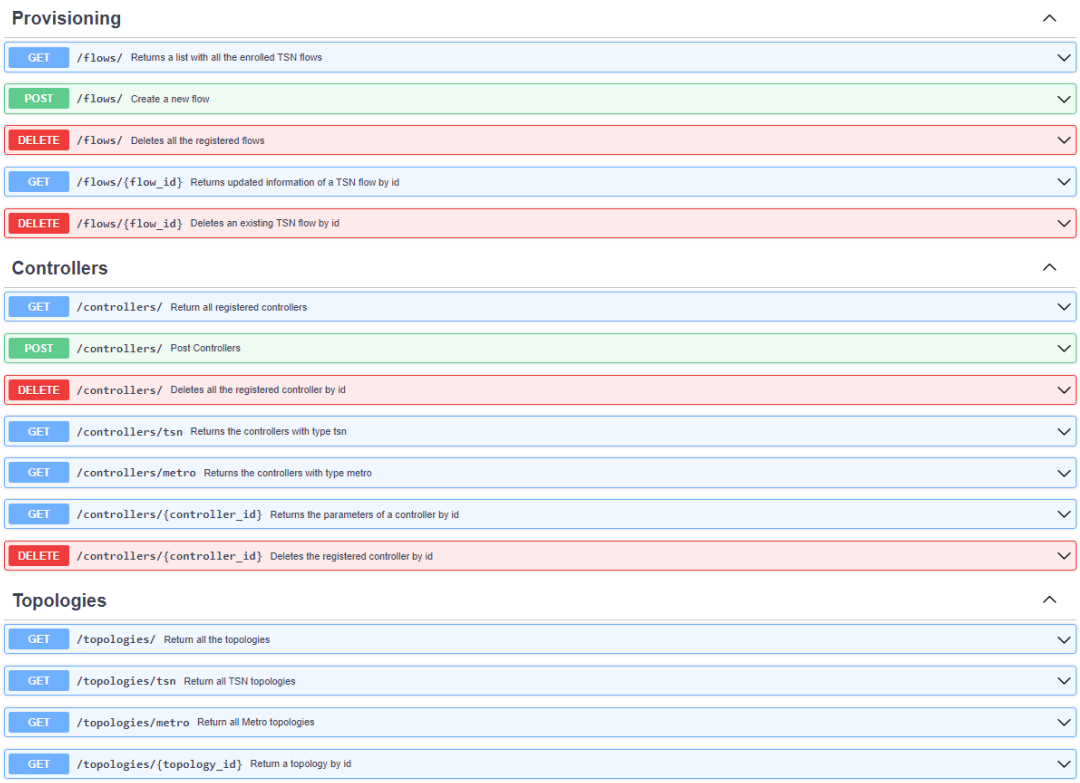

Figure 20. TSN Connectivity Manger NBI current implementation

6.2.1 Controllers

The controllers group allows users to manage SDN controller records. It provides the ability to view all controllers, add new controllers, and delete controllers either individually or all at once. It also categorizes controllers based on their types, such as TSN or metro, allowing for filtered views.

- POST /controllers/: Creates a new controller record.
- DELETE /controllers/: Removes all controllers.
- GET /controllers/tsn: Lists controllers specifically of type TSN.
- GET /controllers/metro: Lists controllers specifically of type metro.
- GET /controllers/{controller_id}: Retrieves the parameters of a controller by its ID.
- DELETE /controllers/{controller_id}: Deletes a specific controller by its ID.

Compared to the NBI definition exposed in D1.2, in this version of the controllers' functional group the *PUT/controllers/* operation/endpoint has been removed. This modification, implemented to simplify the interface and support the assumption of static controller information post-registration in the TSN CM's system. Furthermore, it's important to highlight that in the initial NBI structure, this functional group was integrated within the topological functional group. However, in the current framework, its functionalities and endpoints have been reorganized into a separate, independent division.

6.2.2 Topologies

In the topologies group, the focus is on managing network topologies. The NBI allows users to view all topologies, retrieve topologies filtered by type (TSN or metro), and access detailed information about a specific topology by its ID.

- GET /topologies/: Retrieves all topology records.
- GET /topologies/tsn: Fetches all TSN topologies.
- GET /topologies/metro: Fetches all metro topologies.
- GET /topologies/{topology_id}: Retrieves a specific topology by its ID.

In the latest iteration of the NBI API, significant modifications have been implemented compared to the version detailed in Document D1.2, manifesting in three key aspects. Firstly, as previously mentioned, the controller functionalities have been segregated from the topology group. Secondly, the new design presupposes a single network per topology, eliminating the necessity for an endpoint to fetch network information, which is now intrinsically integrated into the topology data. Thirdly, this version restricts operations to GET requests only. The underlying justification for these alterations stems from the TIMING project's assumption of static topologies. This implies that the network's nodes and links remain constant, thereby obviating the requirement for additional interactive functionalities.

6.2.3 Provisioning

This section of the NBI focuses on the management of flows within the network, providing the ability to list all current flows, create new flows, delete specific flows, or delete all flows, thus offering comprehensive control over the flow management.

- GET /flows/: Retrieves a list of all the enrolled TSN flows.
- POST /flows/: Allows for the creation of a new TSN flow.
- DELETE /flows/: Enables the deletion of all registered flows.
- GET /flows/{flow_id}: Fetches updated information of a specific TSN flow by its ID.
- DELETE /flows/{flow_id}: Removes a specific TSN flow by its ID.

In the current version of the NBI, the approach to managing flows has been refined with a focus on data model implementation (request/response) for both upstream and downstream flows. However, these concepts are not featured as explicit functionalities within the NBI itself, resulting in the removal of endpoints associated with them. Additionally, this implementation does not accommodate updates to flows; consequently, the PUT operation has been excluded.

For future deliverables, it is expected to provide more details about the NBI implementation, and the definition of the concrete data models associated (requests/responses) in deliverable D1.1 of the TIMING's SP2.

6.3 UPDATES OF THE ENP FRONT-END (GUI)

The e-Lighthouse Network Planner (ENP) serves as an essential component in the TSN Connectivity Manager, providing a graphical user interface (GUI) to facilitate the planning and management of network domains. This section delves into the updates made in the Front-End module (GUI), incorporated into the ENP tool. Each update, classified by type, is represented in

the attached images and described below. These updates aim to refine the user experience, improve the precision of network modeling, and extend the functionality to align with the TIMING project's objectives as outlined in the deliverable D1.2.

6.3.1 Topological Information Visualization Updates

Several updates have been introduced in the latest version of the ENP's Front-End to facilitate the management and the comprehension of topological information from a visual perspective. Here, are exposed some of the improvements applied.

In Figure 21, it is shown IP demand details in the GUI's topology panel, that, now includes a geographical (and logical) representation of network IP demands, adding a spatial dimension to data interpretation. This update allows for an easier assessment of network load and demand distribution across different geographic locations.

It is worth noting that the visualization has been significantly improved to facilitate the multi-layer understanding by also including a representation of the path in the underlying layers. Also, a mini-dashboard associated with this IP demand has been developed to summarize the key metrics related to the flow (or flows) related to such IP demand.

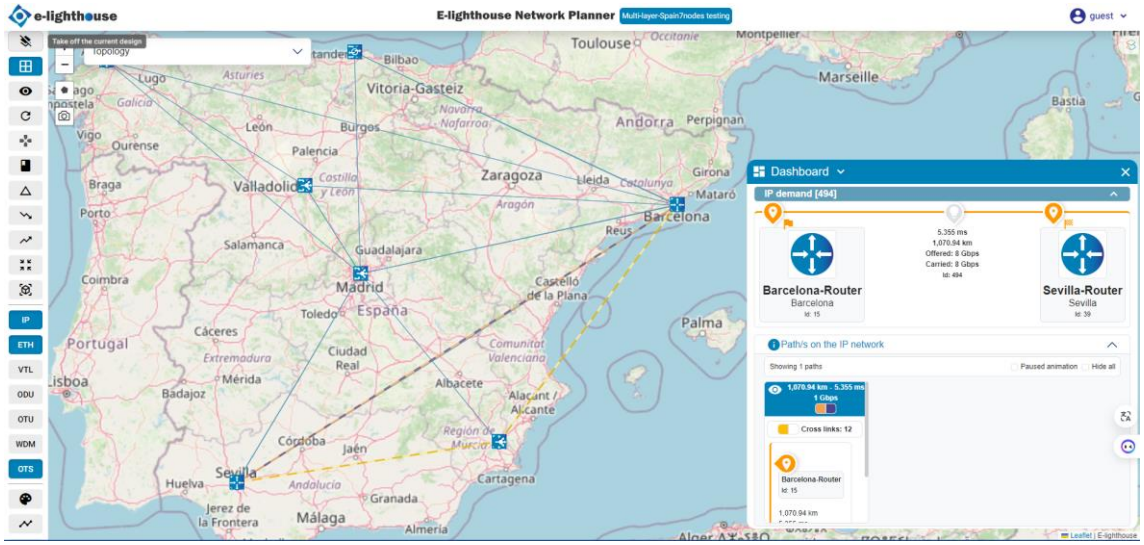

Figure 21. ENP GUI: IP demand details in geographical layout

Similar to IP demands, Figure 22 depicts updates to the node visualisation, which presents mini-dashboards summarising the current status of the node in question. The improved dashboard offers a more intuitive visualisation of nodes and their connections, providing an immediate understanding of the network configuration and status.

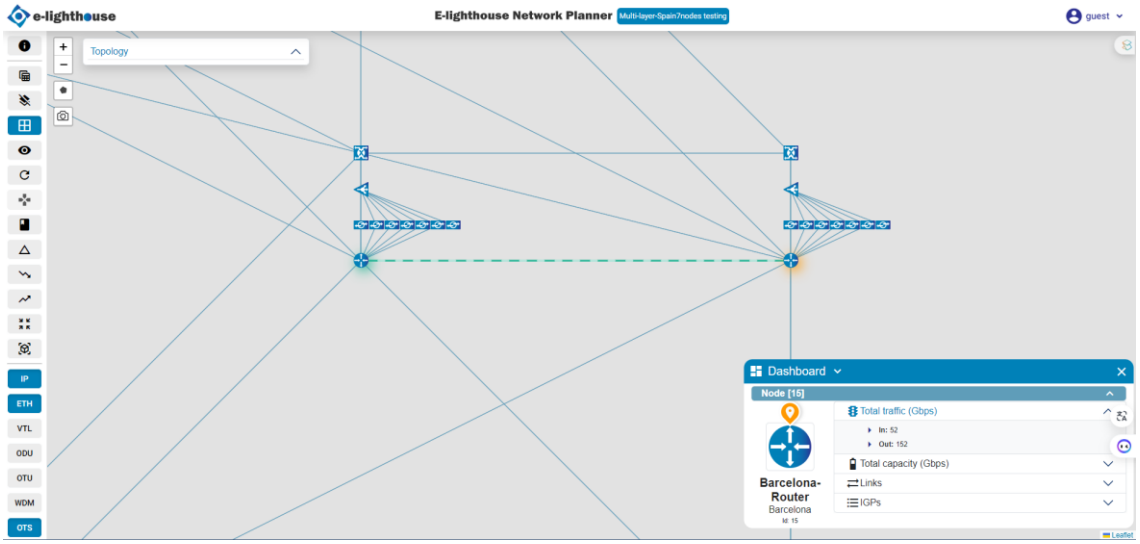

Figure 22. ENP GUI: Node mini-dashboard in logical layout

6.3.2 Control Panel Upgrades

In the ENP's GUI, the control panel serves as a visually intuitive interface, meticulously designed to arrange, and display comprehensive network data through user-friendly and adaptable tables. This design choice significantly augments the user experience by offering robust filtering and sorting capabilities, allowing for streamlined access and manipulation of information within the tables.

The initial Control Panel was designed to give users control over the configuration and monitoring of network elements, but it may have lacked depth in data analytics and customization options. The updated Control Panel (Figure 23) showcases a detailed view of IP links, latency metrics, traffic data, and more. The design appears to be more intuitive and user-friendly, with a focus on easy access to critical network parameters.

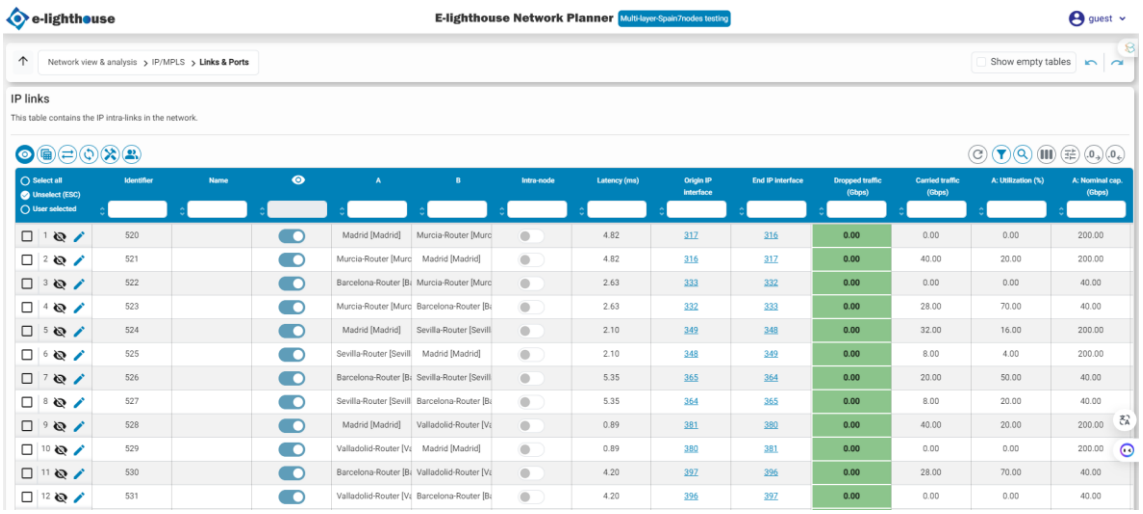

Figure 23 - ENP GUI: Control panel

6.3.3 Summary Report Enhancements

The TSN CM's primary function is to meticulously survey the network to identify all available resources and capacities. It then compiles a comprehensive summary of significant alterations applied to the controllers within the network, post-optimization exercises. These enhancements ensure the ENP tool remains at the forefront of network planning applications, offering comprehensive reporting features that streamline the configuration process within real TSN networks.

The IP Summary Report in D1.2 likely provided essential information on network performance but limited in the scope of data presented and the analytical tools available. The new IP Summary Report, shown in Figure 24, has been enriched with detailed traffic analysis, over-subscription traffic data, and a more detailed breakdown of port utilization. This enables more precise network performance monitoring and capacity planning.

Figure 24. ENP GUI: IP summary report

The updates to the TSN Connectivity Manager GUI in the ENP tool demonstrate a commitment to improving the network planning and management experience. The enhancements have focused on providing more in-depth analytics, a more user-friendly interface, and better integration of data visualization tools, which are essential for managing sophisticated telecom infrastructures. This analysis sets the stage for a detailed review of each updated component and its contributions to the project's overarching goals.

7 SCHEDULING ALGORITHMS

This section includes updated specifications for the scheduling component of the TIMING project, along with a description of four performance evaluation tests and a preliminary performance evaluation.

The scheduling component design encompasses the design of TSN Wi-Fi windows for isochronous traffic, the design of a reinforced learning-based DL/UL splitter, and the scheduling of asynchronous traffic.

For the specification of the TSN Wi-Fi windows for synchronous traffic, not reported in the previous deliverable D1.2, we begin by presenting for the downlink (DL) a bottom-up design, that the TSN Ethernet windows design is assumed to follow. Secondly, we present a top-down design restricted to compatibility with predefined TSN windows. Finally, we extend the scope to the uplink (UL). Preliminary evaluations in terms of the throughput achieved for synchronous traffic are included.

Once isochronous traffic has been taken care of, smart scheduling will improve the throughput and quality of Service (QoS) of asynchronous traffic. To that end, the DL/UL splitter aims to intelligently allocate free slots within the superframe between DL and UL. The updated specifications for the DL/UL splitter using reinforcement learning presented in this section include: 1) the definition of states and actions within the reinforcement learning framework and 2) the design of a strategy to minimize imbalances in DL and UL queues, implicitly reducing the latency of asynchronous traffic. A preliminary performance evaluation is also presented to demonstrate the viability of the reinforcement learning-based approach.

Finally, a structured set of evaluation tests for the scheduling module is presented in this section. Specifically, we present four evaluation tests leveraging an isolated windows design for isochronous traffic. They begin by focusing on the validation of the Wireless Flow Scheduler (WFS) (test 1), upgrading to include a preliminary version of the UL/DL splitter (test 2), followed by the evaluation of the throughput and delay of admitted asynchronous traffic given a load of isochronous traffic (test 3), and the assessment of the performance improvement achieved by activating the Baseline UL/DL Splitter to redistribute free slots (test 4). A preliminary performance evaluation is included following each test.

7.1 TSN WINDOWS DESIGN FOR ISOCHRONOUS TRAFFIC

In this section, we study the minimum number of resource blocks (RB) that need to be reserved for accommodating N isochronous flows in the WiFi superframe. The discussion primarily centers on the downlink (sections 7.1.1 and 7.1.2) and is later extended to the uplink (section 7.1.3). To facilitate integration into the Ikerlan's WiFi node, we consider that the N flows are multiplexed in time, i.e., TDMA. Consequently, the available resource blocks are time slots, with the slot being the minimum unit of scheduling.

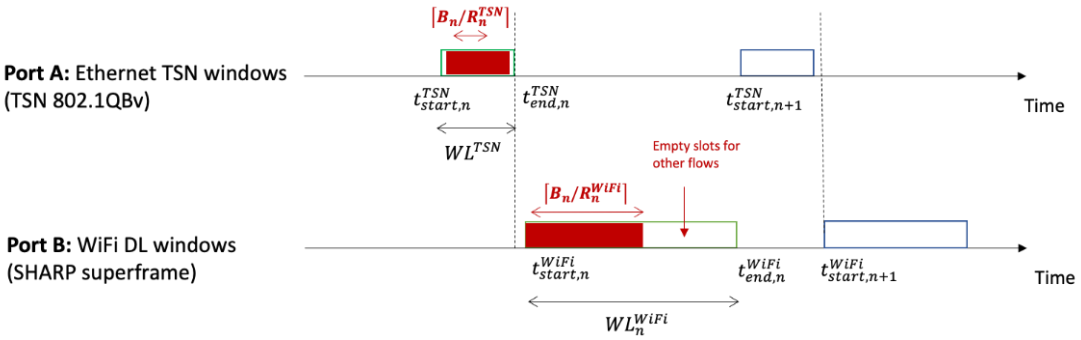

Figure 25. Definition of the WiFi superframe windows for ISO traffic

For every active isochronous flow, we define a window in the WiFi superframe. As indicated in Figure 25, the window for the nth flow opens at slot $t_{start,n}^{WiFi}$ and closes at slot $t_{end,n}^{WiFi} = t_{start,n}^{WiFi} + WL_n^{WiFi}$, where WL_n^{WiFi} represents the window length (in slots). The N windows are designed under the following conditions:

1) Causality. The nth WiFi window cannot be opened until the corresponding TSN window is closed, i.e., $t_{start,n}^{WiFi} \geq t_{end,n}^{TSN}$. In other words, the B_n bits of the nth flow should have been received before the WiFi window is opened. If feasible, the window will be opened as soon as possible to minimize latency, specifically $t_{start,n}^{WiFi} = t_{end,n}^{TSN}$.

2) Bounded delay. The delay of the nth flow in the WiFi node is bounded, i.e., $D_n \leq \breve{D}_n = t_{end,n}^{WiFi} - t_{end,n}^{TSN}$. Assuming that it is feasible to satisfy $t_{start,n}^{WiFi} = t_{end,n}^{TSN}$, the maximum delay $\breve{D}_n = t_{end,n}^{WiFi} - t_{start,n}^{WiFi} = WL_n^{WiFi}$ is directly determined by the window length. Therefore, the shorter the window, the lower the latency of the flow.

3) Reduced packet loss. The window length WL_n^{WiFi} will be designed to ensure a low probability of not having sufficient time to transmit the incoming B_n bits (equivalent to one L2 frame or packet) during the current window. If this happened, the delay would exceed $\breve{D}_n = WL_n^{WiFi}$ because the packet transmission would extend into the next superframe. In such cases, the packet is declared lost. By design, this occurrence is limited to a probability not exceeding $\breve{p}_n$.

4) Zero Jitter. Due to the variable time required to transmit a flow through the wireless channel, the WiFi link introduces jitter, i.e., the inter-packet arrival time is not constant at the receiver. This jitter can be compensated for by the transmitter if the transmission of the nth flow consistently concludes at the same position within the superframe, i.e., the nth flow always ends at time $t_{end,n}^{WiFi}$. Alternatively, the transmitter can initiate transmission as soon as possible, and let the end users to perform delay compensation (hold-and-forward) for de-jittering the sequence of received packets. In such a case, if a packet is received ahead of schedule, the receiver will retain the packet until the end of the window ($t_{end,n}^{WiFi}$) and, at that point, it will either use the packet if it is the end user or forward the packet to the next node if not. We will adopt this alternative approach as it is more straightforward from an implementation point of view: it allows efficient window interlacing without requiring discontinuous transmission.

Objective

The objective is to design $\{t_{start,n}^{WiFi}\}$ and $\{t_{end,n}^{WiFi}\}$ so that the previous conditions are satisfied and the time required to multiplex the N isochronous flows is the minimum possible.

7.1.1 Botton-up design (assuming a compatible TSN Ethernet will be feasible)

Focusing on the downlink and following a bottom-up design, we will design first the WiFi superframe skipping condition 1 and assuming that a compatible TSN Ethernet frame exists that satisfies $t_{end,n}^{TSN} \leq t_{start,n}^{WiFi}$ (condition 1). Later, we will address the design in case the TSN controller has already defined the TSN frame and we have to design the WiFi windows under constraint 1 (top-down design). In this case, the WiFi node is provided with $\{t_{end,n}^{TSN}\}$ as input values and must ensure that $t_{start,n}^{WiFi} \geq t_{end,n}^{TSN}$ for all n.

We assume that at the beginning of the superframe, the scheduler has the following information for all N flows:

1) Transmission rate (R_n^{WiFi}). R_n^{WiFi} is slowly time-varying during the nth flow connection. The channel variability is measured by the coherence time. R_n^{WiFi} is a function of the instantaneous signal-to-interference-plus-noise ratio (SINR). The WiFi node is expected to periodically estimate the SINR and consequently select the most favorable modulation and coding scheme (MCS).

Table 14 applies to WiFi6. It was included already in D1.2 but is included again here as a reference.

MCS	MOD	COD	Rate (bits/slot)	Rate (Mbps)	Req SINR (dB)
0	BPSK	1/2	117	8.6	-0.5
1	QPSK	1/2	234	17.2	2.5
2	QPSK	3/4	351	25.8	5
3	16-QAM	1/2	468	34.4	8
4	16-QAM	3/4	702	51.6	11
5	64-QAM	2/3	936	68.8	15.25
6	64-QAM	3/4	1053	77.4	16.5
7	64-QAM	5/6	1170	86.0	18
8	256-QAM	3/4	1404	103.2	21.75
9	256-QAM	5/6	1560	114.7	23.5
10	1024-QAM	3/4	1755	129.0	27
11	1024-QAM	5/6	1950	143.4	29

Table 14. MCS table from 802.11ax standard [Table 27-79, WLA21] considering 1 spatial stream, LDPC coding, $T_{slot} = T_{ofdm} = 13.6\mu s$ (GI=800ns) and a band of 242 tones (242·78.125kHz=19.1MHz).

2) Burst length (B_n). B_n can also vary over time, but in the context of Timing, we assume that B_n remains fixed throughout the entire connection.

To ensure zero jitter (condition 4), the design of the WiFi windows must remain static for the whole connection time. Accordingly, the windows' edges are designed offline based on the following statistical information:

1) Transmission rate statistics. Probability mass function of R_n^{WiFi}. We model R_n^{WiFi} as a random variable and we assume that its probability mass function has been determined offline: $\Pr(R_n^{WiFi} = r_i) = p_i$, where r_i represents the rate in bits/slot associated with the ith MCS ($i = 0, \dots, i_{max}$), and p_i denotes the probability or frequency of using this MCS during the connection. If the SINR is not sufficiently high for reliable transmission using the lowest MCS, we consider that the packet is lost. This occurs with a probability q. In such case, we consider that $R_n^{WiFi} = 0$ (no transmission)

and $\Pr(R_n^{WiFi} = 0) = q$. For simplicity, we assume that the rate of all flows is identically distributed, so p_i and r_i do not depend on n.

2) Burst length statistics. Probability mass function of B_n. We can also model B_n as a random variable. However, within the context of Timing, it is assumed to be deterministic and constant.

Based on the above information, two designs are proposed for the WiFi windows:

1) Isolated Windows. We impose that $t_{start,n+1}^{WiFi} \geq t_{end,n}^{WiFi}$ and design the N windows separately. Formally, the length of the nth window WL_n^{WiFi} is selected so that

$$\Pr(B_n/R_n^{WiFi} \geq WL_n^{WiFi}) \leq \check{p}_n.$$

Because $R_n^{WiFi} \in \{0, r_0, r_1, \dots, r_{i_{max}}\}$ is a discrete random variable, it follows that

$$WL_n^{WiFi} = \lceil B_n/r_{t_n} \rceil$$

with t_n being the index of the best MCS that allows designing the nth window with the agreed probability of loss $\check{p}_n$. In particular, t_n is the highest MCS holding that $q + \sum_{i=0}^{t_n-1} p_i \leq \check{p}_n$.

If it results that $t_n = 0$, the design is carried out for the worst case (MCS 0), and the window length is set to the maximum value.

If $t_n > 0$, the window length can be reduced while still holding the outage probability $\check{p}_n$. The minimum number of slots that are required to accommodate N downlink flows is given by

$$T_{DL} \geq \sum_{n=1}^{N} WL_n^{WiFi} = \sum_{n=1}^{N} \lceil B_n/r_{t_n} \rceil = N \cdot \lceil B/r_t \rceil$$

where the last equality holds in the case of N identical flows.

At the beginning of the current superframe, the instantaneous rate R_n^{WiFi} of flow n is estimated. If $R_n^{WiFi} < r_{t_n}$, the packet is lost. If $R_n^{WiFi} = r_{t_n}$, the packet will occupy the entire window whereas, if $R_n^{WiFi} > r_{t_n}$, only the first $\lceil B_n / R_n^{WiFi} \rceil$ slots of the window are occupied, as indicated in Figure 25. The remaining slots, $WL_n^{WiFi} - \lceil B_n / R_n^{WiFi} \rceil$, will be assigned to other classes of traffic (asynchronous TSN and best-effort).

2) Interlaced Windows. We admit that $t_{start,n+1}^{WiFi} < t_{end,n}^{WiFi}$ allowing the windows to be interlaced. The windows are designed sequentially. The first window starts at $t_{start,1}^{WiFi} = 0$, and its length WL_1^{WiFi} is determined as in the isolated case:

$$\Pr(B_1/R_1^{WiFi} \geq WL_1^{WiFi}) \leq \check{p}_1.$$

The design of the second window is as follows:

$$\Pr\left(B_2/R_2^{WiFi} \geq WL_2^{WiFi} - NOS_2\right) \leq \check{p}_2$$

with NOS_2 a random variable that indicates the *Number of Occupied Slots* by flow 1 in the second window. Since $NOS_2 \geq 0$, if $\check{p}_2 = \check{p}_1$ and $B_2 = B_1$, WL_2^{WiFi} will be larger than WL_1^{WiFi}, increasing somewhat the latency of flow 2. On the other hand, overlapping will allow to reduce the total time reserved for flows 1 and 2, $t_{end,2}^{WiFi} - t_{start,1}^{WiFi} = t_{end,2}^{WiFi}$, as shown in Figure 26.

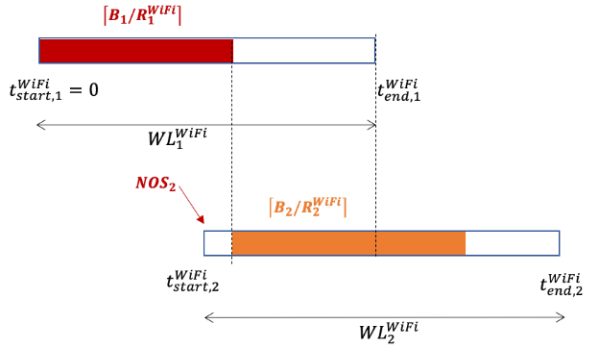

Figure 26. Allocation of the B2 bits of the second flow in its window (orange color).

Since the first window is already designed, the overlap $\Delta_2 = t_{end,1}^{WiFi} - t_{start,2}^{WiFi}$ is determined by the start of the second window ($t_{start,2}^{WiFi}$). If $\Delta_2 = WL_1^{WiFi}$ ($t_{start,2}^{WiFi} = t_{start,1}^{WiFi}$), we have full overlap. In this case, WL_2^{WiFi} takes the largest value (worst latency) but the number of slots reserved for the first two flows $t_{end,2}^{WiFi}$ is minimum (best multiplexing). On the other extreme, if $\Delta_2 = 0$ ($t_{start,2}^{WiFi} = t_{end,1}^{WiFi}$), there is no overlapping and we end up with isolated windows, as in the first design. In this case, the delay is minimum but the number of required slots $t_{end,2}^{WiFi}$ is substantially larger.

For intermediate values of Δ_2 there is a trade-off between latency (WL_2^{WiFi}) and multiplexing efficiency ($t_{end,2}^{WiFi}$), as shown in Figure 27. If the latency WL_2^{WiFi} is fixed to a given value $\breve{D}_2$, the optimum overlap $\breve{\Delta}_2$ can be obtained by simulation as indicated in the following figure. As it can be appreciated, there is an optimal overlap, approximately 30%, that allows to reduce the frame length $t_{end,2}^{WiFi}$ from 104 slots to about 87 slots without increasing at all the latency of the second flow ($WL_2^{WiFi} = 55$).

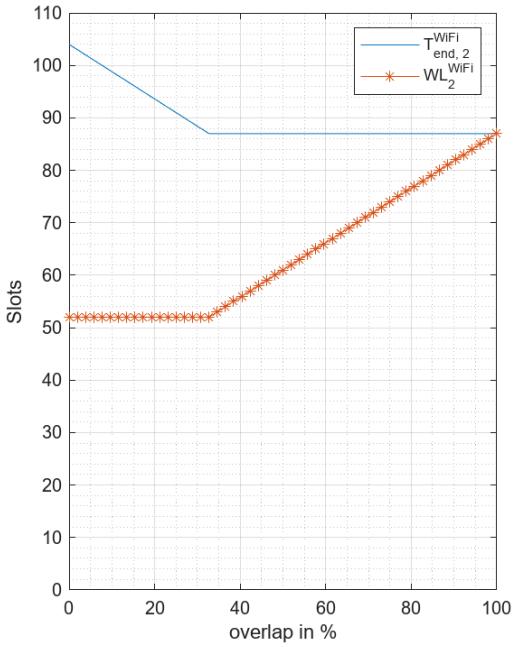

Figure 27: Design of window 2: latency vs. multiplexing efficiency trade-off for IEEE 802.11ax [WLA21], $WL_1^{WiFi} = 52$, $B_n = 1500\ (bytes)$, $\breve{p}_n = 2 \cdot 10^{-2}$, $BW = 20MHz, B_{coh} >> BW$ (flat-fading channel), $T_{slot} = 13.6\mu s, E\{SNR_n\} = 20dB$

The rest of windows are designed similarly applying:

$$\Pr\left(B_n/R_n^{WiFi} \geq WL_n^{WiFi} - NOS_n\right) \leq \check{p}_n \qquad n = 3, \ldots, N$$

where NOS_n is a random variable that accounts for the slots in the nth window already occupied by the previous flow.

The use of interlaced windows reduces the time required to multiplex N isochronous flows. This allows to work with shorter superframes and hence shorter control cycles or, alternatively, to increase the number of flows N that can be transmitted in a given superframe length.

Adopting a bottom-up design, we have designed the windows of the WiFi link without any external constraint assuming that the windows of the input Ethernet port will be designed later so that they are compatible with the designed WiFi windows. As indicated previously, compatibility means that $t_{end,n}^{TSN} \leq t_{start,n}^{WiFi}$ (causality condition).

In a typical scenario, the transmission rate over the Ethernet cable will be much higher than over the wireless link and, consequently, TSN windows are expected to be much shorter than WiFi windows. As shown in Figure 28, if the TSN windows are consecutive in time, in general, this will permit to overlap optimally the WiFi windows following the bottom-up design presented in this section.

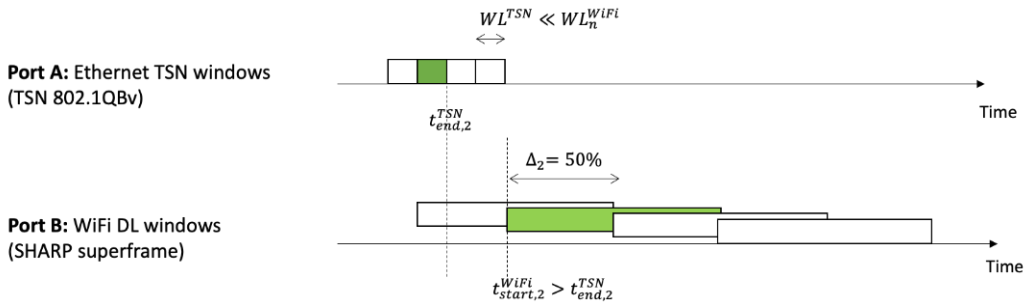

Figure 28. Representative scenario allowing interlaced WiFi windows (bottom-up design). We represent four DL flows with 50% window overlap in the wireless link. As indicated, causality holds for the second flow ($t_{end,2}^{TSN} < t_{start,2}^{WiFi}$) and the same occurs with the other 3 flows.

7.1.2 Top-down design (design compatible with pre-defined TSN windows)

In this section, we address the design in case the TSN controller has already defined the Ethernet TSN frame and we are obliged to satisfy the causality constraint: $t_{start,n}^{WiFi} \geq t_{end,n}^{TSN}$ for all n.

Starting from the Bottom-up design described in section 7.1.1, we follow the following procedure to impose the causality constraint to all the windows:

- First, we fix $t_{start,1}^{WiFi} = t_{end,1}^{TSN}$

- If $t_{start,2}^{WiFi} < t_{end,2}^{TSN}$, the start of the second WiFi window must be delayed $\Delta t_2 = t_{end,2}^{TSN} - t_{start,2}^{WiFi}$ slots. As a consequence of this delay, the overlap Δ_2 of windows 1 and 2 is reduced and its length WL_2^{WiFi} has to be recomputed.

As indicated previously, reducing the overlap will reduce WL_2^{WiFi} (lower latency) but it will also postpone the end of the second window $t_{end,2}^{WiFi}$ (worse multiplexing). Note that, if the required

Δt_2 is sufficiently large, the overlap will be zero ($\Delta_2 = 0$). See Figure 29 for a graphical representation.

- Proceed in the same way with next windows one after the other (n=3, 4, 5, etc.).

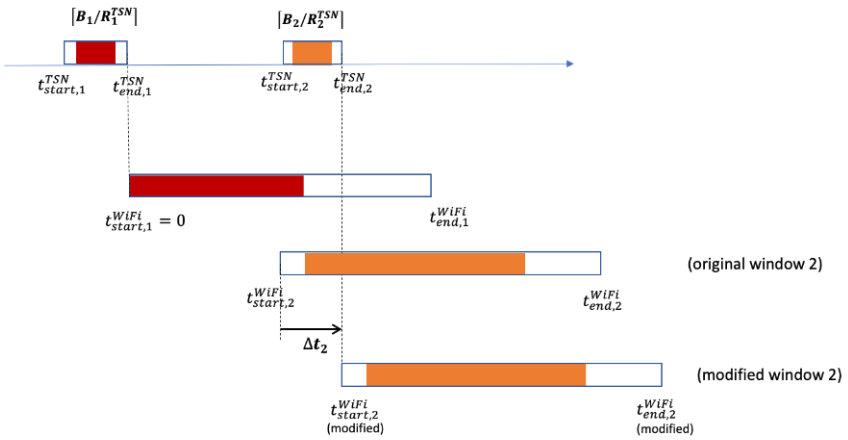

Figure 29. Modified design of window 2 to force $t_{start,2}^{WiFi} \geq t_{end,2}^{TSN}$

Note that the proposed top-down design does not increase the windows length and, therefore, it does not increase the latency of flows. If the latency were not a limiting factor, we could use the original bottom-up design (best multiplexing) and delay all the windows $\Delta t = \max_n(t_{end,n}^{TSN} - t_{start,n}^{WiFi})$ slots to guarantee that $t_{start,n}^{WiFi} + \Delta t \geq t_{end,n}^{TSN}$ for all n. See Figure 30. In general, this approach is not feasible because it increases excessively the flows delay. An intermediate solution is possible in which we first delay all the flows $\Delta t' < \Delta t$ and, afterwards, we impose individual delays Δt_n to guarantee causality.

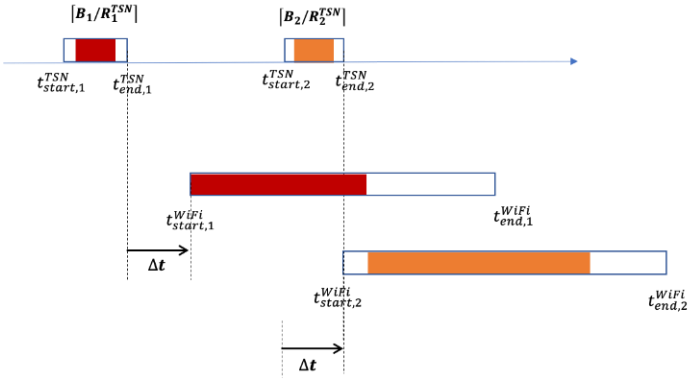

Figure 30. Alternative modified design of window 2 to force $t_{start,2}^{WiFi} \geq t_{end,2}^{TSN}$

The proposed top-down design guarantees compatibility with the input TSN frame but the gain of overlapping windows is reduced with respect to the unconstrained bottom-up design. This reduction depends on the tolerated latency and the layout of the input TSN windows.

7.1.3 Extension to the Uplink

To facilitate synchronization of uplink bursts, the uplink windows have to be extended:

$$WL_n^{WiFi,UL} = WL_n^{WiFi,DL} + T_{p,UL}$$

with $T_{p,UL}$ the uplink L1 overhead (in slots) that includes the uplink preamble and a guard interval (interframe spacing).

7.1.4 Preliminary Performance Evaluation

In this section, we present some simulations showing the benefits of the interlaced windowing design in terms of throughput.

The first experiment corresponds to the downlink of a WiFi 6 network with N stations connected to the AP through independent channels. The average received SNR is the same for all users and is set to 20dB. Each station receives a single isochronous flow. The packet length is set to 1500 bytes (maximum payload length in Ethernet). The simulation parameters are summarized in Table 15.

Model	Parameter	Value
Pathloss (slow fading)	Average Received SNR	20 dB
	Time variability	Constant
Multipath (fast fading)	Delay Spread	50ns
	Power Delay Profile	Exponential
	Statistics	Rayleigh
	Coherence Time	30ms
	Doppler Spectrum	Block fading
Superframe Structure	Slot duration (τ_{slot})	1 OFDM symbol
	OFDM Symbol Duration	13.6us (12.8us+0.8us)
	Bandwidth	20MHz
	Number of data subcarriers	242
	MCS	0, ..., 11
	MCS performance	802.11ax (see Table 14)
ISO Traffic	Periodicity	Superframe duration
	Packet length (B)	1500 bytes = 12000 bits

Table 15. Simulation setup for evaluating the interlaced window design (802.11ax WLA21])

Interlacing windows allows for a more compact superframe, as appreciated in Figure 31. The maximum delay of flow n is determined by its window length (WL_n^{WiFi}). As the delay constraint becomes more restrictive, the overlap gradually decreases and, as a consequence, the frame length augments. Note that if $WL_n^{WiFi} = 26$ (for all n), which is the window length of the isolated windows' design (minimum possible value), the interlaced windows configuration still provides a shorter superframe than with isolated windows. Moreover, for $WL_n^{WiFi} = 52$, doubling the minimum value, the superframe is as compact as with full overlap ($t_{start,n}^{WiFi} = 0$ for all n).

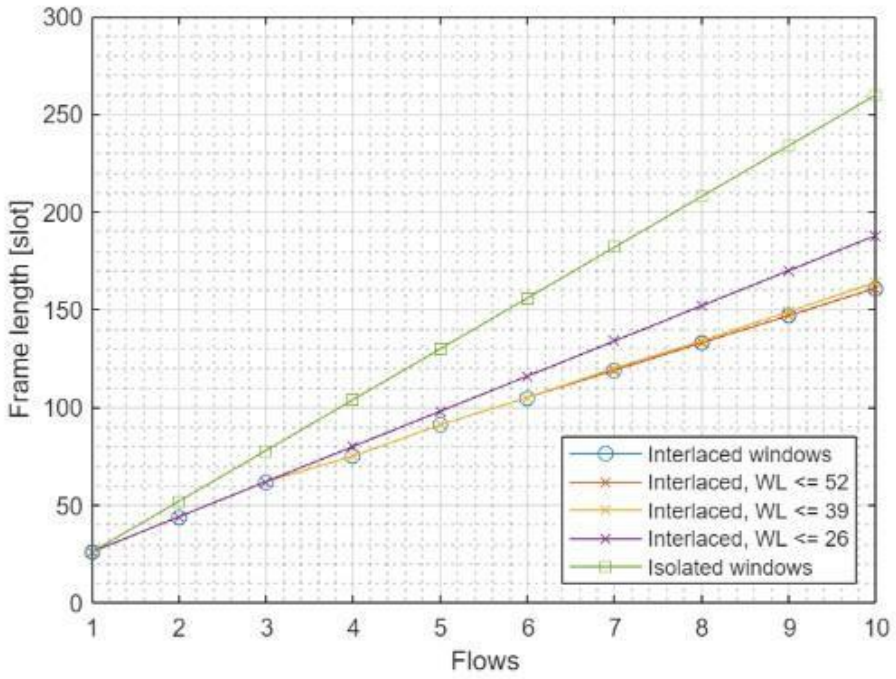

Figure 31. TDL vs. N for different constraints on the window length (WL), for $B_n = 1500\ (bytes)$, $\check{p}_n = 2 \cdot 10^{-2}$, $T_{slot} = 13.6\mu s, E\{SNR_n\} = 20dB$

Complementarily, Figure 32 shows the overall downlink throughput assuming that the UL portion of the superframe has the same duration as its DL counterpart. The throughput is computed as follows: $TH_{DL} = \frac{B \cdot N}{(2 \cdot T_{DL}) \cdot \tau_{slot}}$, where N is the number of flows and B the packet length in bits. The reduced duration of the superframe in the interlaced design results in a very significant increase in the overall DL throughput.

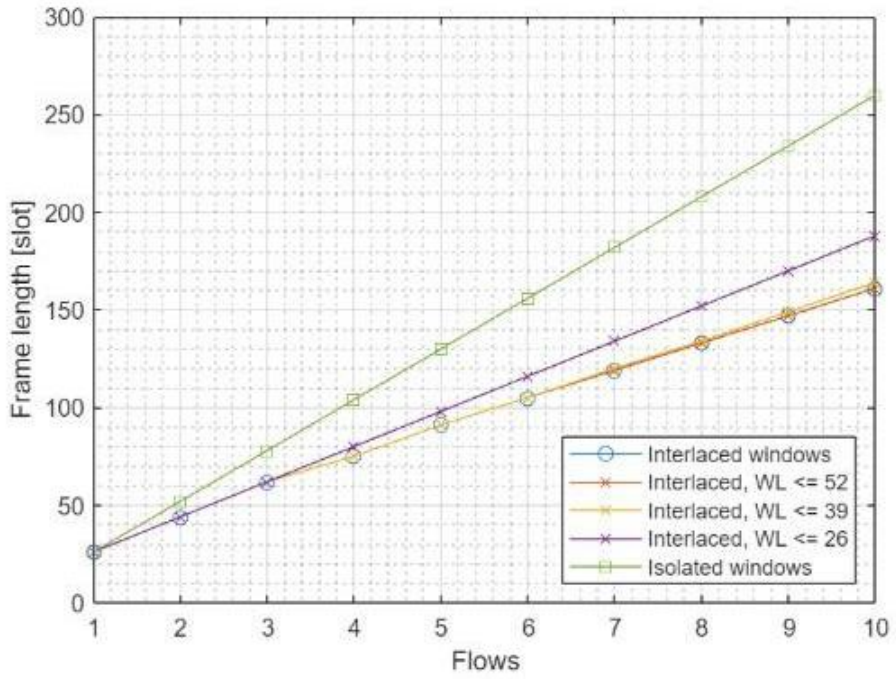

Figure 32. DL throughput for different constraints on the window length (WL), for $B_n = 1500\ (bytes)$, $\check{p}_n = 2 \cdot 10^{-2}$, $T_{slot} = 13.6\mu s, E\{SNR_n\} = 20dB$

Next, a second simulation is carried out for the DL of a 802.11a/g network [WLA99] with N stations connected to the AP through independent propagation channels. This simulation is included to assess the gain of the interlaced design when applied to the WiFi implementation that is adopted in IKL's SHARP platform. The simulation parameters are summarized in Table 16.

Model	Parameter	Value
Pathloss (slow fading)	Average Received SNR	20 dB
	Time variability	Constant
Multipath (fast fading)	Delay Spread	50ns
	Power Delay Profile	Exponential
	Statistics	Rayleigh
	Coherence Time	30ms
	Doppler Spectrum	Block fading
Superframe Structure	Slot duration (τ_{slot})	1 OFDM symbol
	OFDM Symbol Duration	4us (3.2us+0.8us)
	Bandwidth	20MHz
	Number of data subcarriers	48
	MCS	0, ..., 7
	MCS performance	802.11a/g table (Table 19)
ISO Traffic	Periodicity	Superframe duration
	Packet length (B)	30 bytes = 240 bits

Table 16. Simulation setup for evaluating the interlaced window design

Figure 33 shows the simulation results of interlacing windows with parameters specific to the Wi-Fi 3 standard (IEEE 802.11a/g [WLA99]). Due to the discretization of time in slots, packets that are too small would not benefit from the advantage that interlacing provides. However, in the Figure 33 the positive effect of interlacing is still very much noticeable in the form of more compact superframes compared to isolated windows, even with a small packet size (30 bytes).

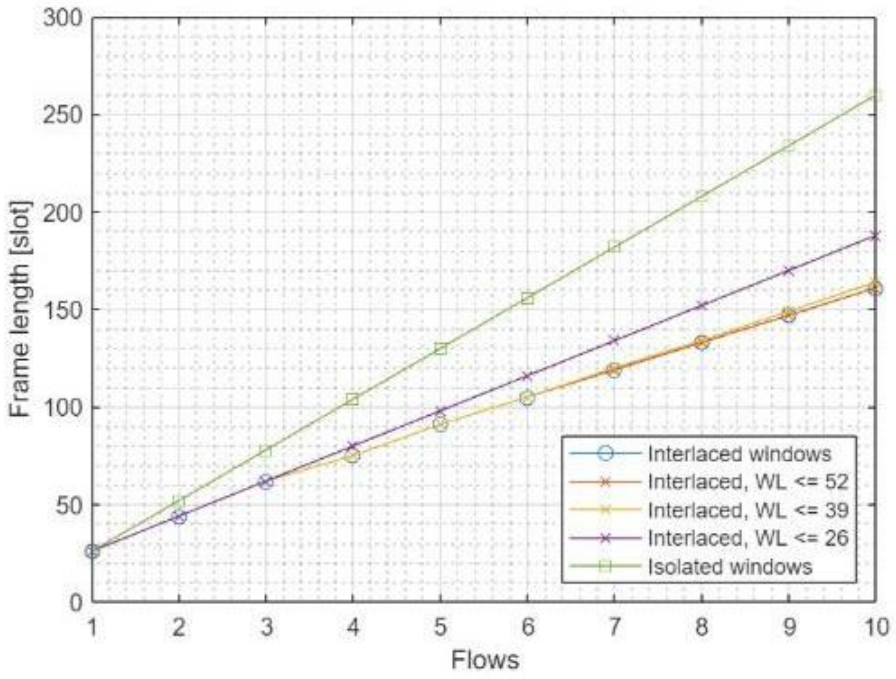

Figure 33. TDL vs. N for different constraints on the window length (WL), for $B_n = 30\ (bytes)$, $\check{p}_n = 10^{-2}$, $T_{slot} = 4\mu s, E\{SNR_n\} = 20dB$

Finally, Figure 34 shows the overall downlink throughput, computed as follows: $TH_{DL} = \frac{B \cdot N}{(2 \cdot T_{DL}) \cdot \tau_{slot}}$, where N is the number of flows and B the packet length in bits. Again, the compactness of the superframe achieved by interlacing is also appreciated in the form of increased throughput.

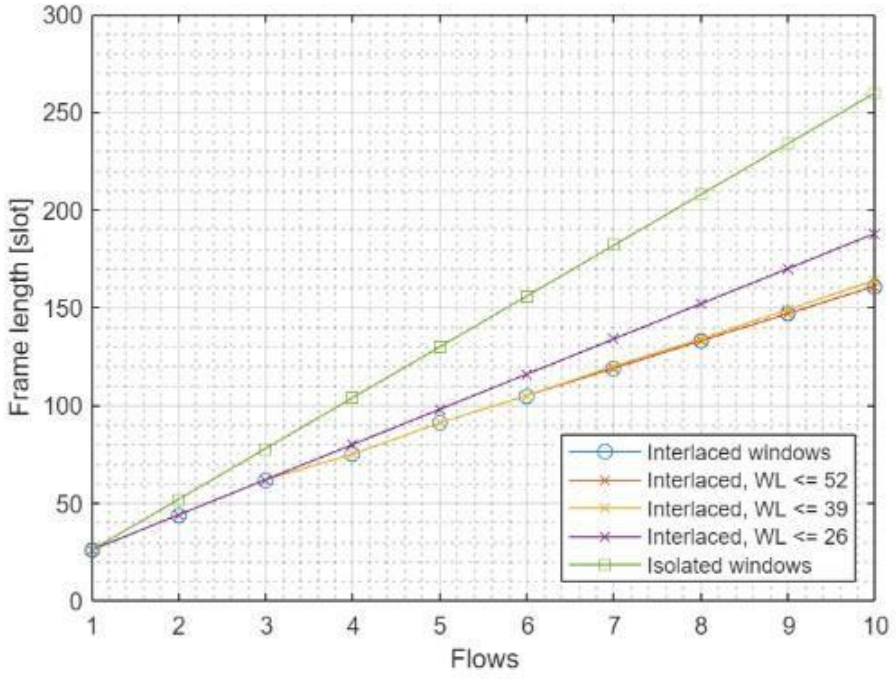

Figure 34. DL throughput for different constraints on the window length (WL), for $B_n = 30\ (bytes)$, $\check{p}_n = 10^{-2}$, $T_{slot} = 4\mu s, E\{SNR_n\} = 20dB$

7.2 DL/UL SPLITTER: UPDATED SPECIFICATIONS

After designing the TSN Wi-Fi windows for isochronous traffic using one of the strategies presented in section 7.1, the number of free slots in the superframe can be computed. In this section, we present a Reinforcement Learning (RL) based DL/UL splitter (RL Splitter) to allocate the free slots within the superframe between DL and UL.

The DL/UL splitter subsystem operates at the superframe time periodicity. At the onset of each superframe, depending on the state of the asynchronous traffic queues with quality of service (QoS), it decides the proportion of free slot distribution between DL and UL. If there was no QoS traffic, then the decision would be made based on best effort (BE) traffic.

Let us assume that once the isochronous traffic has been allocated in the superframe, the number of non-occupied slots is denoted as N and the instantaneous aggregated content given in bits of the DL queues at the end of slot n is measured using the variable $q_{DL}\ (n)$. Analogously, the instantaneous aggregated content of the UL queues at the end of slot *n* is measured through the variable $q_{UL}\ (n)$, as shown in Figure 35

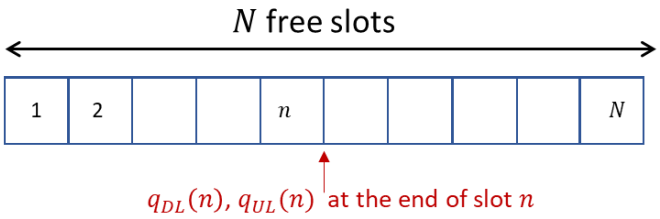

Figure 35. Aggregated queue size measurement

7.2.1 States and Actions

As introduced in previous section, the slot is the minimum temporal unit of granularity considered for scheduling and it will be also used in this section for defining the state in terms of the instantaneous occupancy of the queues of all asynchronous flows with QoS and defining the decisions or actions at the beginning of each superframe.

State

The state at the end of the superframe is defined in order to measure the quantified imbalance between the size of the DL queues and that of the UL queues. This figure is averaged to obtain a greater representativeness of the queue imbalance of the entire most recent superframe t.

$$\bar{\mu}_t = \frac{1}{N}\sum_{n=1}^{N}\mu(n)\,; \qquad \text{where} \qquad \mu(n) = \frac{q_{DL}(n)}{q_{DL}(n) + q_{UL}(n)}$$

Then, the state at the end of superframe t^{th} is defined as a quantified measure of the averaged imbalance as

$$S_t = \begin{cases} s_1 & \bar{\mu}_t < \frac{1}{N_s} \\ s_2 & \frac{1}{N_s} < \bar{\mu}_t < \frac{2}{N_s} \\ & \vdots \\ & \vdots \\ s_{N_s} & \frac{N_s-1}{N_s} < \bar{\mu}_t \end{cases}$$

where N_s is the number of states, which is a parameter to be determined. Note that if the status S_t equals s_1, it means that the DL queue has been almost empty along the superframe t^{th} when compared with the UL queue size. The opposite happens when the status is equal to s_{N_s}

Action

The action set is defined as

$$\mathcal{A} = \{a_1, \dots, a_{N_a}\};$$

with N_a as the size of the action set. The action is defined as the number of free slots devoted to DL. A mapping from the action set to the number of slots devoted to UP and to DL in the radioframe t^{th} is given below

$$\text{If } A_t = a_i \quad \Rightarrow \; N_{DL}(t) = i.\Delta_{slots}; \quad N_{UL}(t) = N - N_{DL}(t)$$

Where Δ_{slots} is an integer parameter that determines the action set size as

$$N_a = \frac{N}{\Delta_{slots}}$$

Figure 36 shows an example where 11 patterns are used providing 11 different actions. Action $a = 0$ means that all the free slots are assigned for UL, $a = 1$ means that a set of Δ_s slots is assigned for DL while $N - \Delta_{slots}$ are assigned for UL, and so on.

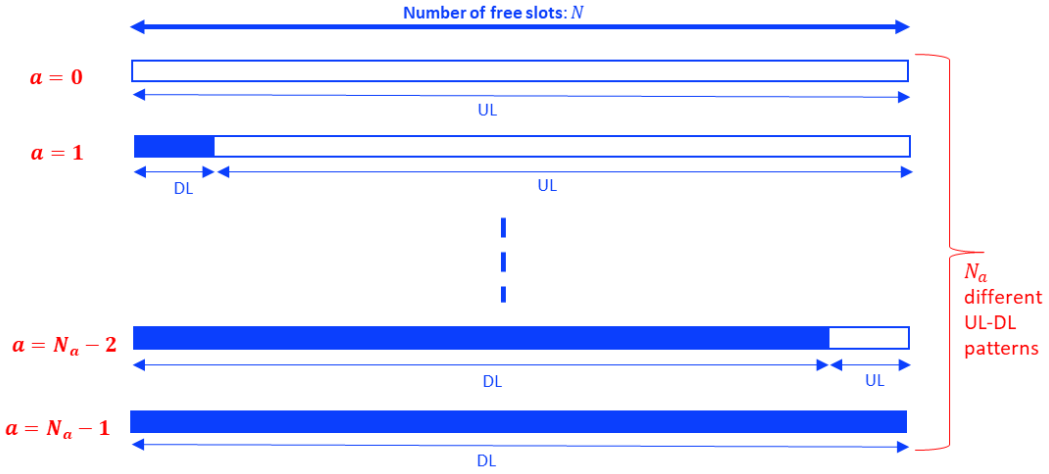

Figure 36. Example of action patterns.

Reward

In section 7.2.2 , an RL based strategy is presented to decide best action A_t at the end of superframe t^{th} and given the state S_t. The goal will be to minimize the imbalance between DL and UL queue as an implicit manner of reducing the latency of asynchronous traffic.

7.2.2 RL based DL/UL splitter

An RL Agent in a given state selects an action for the environment. The state changes after the environment accepts the action. Meanwhile, reward feedback is generated to the Agent. The Agent selects the next action according to the expected future reward and the current state of the environment. Accordingly, the goal of an RL agent is to learn an optimal policy $\pi^*: \mathcal{S} \rightarrow \mathcal{A}$, which determines an action $a \in \mathcal{A}$ under state $s \in \mathcal{S}$, thus, to optimally maximize or minimize a pre-defined value function V^{π}. The value function is typically expressed in terms of the expectation of the return or discounted cumulative reward G_t as

$$V^{\pi}(s) = \mathbb{E}_{\pi}[G_t|S_t = s] = \mathbb{E}_{\pi}[r_t + \gamma V^{\pi}(S_{t+1})|S_t = s]$$

where

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

$\mathbb{E}[\cdot]$ stands for the expectation or statistical mean, r_t is the immediate reward and $\gamma \in [0,1]$ is the discount factor. To better determine the optimal policy, the action-state value function defined as

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[G_t|S_t = s, A_t = a] = \mathbb{E}_{\pi}[r_{t+1} + \gamma V^{\pi}(S_{t+1})|S_t = s, A_t = a]$$

denotes the expected long-term return when at state $S_t = s$ the RL agent performs action $A_t = a$ following the policy π. The goal of training an RL agent is to find an optimal strategy, that is, the policy that gets the most return. All the optimal policies share the same optimal action-value function, denoted Q^*, and defined as

$$Q^*(s,a) \triangleq \max_{\pi} Q^{\pi}(s,a)$$

The Q-Learning (QL) algorithm is one of the most effective model free approaches to rapidly learn an optimal policy π^* by estimating the function $Q^*(s,a)$ iteratively, according to the Bellman equation-based iteration

$$Q(S_t, A_t) = (1-\alpha)Q(S_t, A_t) + \alpha\left(r_{t+1} + \gamma \max_a Q(S_{t+1}, a)\right)$$

Where α is the learning rate which affects the learning speed of $Q(s,a)$. The convergence of the previous iteration is guaranteed by the fixed-point theorem.

In our application we define the immediate reward as the benefit regarding imbalance and latency measured at the beginning of superframe t^{th} . So, once measured queues and lost packets in the recently past superframe, Immediate reward is defined as

$$r_{t+1} = -\left|\bar{\mu}_t - m_{\mu}\right| - L_t$$

Where parameter m_{μ} is fixed to 0.5 to denote a balanced buffered traffic distribution, but a different value can be used if different priorities are defined between UL and DL traffic. L_t stands for the total number of lost packets while waiting for in the queues. In case buffer queues are full and still packets arrive, the oldest packet is lost.

The resultant RL agent is trained, i.e., the target policy function $Q(s,a)$ is learned, following a behavior ϵ-greedy policy to trade-off action exploration versus exploitation. Thus, at each step

(radioframe time), a random number is drawn from a uniform distribution $u \in \mathcal{U}(0,1)$, and it is compared against the pre-defined exploration probability ϵ. If $u \leq \epsilon$ is satisfied, a random action is selected (exploration); otherwise, a greedy action according to $Q(s,a)$ is adopted (exploitation).

During training, the action-value function entries $Q(s,a)$ are iteratively updated to reflect the learning experiences as follows, where each episode lasts N_{SF} superframe periods and each superframe is a step.

RL Splitter:

1: Initialize parameters: $N_a, N_s, \Delta_{slots}, \alpha, \epsilon, m_\mu, N_{SF}$

2: Initialize Action-State Value function $Q(s_i, a_j); i = 1, \dots, N_s; j = 1, \dots, N_a$

3: Compute initial state S_0

4: for t=0: N_{SF}

5: Generate a random number $u \in \mathcal{U}(0,1)$

6: if $u < \epsilon$, take a random action $a \in \mathcal{A}$

7: else $a = \arg\max_a Q(S_t, a)$

8: At the end of superframe t^{th} compute reward r_{t+1} and S_{t+1}

9: Update $Q(S_t, a)$

10: end

Once the training has been completed, the RL agent is ready for exploitation following a greedy and deterministic policy $\pi(s)$, from the learned table $Q(s,a)$ as

$$\pi(s) \triangleq \max_a Q(s,a)$$

7.2.3 Preliminary Performance Evaluation

The RL Splitter has been trained with non-stationary synthetic traffic in order to face the RL agent with the complete set of pairs state-action, enabling the learning process of table $Q(s,a), \forall\, s \in \mathcal{S}, a \in \mathcal{A}$. A simulation has been carried out in which 500,000 superframes of incoming traffic have been generated following the model given in Table 17. 80,000 superframes are dedicated to training, initially with an exploration level of 10% and starting with superframe 40,000 with an exploration level of 1%. For the last 20,000 superframes, training was stopped and only exploitation was carried out using the Q-function learned during training.

Superframe Structure	Superframe duration	10 ms
	Slot duration	1 ms
UL&DL Aggregated traffic	Distribution	Poisson
	Average Poisson Parameter	15 msec.
	Packet size	1500 bytes
	Aggregated Buffer Size at UL	1000 Packets
	Aggregated Buffer Size at DL	1000 Packets
	Average Transmission Rate at UL	480 Mbps
	Average Transmission Rate at DL	480 Mbps
	Maximum packet waiting time	100 msec.
RL Agent	Number of states	$\underline{N_s}$=11
	Number of actions	$\underline{N_a}$=11
	Parameter of exploration	ϵ =0,1&0.01 (changed in the middle of the training period)
	Learning parameter	α=0.1
	Forgetting factor	γ=0.9

Table 17. Simulation parameters to test the DL/UL Splitter

In Figure 37, the aggregated input and output traffic (UL&DL) is depicted with 500,000 superframes. The input traffic exhibits variable rates, characterized by alternating peaks and valleys in packet entries, switching between UL and DL, respectively. Outbound traffic follows the same pattern, but with greater variation depending on the exploration parameter ϵ. In the final part of the execution, pure exploration is performed and it is observed that the outgoing traffic follows the same pattern as the incoming traffic.

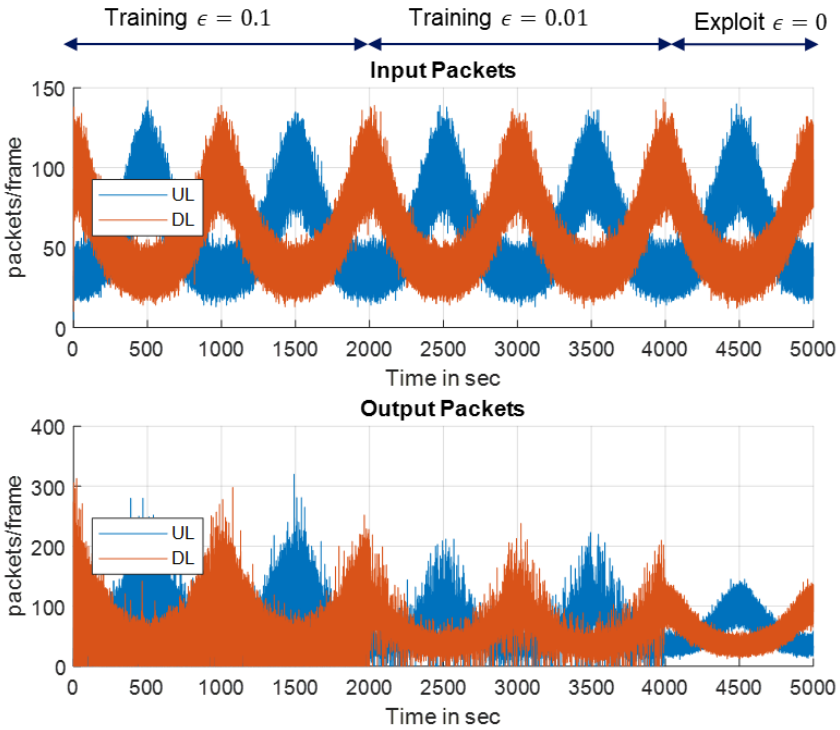

Figure 37. Input and output asynchronous aggregated traffic at the WiFi node measured in arrival and outgoing packets/superframe.

In Figure 38, the evolution of the aggregated queue size and the evolution of the average waiting time in queue is depicted for both, DL and UL.

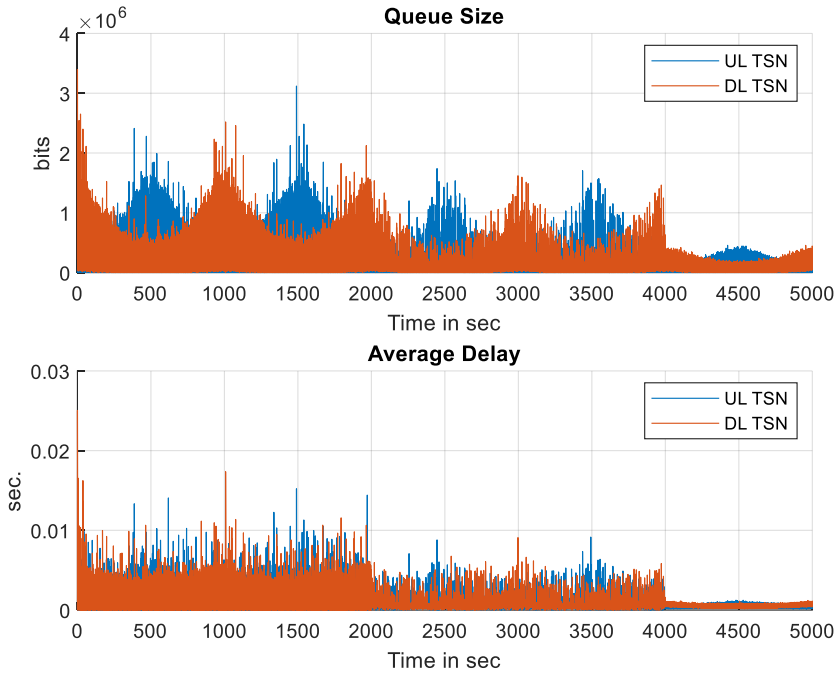

Figure 38. Evolution of the aggregated queue size and the average delay (waiting time in queue) at UL and DL.

The learned Q-function in terms of state and action is shown in Figure 39. Note that in highest states, the actions for which the function is maximum, assign more slots to DL than to UL, while in lowest states the opposite happens.

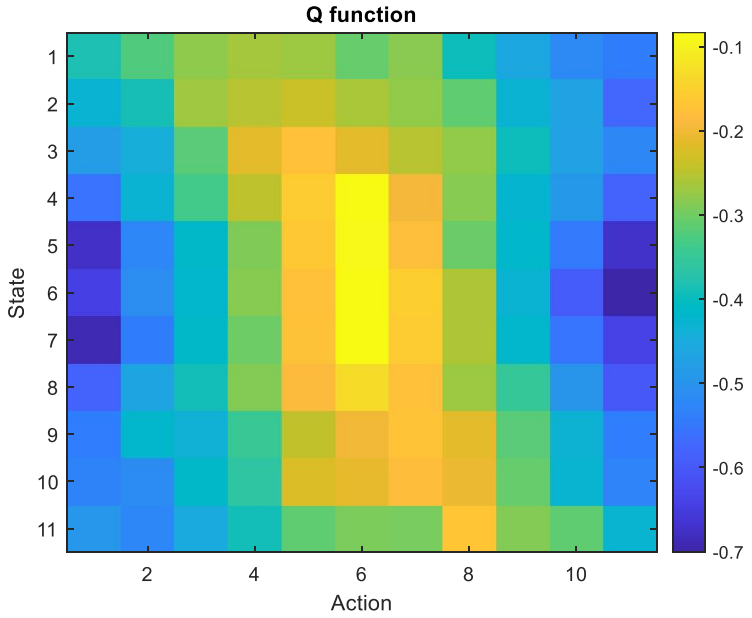

Figure 39. Learned Q function

A baseline strategy has also been designed which consists on assigning to DL a number of slots proportional to the size of the aggregated DL and UL queues. To do this, the number of states is

set to be equal to the number of actions $N_s = N_a$, which implies that at the beginning of the superframe $(t+1)^{th}$ the number of slots assigned to DL is

$$N_{DL}(t+1) = \frac{s_t}{N_s} N$$

This strategy is proposed mainly as a baseline with the purpose of comparing the performance of both methods, the RL splitter and the Baseline splitter. In this regard, Figure 40 depicts the cumulative average delay for the simulation presented in previous figures. The average delay is measured in the exploitation period (100,000 superframes) using the RL Splitter.

The waiting time per package is measured as the difference between its arrival time-stamp at the queue and the time-stamp when it is served and consequently removed from the corresponding queue. These measurements are averaged over all packets transmitted in a superframe. Finally, the cumulative distribution function of all the averaged delays along the exploitation period is calculated. The Baseline splitter is also applied in the same period of generated traffic and the cumulative density function (cdf) of the average delay is computed.

As Figure 40 shows, percentiles up to 95% showcase an average delay below 0.8 ms when the RL splitter is utilized, indicating that the majority of packets experience low latency, even in scenarios with increased network load or congestion. In contrast, when the baseline method is applied, percentiles up to 95% degrade noticeably below 1.8 ms.

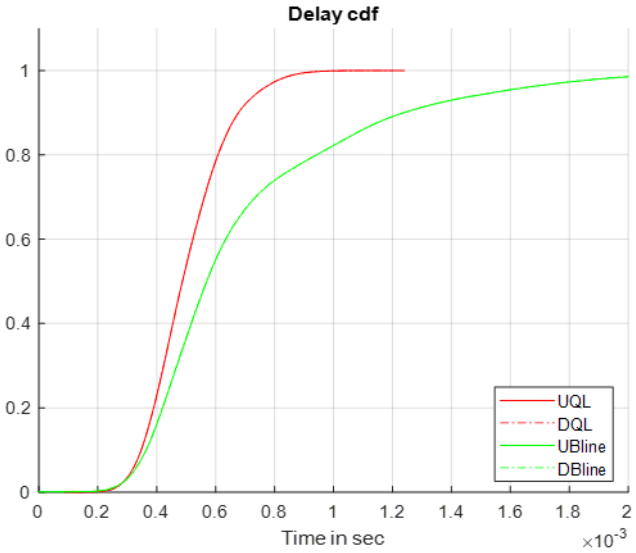

Figure 40. Average delay cdf measured during the exploitation period (100,000 superframes). The delay is the packet waiting time and it is averaged using all the packets transmitted in each superframe.

7.3 Selected Algorithms for the Wireless Flow Scheduler (WFS)

7.3.1 TSN Windows Design for Isochronous Traffic

To maximize compatibility, the isolated windows design of Section 7.1 is adopted for evaluating the WFS performance. We consider that we have N_{DL} active isochronous flows in the downlink and N_{UL} in the uplink. All the active flows have the same period, that coincides with the superframe duration $T_{\text{superframe}}$. Also, we consider that all packets have the same length (B

bits). In order to minimize packet loss, all windows are designed for the lowest MCS (MCS=0). Using the definitions in Section 7.1, the probability of packet loss is $\check{p} = q = \Pr(R_n^{WiFi} = 0)$ and the window length $WL_n^{WiFi} = \lceil B/r_0 \rceil$ for the downlink and, $WL_n^{WiFi} = T_{p,UL} + \lceil B/r_0 \rceil$ for the uplink.

Accordingly, the number of slots reserved for the DL and UL flows is $N_{DL}\lceil B/r_0 \rceil \leq T_{DL}$ and $N_{UL}\left(T_{p,UL} + \lceil B/r_0 \rceil\right) \leq T_{UL}$. The following superframe design is considered in the following:

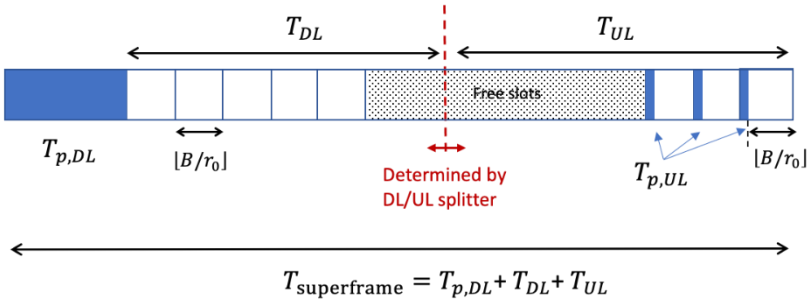

Figure 41. Superframe design for $N_{DL} = 5$ and $N_{UL} = 3$. Overhead is plotted in blue. $T_{p,DL}$ is the downlink overhead (in slots) and includes CTS + IFS + Beacon + DL preamble [SEI18]

As shown in Figure 41, the N_{UL} slots are placed at the end of the superframe to have all the free slots in the middle of the superframe and giving so maximum flexibility to the UL/DL splitter for distributing these free slots between DL and UL. Note that, because of the conservative design of the windows, additional free slots will be available within the limits of the windows in case some isochronous flows are transmitted using faster MCS modes (i.e., $R_n^{WiFi} > r_0$).

After allocating all the isochronous flows within their windows, the number of available slots is

$$T_{DL}^{free} = T_{DL} - \sum_{n=1}^{N_{DL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil$$

$$T_{UL}^{free} = T_{UL} - N_{UL} T_{p,UL} - \sum_{n=1}^{N_{UL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil$$

where T_{DL} and, indirectly, $T_{UL} = T_{superframe} - T_{DL} - T_{p,DL}$ are adjusted by the UL/DL splitter described in Section 7.2. Note that T_{DL}^{free} and T_{UL}^{free} change from superframe to superframe due to the randomness of R_n^{WiFi}. At the beginning of every superframe, the dynamic schedulers proposed in sections 7.3.3 and 7.3.4 assign the available free slots among the lower-priority flows considered in the project: asynchronous TSN flows (ASYN) and best-effort flows (BE).

7.3.2 UL/DL Splitter for Asynchronous Traffic

The Baseline UL/DL Splitter introduced in previous section is activated in Test 4 to redistribute free slots between UL and DL. In next deliverables the RL based splitter will be also included.

Let us define T^{free} as the number of slots that fulfills

$$T^{free} = T_{DL} + T_{UL} - N_{DL} \sum_{n=1}^{N_{DL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil - N_{UL} T_{p,UL} - \sum_{n=1}^{N_{UL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil$$

and must be distributed between UL and DL to allocate ASYNC flows. When the splitter is activated, it computes T_{UL}^{free} and T_{DL}^{free} such that

$$T_{UL}^{free} + T_{DL}^{free} = T^{free}$$

The baseline splitter assigns slots proportionally to the traffic volume of the queues of all flows in each of the two cases, UL and DL.

7.3.3 Asynchronous Traffic DL Scheduling

In every superframe, the available T_{DL}^{free} slots are assigned among the asynchronous TSN traffic flows using the LWDF (Largest-Weighted-Delay-First) scheduler [RAM01] presented in D1.2 (SP1). As indicated previously, we consider a single band (J=1).

The LWDF scheduler prioritizes flows according to the following utility function:

$$U_n = R_n^{WiFi} Q_n$$

where Q_n stands for the state of the nth flow buffer (queue) at the beginning of the superframe (number of bits in the queue) and R_n^{WiFi} is the instantaneous rate of flow n during this superframe. The flow maximizing the utility U_n is allowed to transmit first. It will transmit using as many slots as required to empty its buffer or until it has consumed all the available slots (T_{DL}^{free}). If some slots are still free, the second flow with the highest utility is ordered to transmit following the same procedure, and so on. In order to minimize latency, the available slots are assigned sequentially from the start to the end of the DL real-time subframe.

Finally, if there are still free slots after allocating all the ASYN traffic, BE flows without QoS are transmitted using a round-robin (RR) scheduler following an arbitrary order. If the last BE flow allocated in the previous superframe was the flow with index k, the first BE flow transmitting in the current superframe (if this is feasible) will be the one with index k+1. Although other approaches are possible, we consider that scheduled flows transmit until emptying their queues or exhausting all the available slots (T_{DL}^{free}).

7.3.4 Asynchronous Traffic UL Scheduling

The same scheduling algorithms proposed for the downlink are adopted for the uplink with the following adjustments:

- Assuming a single uplink flow per station, every ASYN and BE uplink flow has to include an interframe spacing and short preamble of duration $T_{p,UL}$. Note that this overhead could be avoided if the same station was generating several simultaneous uplink flows.
- All stations having active uplink flows have to report their buffer size Q_n at the beginning of the superframe so that the AP is able to implement the LWDF scheduler [RAM01]. As explained in subsection 7.3.3, this scheduler prioritizes flows according to metric $U_n = R_n^{WiFi} Q_n$ with the aim of minimizing the average latency. We assume that these buffer status reports (BSR) are sent periodically to the AP. Preliminary simulations will be carried out assuming that updated reports are available at the beginning of every superframe.

7.3.5 Preliminary Performance Evaluation

A simple but representative scenario is defined to test the performance of the wireless flow scheduler. For simplicity, we consider that flows of the same class are identical and stations are connected to the AP by means of independent and identically distributed propagation channels. Although it does not affect the results, we consider that most stations have a bidirectional link with the AP, that is, they have a pair of active flows: a DL flow and an UL flow. As introduced in D1.2, we have also used the PHY layer abstraction included in Matlab [MAT21] that provides tables to evaluate the effective SINR in case of frequency-selective channels as well as PER tables for the standardized forward-error correcting codes. The main simulation parameters are specified in Table 18.

Regarding the Wi-Fi physical layer specification, in compliance with the IKL SHARP platform, we have considered Table 19, which is a reduced MCS table based on the 802.11a/g standard (non-HT format) [WLA99]. The required SINR for achieving PER≤10^{-4} with the binary convolutional code of the standard (codeword length 32 bytes) is detailed in the last column of Table 19.

The performance evaluation tests are defined in Table 20.

Model	Parameter	Value
Pathloss (slow fading)	Average Received SNR	25.3 dB (fixing $q \approx 10^{-4}$)
	Time variability	Constant
Multipath (fast fading)	Delay Spread	50ns *29-89 ns according to [SEI18]*
	Power Delay Profile	Exponential
	Statistics	Rayleigh
	Coherence Time	30ms *10-30ms according to [SEI18]*
	Doppler Spectrum	Block fading
Superframe Structure	Superframe duration ($T_{\text{superframe}}$)	10 ms = 2500 slots
	Slot duration (τ_{slot})	1 OFDM symbol
	OFDM Symbol Duration	4us (3.2us+0.8us)
	Bandwidth	20MHz
	Number of data subcarriers	48
	MCS	0, ..., 6
	MCS performance	802.11a/g table (Table 19)
	Downlink overhead ($T_{p,DL}$) CTS + IFS + Beacon + DL preamble	11 + 2.5 + 5 + 5 = 23.5 slots ≈ 24 slots
	Interframe spacing (IFS) between UL bursts ($T_{IFS,UL}$)	0.5 ≈ 1 slot
	Uplink overhead ($T_{p,UL}$) IFS + UL preamble	0.5 + 1 slot = 1.5 slots ≈ 2 slots
	Retransmissions	NO
ISO Traffic	Periodicity	10 ms
	Packet length (B)	30 bytes = 240 bits
	Window length ($\lceil B/r_0 \rceil$)	240/(48*0.5) = 10 slots

ASYN Traffic 1 (Easybot 1)	Periodicity	5ms
	Inter-packet variability	0
	Packet length	90 bytes = 720 bits
	Average number of bits per superframe (C)	1440 bits
ASYN Traffic 2 (D435 Camera)	Periodicity	0.2ms
	Inter-packet variability	0
	Packet length	300-1440bytes 2400-11520 bits (uniform distrib)
	Average number of bits per superframe (C)	348 kbits

Table 18. Simulation scenario

MCS	MOD	COD	Rate r_i (bits/slot)	Rate (Mbps)	Req SINR (dB)
0	BPSK	1/2	24	6	2.0
1	BPSK	3/4	36	9	5.0
2	QPSK	1/2	48	12	4.7
3	QPSK	3/4	72	18	7.4
4	16-QAM	1/2	96	24	10.7
5	16-QAM	3/4	144	36	14.0
6	64-QAM	2/3	192	48	18.2
7	64-QAM	3/4	216	54	19.5

Table 19. MCS table for 802.11a/g [WLA99]. MCS1 is not used because it offers worst performance than MCS2 and it was removed from the table in subsequent revisions of the standard.

Test number	Deliverable	Description
Test 1	D1.3 (SP1)	Test that validates the correct operation of the WFS and its interfaces with the WiFi node.
Test 2	D1.1 (SP2)	Upgrade of Test 1 including a preliminary version of the UL/DL splitter
Test 3	D1.4 (SP1)	Test that evaluates the performance of the WFS in terms of the admitted ASYN traffic throughput and delay for a given load of ISO traffic.
Test 4	D1.4 (SP1)	Test that evaluates performance improvement of the ASYN traffic delay when the Baseline UL/DL Splitter is activated to redistribute free slots between UL and DL

Table 20. Definition of performance evaluation tests

7.3.6 Test 3 description

Next, we describe how Test 3 is implemented in the WFS simulator and what results are obtained:

- Deactivate the UL/DL splitter and fix T_{DL} and T_{UL} manually to approximately the same value. For example, choose $T_{DL} = 1240$ and $T_{UL} = 1236$ holding that $T_{DL} + T_{UL} = T_{\text{superframe}} - T_{p,DL} = 2476$.
- Fix the number of ISO flows in the downlink (N_{DL}) so that their windows occupy approximately 50%, 75% and 100% of the time assigned to the downlink (T_{DL}), i.e.,

$$\frac{N_{DL}}{T_{DL}}\left\lceil\frac{B}{r_0}\right\rceil = N_{DL}\frac{10}{1240} \approx \gamma$$

with γ equal to 0.5, 0.75 or 1. Proceed in the same way with the uplink:

$$\frac{N_{UL}}{T_{UL}}\left(T_{p,UL}+\left\lceil\frac{B}{r_0}\right\rceil\right)=N_{DL}\frac{12}{1236}\approx\gamma$$

Solve the above equations to obtain the number of active ISO flows for every configuration. The result is shown in Table 21.

γ	N_{DL}	N_{UL}
0.5	62	52
0.75	93	77
1	124	103

Table 21. Number of active ISO flows

- In every superframe, evaluate the number of free slots, which is random because it depends on the random rate R_n^{WiFi}:

$$T_{DL}^{free}=T_{DL}-\sum_{n=1}^{N_{DL}}\left\lceil\frac{B}{R_n^{WiFi}}\right\rceil=1240-\sum_{n=1}^{N_{DL}}\left\lceil\frac{240}{R_n^{WiFi}}\right\rceil$$

$$T_{UL}^{free}=T_{UL}-N_{UL}T_{p,UL}-\sum_{n=1}^{N_{UL}}\left\lceil\frac{B}{R_n^{WiFi}}\right\rceil=1236-N_{UL}2-\sum_{n=1}^{N_{UL}}\left\lceil\frac{240}{R_n^{WiFi}}\right\rceil$$

with $\Pr\left(R_n^{WiFi}=r_i\right)=p_i$ for $i=0,\ldots,i_{max}=7$. Consider that the window of the nth flow is fully available for asynchronous flows if $R_n^{WiFi}=0$ (packet loss), which happens with probability q. For the simulated scenario in Table 18, Table 22 shows the approximated probability associated to each MCS of Table 19:

q	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7
10^{-4}	$2.7\cdot10^{-4}$	x	$6.4\cdot10^{-3}$	$3.8\cdot10^{-4}$	$1.6\cdot10^{-1}$	$8.7\cdot10^{-2}$	$1.8\cdot10^{-1}$	$5.7\cdot10^{-1}$

Table 22. Probability associated to each MCS

Using these probabilities, the average number of free slots can be approximately computed. The result is shown in Table 23.

γ	$E\{T_{DL}^{free}\}$	$E\{T_{UL}^{free}\}$
0.5	1104.9	1018.7
0.75	1037.4	914.2
1	969.8	805.6

Table 23. Average number of free slots

using that $E\left\{\left\lceil\frac{240}{R_n^{WiFi}}\right\rceil\right\}\approx 2.1786$ slots.

- In every superframe, schedule an increasing number of N_{DL}^{ASYN} asynchronous users using the available slots (T_{DL}^{free}) using the LWDF scheduler described in Section 7.3.3. Consider asynchronous traffic 1 (Easybot 1) from Table 18. For every value of N_{DL}^{ASYN}, simulate a large number of superframes (N_{sim}) and compute the statistics of the packet delay (average delay, outage delay, jitter, etc.). Consider that that flows are endowed with queues sufficiently large to neglect overflow. Compute also the throughput

TH_{DL}^{ASYN} (bps) defined as the number of bits of the N_{DL}^{ASYN} flows that were downloaded during the N_{sim} simulated superframes divided by $N_{sim} \cdot T_{\text{superframe}} \cdot \tau_{\text{slot}}$.

- Do the same for the uplink bearing in mind that you have to append $T_{p,UL} = 2$ slots to every scheduled flow.
- In order to guarantee the stability of the queues of all the asynchronous flows, we have to check that, on average, the number of bits entering the queue during one superframe is lower than the number of bits that can be served in this superframe. Formally, we have that:

$$N_{DL}^{ASYN} 2B < E\{R_n^{WiFi}\} E\{T_{DL}^{free}\} \lesssim 216 \cdot E\{T_{DL}^{free}\}$$

which provides the maximum number of admitted asynchronous flows in the DL. The second inequality takes into account that, when approaching saturation, the LWDF scheduler selects for transmission a reduced number of users $N_{UL,served}^{ASYN}$ having the highest data rates and, therefore, $E\{R_n^{WiFi}\}$ is shifted towards $\max\{R_n^{WiFi}\} = 216$ (Table 19). In the literature, this effect is generally called *multiuser gain*. In the proposed test (high SNR), the approximation above is accurate. Note that we have taken into account that asynchronous flows convey 2 packets of length B per superframe (Table 18).

Proceeding in the same way in the UL, we have that

$$N_{UL}^{ASYN} 2B < E\{R_n^{WiFi}\} E\{T_{UL}^{free}\} - E\{R_n^{WiFi} N_{UL,served}^{ASYN} T_{p,UL}\} \lesssim 216 \cdot E\{T_{UL}^{free}\}$$

where $N_{UL,served}^{ASYN}$ is again the number of served flows in every superframe, which is random and decreases as the system approaches saturation. We consider that, near saturation, few flows are served in every superframe and the UL overhead term can be neglected.

Finally, based on the above approximations, the maximum number of admitted asynchronous flows is given by:

γ	N_{DL}^{ASYN} (max)	N_{UL}^{ASYN} (max)
0.5	165	152
0.75	155	137
1	145	120

Table 24. Maximum number of admitted asynchronous flows

The values listed in Table 24 are only provided as a reference. The number of accepted users N_{DL}^{ASYN} will be significantly lower in order to work with tolerable delays.

- Plot average and outage delay as a function of the throughput TH_{DL}^{ASYN} for the three values of γ under test. Plot also the delay jitter defined as the standard deviation of the delay.
- Compare results to the following benchmark:
  - Round Robin scheduler
  - Only free slots outside the ISO windows are available for serving ASYN traffic. The number of free slots in this case is smaller and constant in time (see table below). Note that no room for ASYN traffic is left in case of $\gamma = 1$.

γ	$T_{DL}^{free} = 1240 - N_{DL}10$	$T_{UL}^{free} = 1236 - N_{UL}12$
0.5	620	612
0.75	310	312
1	0	0

Table 25. Number of free slots

In the adopted benchmark, the maximum number of users that can be admitted in order to guarantee stability is severely reduced. In particular, we obtain that

$$N_{DL}^{ASYN} < \frac{E\{R_n^{WiFi}\}}{2B} T_{DL}^{free} = \frac{185.4}{1440} \cdot T_{DL}^{free}$$

$$N_{UL}^{ASYN} \lesssim \frac{E\{R_n^{WiFi}\}}{2B} T_{UL}^{free} \lesssim \frac{185.4}{1440} \cdot T_{UL}^{free}$$

using that, in this case, there is no multiuser gain and the average transmission rate is $E\{R_n^{WiFi}\} = \sum_{i=0}^{i_{\max}} p_i r_i = 185.4$. Again, for simplicity, we neglect the UL overhead when the system works close to saturation.

γ	N_{DL}^{ASYN} (max)	N_{UL}^{ASYN} (max)
0.5	80	79
0.75	40	40
1	0	0

Table 26. Maximum number of users that can be admitted with guaranteed stability

7.3.7 Simulation results (Test 3)

Plots	γ (%)	Scheduler	Description
Figure 42	50	Proposed Dynamic Scheduler	Average delay as a function of DL&UL ASYN throughput
			Outage delay (prob=0.1) as a function of DL&UL ASYN throughput
			Jitter (std) as a function of DL&UL ASYN throughput
Figure 43	75	Proposed Dynamic Scheduler	Average delay as a function of DL&UL ASYN throughput
			Outage delay (prob=0.1) as a function of DL&UL ASYN throughput
			Jitter (std) as a function of DL&UL ASYN throughput
Figure 44	100	Proposed Dynamic Scheduler	Average delay as a function of DL&UL ASYN throughput **(benchmark)**
			Outage delay (prob=0.1) as a function of DL&UL ASYN throughput
			Jitter (std) as a function of DL&UL ASYN throughput
Figure 45	50	Benchmark Scheduler	Average delay as a function of DL&UL ASYN throughput
			Outage delay (prob=0.1) as a function of DL&UL ASYN throughput
			Jitter (std) as a function of DL&UL ASYN throughput
Figure 46	75	Benchmark Scheduler	Average delay as a function of DL&UL ASYN throughput
			Outage delay (prob=0.1) as a function of DL&UL ASYN throughput
			Jitter (std) as a function of DL&UL ASYN throughput

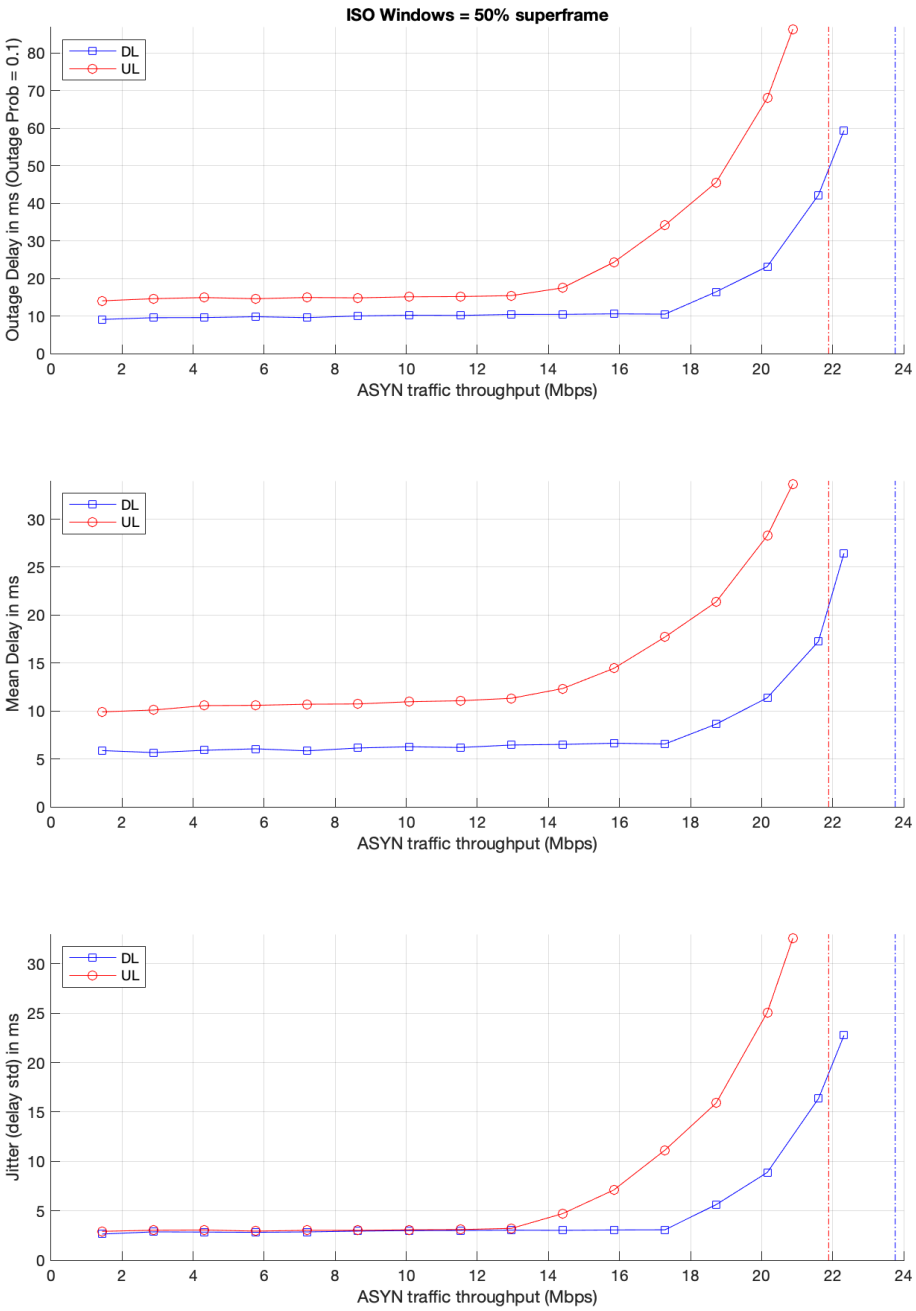

Figure 42. Delay evaluation with the proposed dynamic scheduler. ISO windows occupies 50% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

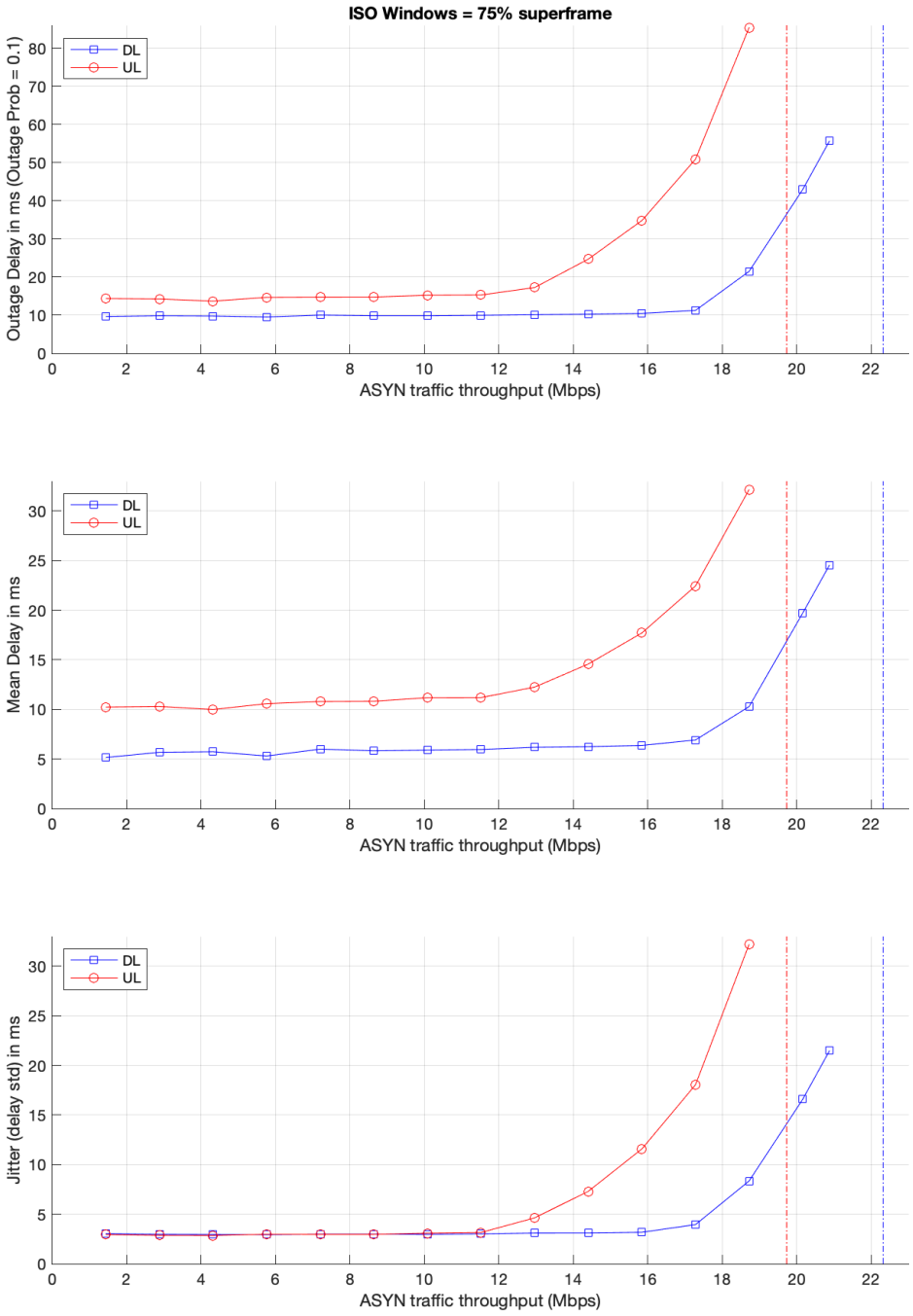

Figure 43. Delay evaluation with the proposed dynamic scheduler. ISO windows occupies 75% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

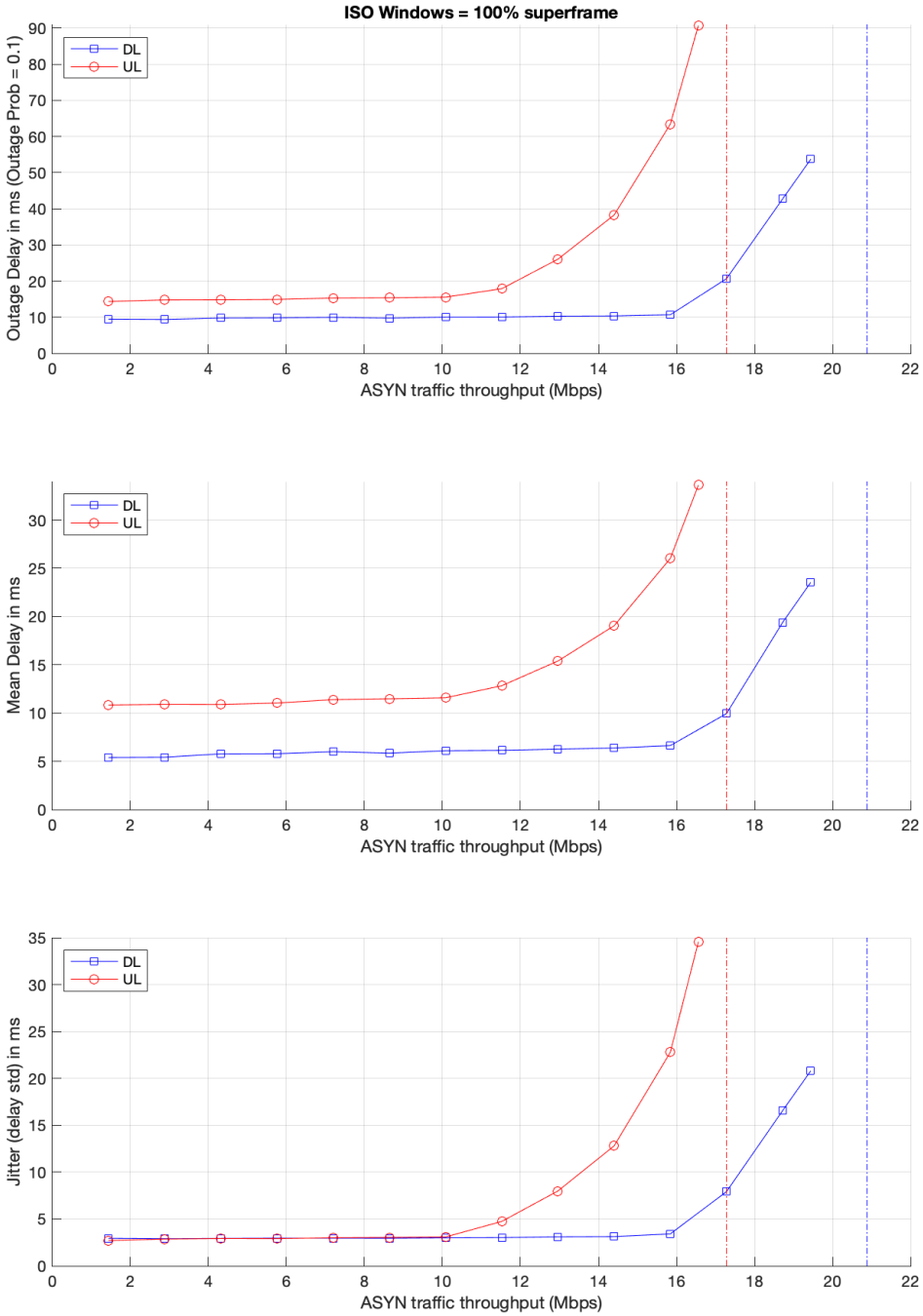

Figure 44. Delay evaluation with the proposed dynamic scheduler. ISO windows occupies 100% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

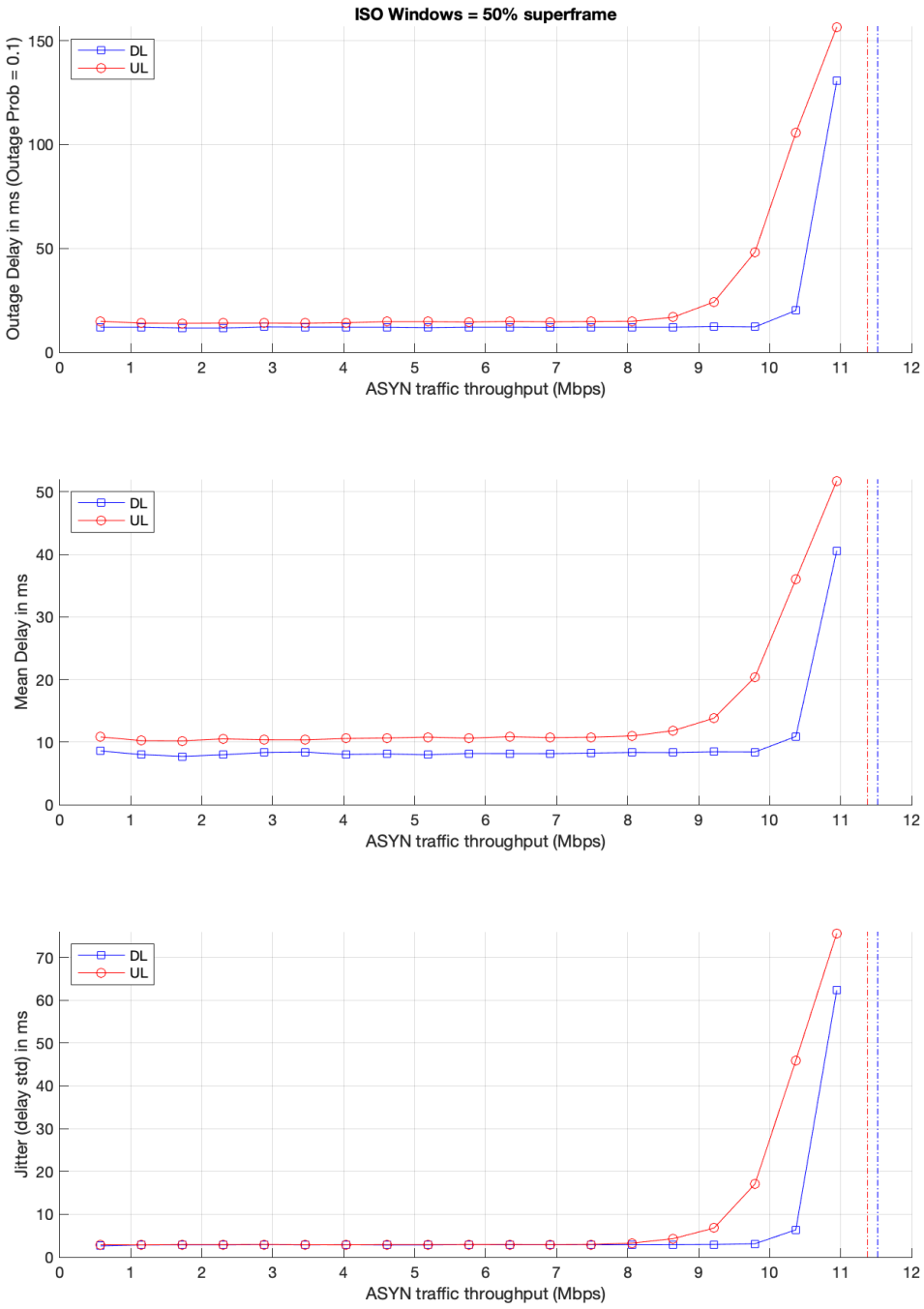

Figure 45. Delay evaluation with the benchmark scheduler. ISO windows occupies 50% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

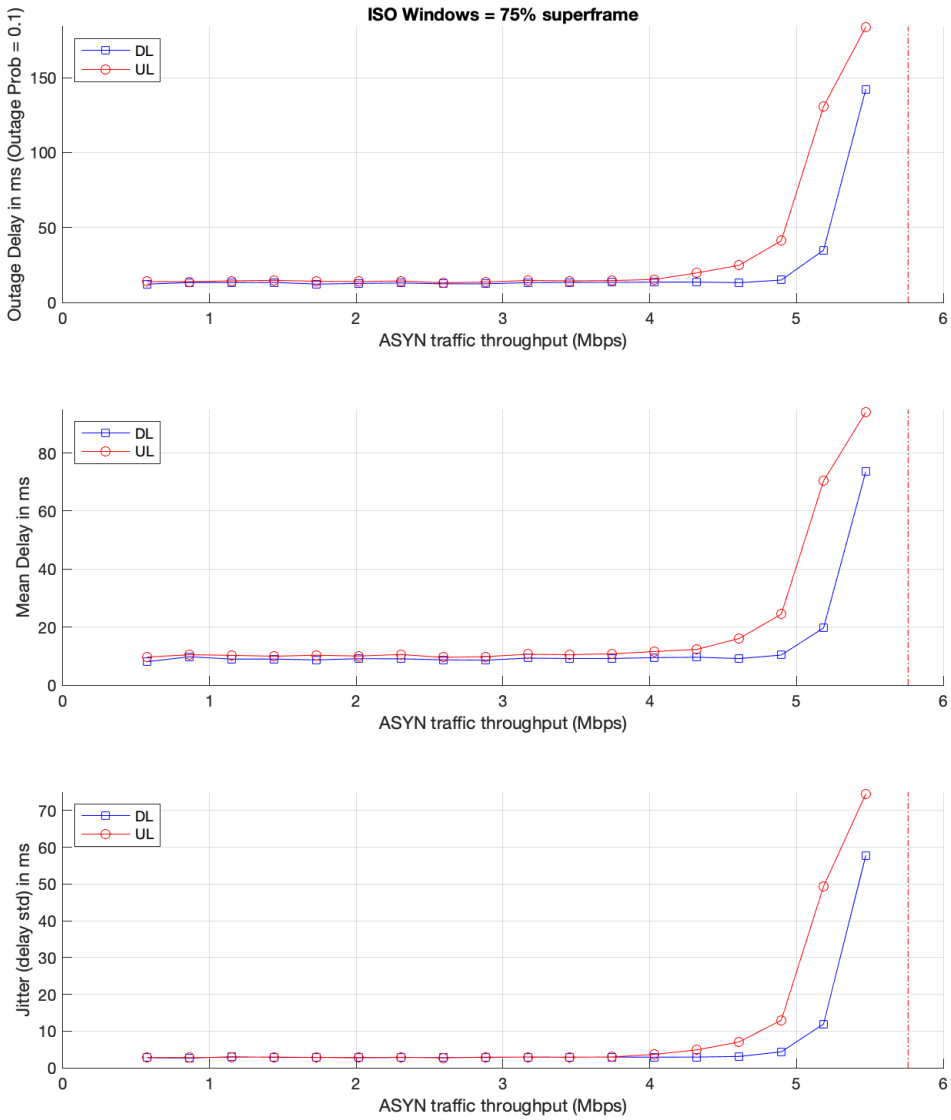

Figure 46. Delay evaluation with the benchmark scheduler. ISO windows occupies 75% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

From the above simulations we can draw the following conclusions:

- By exploiting the empty slots inside the TSN windows and adjusting the MCS according to the instantaneous channel state, the proposed dynamic scheduler can increase notably the admitted asynchronous throughput with respect to the benchmark. Focusing first on the downlink, simulations show that it can be increased from 10Mbps up to 17Mbps (70% gain) when the superframe is moderately loaded with isochronous traffic (50%). This gain increases when the number of isochronous flows increases. When the reserved time for isochronous traffic is 75%, we can increase the asynchronous throughput from 5Mbps until about 17Mbps (340% gain). In particular, we can transmit about 16Mbps even if the ISO windows span over the whole superframe. Similar gains are exhibited in case of the uplink.

- The UL is always degraded with respect to the DL. The reason is that UL transmissions incorporate individual short preambles ($T_{p,UL} = 2$) whereas a common long preamble is used

in the DL ($T_{p,DL} = 24$). Moreover, UL packets have to wait until the start of the UL subframe to be transmitted, that in the simulated scenario it is near the center of the supeframe, inducing a systematic delay of half a superframe (5ms).

7.3.8 Test 4 description

Test 4 is conducted to validate the effectiveness of employing a dynamic UL/DL Splitter. It assesses the improvement in performance of ASYN traffic delay when the Baseline UL/DL Splitter is utilized to redistribute free slots (T^{free}), between UL (T_{UL}^{free}) and DL (T_{DL}^{free}).

In order to validate the Baseline UL/DL splitter, test 4 focuses on the case where ISO windows occupies 50% of the superframe duration. In Test 4, the number of flows in the uplink (N_{UL}^{ASYN}) remains constant at 70 for all measured points, while the number of flows in the downlink (N_{DL}^{ASYN}) ranges from a minimum of 10 to a maximum of 155, as in Test 3. This allows for the assessment of various unbalanced traffic scenarios between UL and DL.

Two experiments are conducted in Test 4. In the first one the Baseline UL/DL splitter is activated while in the second experiment T_{DL}^{free} and T_{UL}^{free} are computed as shown in table, ensuring an approimately balanced allocation of slots between uplink and downlink regardless of the traffic distribution.

The relevant parameters of Test 4 are presented in Table 27.

ISO Window occupancy	γ	0.5
DL and UL Schedulers	LWDF	
DL Asynchronous users	N_{DL}^{ASYN}	Variable [10,…,155]
UL Asynchronous users	N_{UL}^{ASYN}	Fixed at 70
Experiment 1 UL/DL splitter deactivated	T_{DL}^{free}	$T_{DL} - N_{DL} \sum_{n=1}^{N_{DL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil$
	T_{UL}^{free}	$T_{UL} - N_{UL} \sum_{n=1}^{N_{UL}} \left\lceil \frac{B}{R_n^{WiFi}} \right\rceil$
Experiment 2 Dynamic Baseline UL/DL Splitter	T_{DL}^{free}	Assigned dynamically
	T_{UL}^{free}	Assigned dynamically

Table 27. Relevant parameters of Test 4

7.3.9 Simulation results (Test 4)

Plots	γ (%)	Scheduler	Description
Figure 47	50	Proposed Dynamic Scheduler	DL Average delay as a function of DL ASYN throughput
			DL Outage delay (prob=0.1) as a function of DL ASYN throughput
			DL Jitter (std) as a function of DL ASYN throughput
Figure 48	50	Proposed Dynamic Scheduler	UL Average delay as a function of DL ASYN throughput
			UL Outage delay (prob=0.1) as a function of DL ASYN throughput
			UL Jitter (std) as a function of DL ASYN throughput

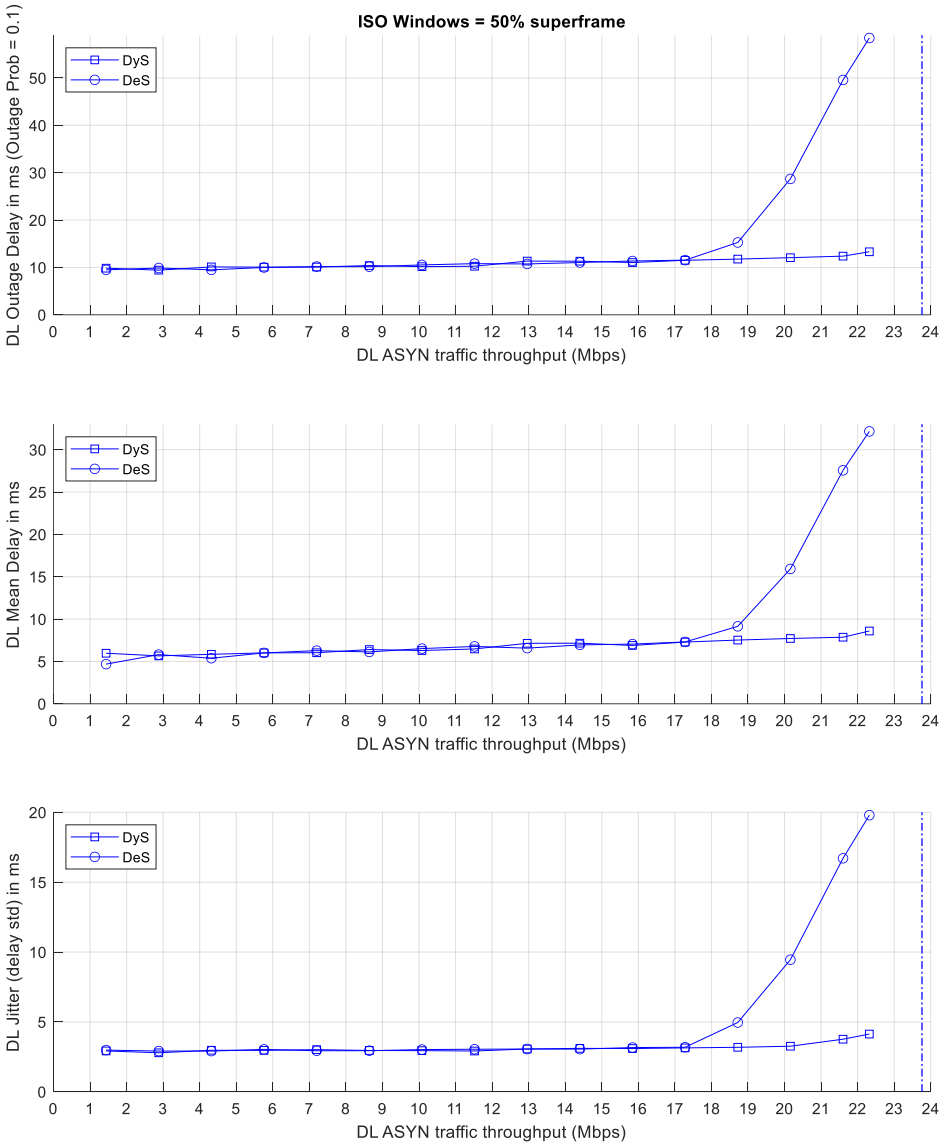

Figure 47. Delay evaluation at the DL with the proposed dynamic scheduler and both splitters: Dynamic (DyS) and Deactivated (DeS). ISO windows occupies 50% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

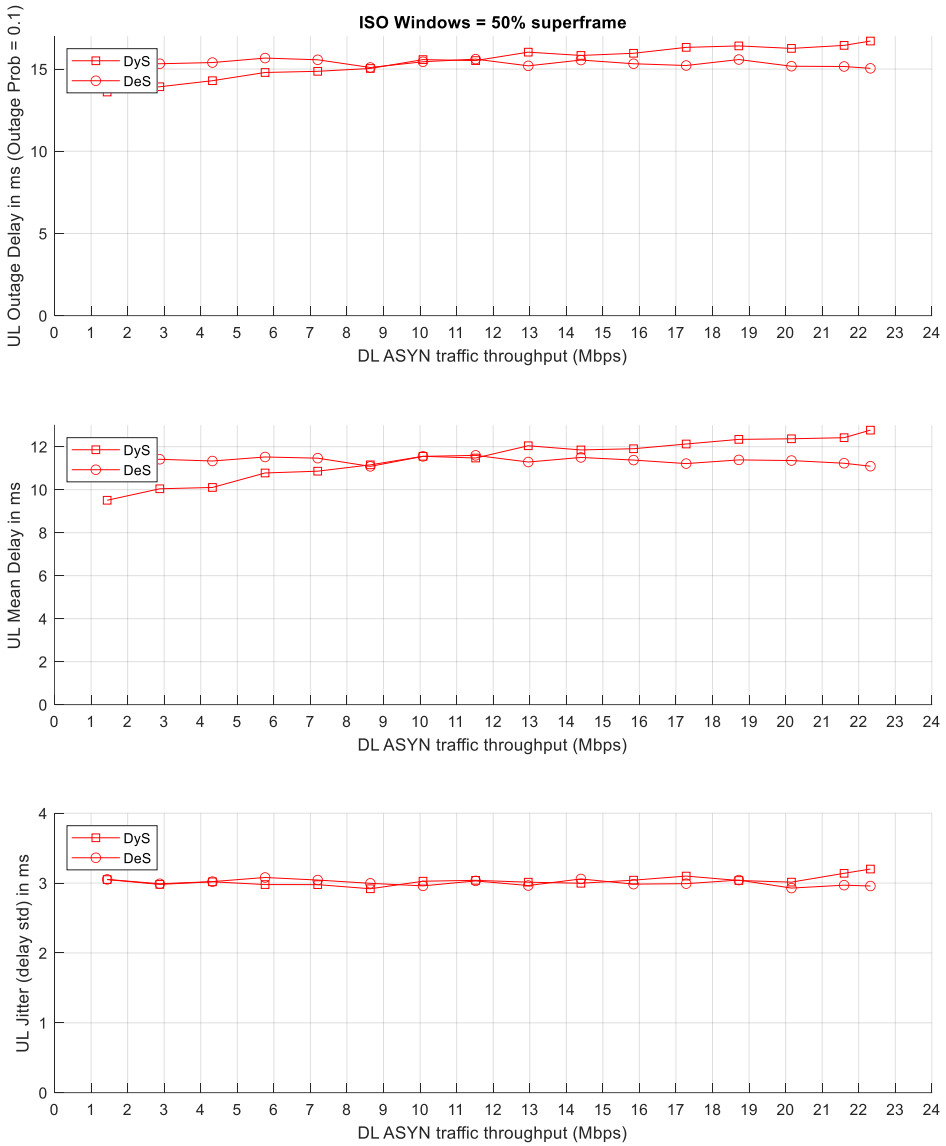

Figure 48. Delay evaluation at the UL with the proposed dynamic scheduler and both splitters: Dynamic (DyS) and Deactivated (DeS). ISO windows occupies 50% of the superframe duration. The saturation throughput is indicated with a dashed vertical line.

From the above simulations we can draw the following conclusions:

- The dynamic baseline splitter effectively reduces latency for dense traffic scenarios. Specifically, Figure 47 corresponds to the admission of ASYN traffic up to 22 Mbps, resulting in the accommodation of a greater number of ASYNC flows compared to scenarios where the splitter is deactivated. On the other hand, in conditions of moderate throughput, as is the case of the UL (10 Mbps), the latency is not degraded by the use of the dynamic splitter as shown in Figure 48.

8 DT AND KPI ESTIMATION

From the previous section, it can be seen that scheduler works by defining a superframe (SF) of fixed length as a set of time slots that repeats over time. Each time slot can be assigned to a single flow so as to guarantee that time sensitive flows (hereafter, referred as TS flows) meet the required performance, defined in terms of KPIs, such as end-to-end delay and jitter. In this way, traffic flows of multiple classes can coexist.

However, the allocation of such time slots needs to be faced from a **network perspective** to ensure that e2e performance. During the execution of this project, we realized about the **need of extending the control plane architecture** and adding a novel component to deal with that network-wide allocation, called *Time Sensitive Flow Scheduler Planner* (**TS-FSP**).

Therefore, this section extends the previous sections in D1.1 and D1.2 regarding DT and KPI estimation, in the following way:

1) Introducing the role of the new TS-FSP component in the provisioning workflow, highlighting its relation with the DT.
2) Presenting the definition of the optimization problem to be solved in the TS-FSP component.
3) Proposing extensions and updates of the DT modules, in particular for traffic generation and queueing models, as well as for the overall procedure.
4) Presenting illustrative results that validate the performance of the DT, comparing its estimations with selected simulation results already presented in Section 7, as well as showing KPI estimation results under the event of two different provisioning requests.

8.1 PROVISIONING WORKFLOW UPDATES

The role of the new TS-FSP module is to plan TS flow time windows across a defined path, producing worst-case scheduling for the TS flows to be deployed in the network. In addition, the DT is in charge of evaluating a set of KPIs of non-TS flows before new (TS or non-TS) flows are deployed. The DT considers non-TS flows with different priorities, e.g., QoS committed and BE. Note that although all flows are served on a particular path defined for each flow, TS flows have specific resources that are reserved along their path, whereas non-TS ones use the remaining resources, which are assigned by their priority.

The extended provisioning workflow is depicted in Figure 49. The algorithm starts when a flow request arrives specifying the characteristics of the flow, including the end-points, class of service (e.g., TS, QoS, and BE) KPI requirements, if any, traffic profile including periodicity in the case of a TS flow, and others (step 1 in Figure 49). The algorithm follows a different procedure for TS and non-TS flows (2). In the case of TS flows (3), the shortest path is computed on a subgraph that includes the end-points of the flow and nodes with TSN capabilities. Next, the TS-FSP module finds a scheduling plan for the new TS flow, and changes in the scheduling of already deployed TS flows, so as to meet the requirements. If no scheduling plan is found, conflicting links and disconnected partitions not including the end-points of the requested flow are removed from the subgraph. If the resulting subgraph is disconnected, no resources are available for the new TS flow request, which is rejected (4). If a scheduling plan is found, the NDT is called to estimate the performance of the non-TS flows already being served as if the TS flow

were setup (5). This is a crucial step, as the new TS flow will be assigned resources to detriment of non-TS flows, which will impact their KPIs. In case the performance of QoS committed flows can be guaranteed, the new request is accepted (6). Otherwise, a procedure that excludes the conflicting link, similar to the one introduced above is followed (7) until a solution is found or the request is finally rejected (8). Note that non-TS flows provisioning follow a similar procedure except for the scheduling plan.

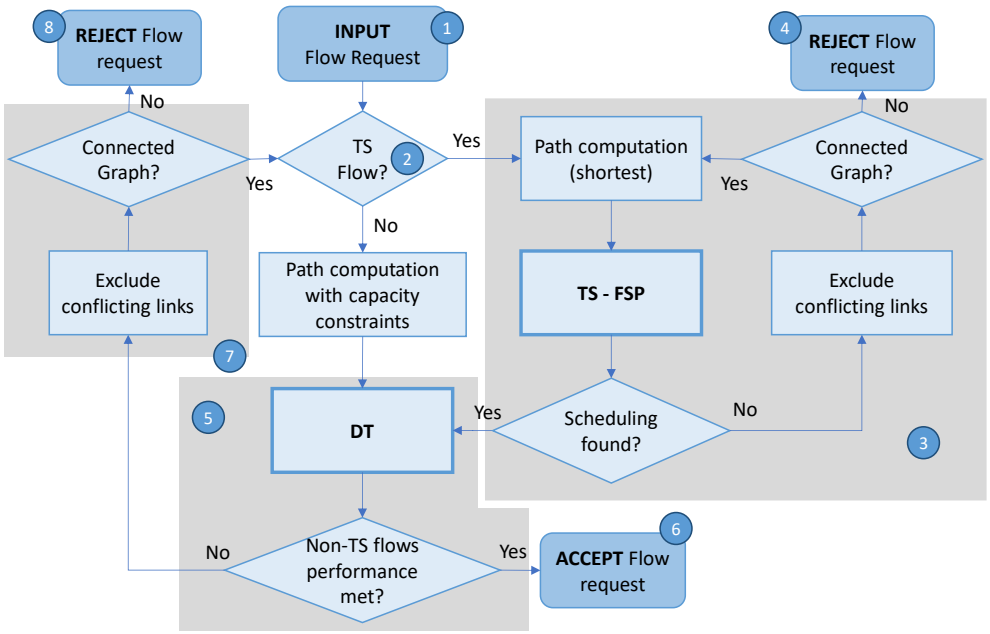

Figure 49. Updated Provisioning Workflow

8.2 TS-FSP Problem Definition

TS-FSP is executed for a TS flow request to be served on a computed path with the objective of reserving resources along that path to support the flow. Changes in the scheduling of already deployed TS flows might be needed, so TS-FSP needs to determine the new resource allocation for those TS flows which resource allocation changes. The resources to be allocated for the TS flows are a set of time windows, with duration specific for each TS flow, on every hop along the defined path. The resource allocation repeats with a given periodicity, which is also specific to the TS flow. In the case that the required resources cannot be reserved for the TS flow, the flow request is blocked. The transmission on any link e in the network is organized in terms of a SF, which consists of a set of time slots in the range [1..tmax], each of duration τe, where each time slot t can be allocated to only one TS flow. To that end, a resource allocation window (Tf) with a number of contiguous time slots for each TS flow f is computed, whose aggregated capacity considers the specifics of flow f. Formally, TS-FSP can be formally stated as the following optimization problem:

Given:

- The network topology G(*E*), modelled as a set of *directed* links *E*. Each link is characterized by: *i*) the speed of the interfaces B_e. In the case that an interface can offer different speeds (e.g., it is common that wireless interfaces can adapt their modulation format as a function of the quality of the signal of the receiver), the lowest speed is considered. This assumption

guarantees the performance of the TS flows even under the worst-case scenario; *ii*) its transmission delay d_e.

- The duration T of the SF. We assume a fixed duration that limits the longer periodicity of the TS flows that can be served. In every link e, a superframe SF_e, in the form of an ordered list, is defined. Note that the duration of each time slot τ_e is defined by speed B_e. In addition, each link can be *full-duplex* or *half-duplex*, where the former links have time slots available during the whole SF_e duration, whereas the latter links have time slots available only during part of the SF_e duration.
- The set of TS flows F already deployed in the network. Each TS flow f is being served through a path defined by a set of links E_f. In addition, the TS flow f has a window T_f of time slots reserved in each link e, which repeats during the SF with periodicity P_f. Finally, the maximum delay that f can support is defined by δ_f and the maximum jitter is defined by υ_f. The delay is computed for every period and the jitter is computed as the difference between the maximum and minimum delay in the different periods.
- The current scheduling plan of the network NSF = $\{SF_e, \forall e \in E\}$. Every SF_e defines the allocations of slots to flows, i.e., $SF_e = [s_{ft}]$, where every s_{ft} identifies the TS flow f to which slot t is allocated to; 0 otherwise.
- A new TS flow request $r=(E_r, T_r, P_r, \delta_r, \upsilon_r)$. The new request r can be served iff time slots can be reserved along path E_r satisfying the size of the allocation window T_r, and the periodicity P_r, so that delay and jitter constraints are met. Changes in the scheduling of the existing TS flows can be made provided that their constraints are also met.

Objective: To minimize total jitter and the number of TS flows that change their resource allocation, as a way to minimize jitter transients every time a new TS flow is established.

In the case that it is feasible to serve the TS flow request, the new scheduling plan for the network NSF'=$\{SF'_e\}$ is returned.

8.3 DT Extensions and Updates

Estimation of KPIs of requested and already deployed non-TS flows is based on emulating a *partition* of the real network scenario defined by the path though out the requested flow will be deployed. As introduced in D1.1 and D1.2, the emulation carried out in the DT is based on three components, i.e., *generators*, *queues*, *links*, and *sinks*, that can reproduce the expected traffic, as well as the real network devices and links with high accuracy and fine granularity. Following, we describe the main extensions of generators and queues, as well as for the DT architecture and main KPI evaluation workflow.

8.3.1 Traffic generators

Generators produce synthetic flow traffic at two different levels:

1) **at macroscopic level**, the *scale* (traffic intensity) is generated according to a periodical profile (e.g., daily) and with coarse resolution (e.g., one value per hour);
2) **at microscopic level** and for each scale value, a fine resolution calculation (e.g., at µs scale) of a short period (1 to 10 seconds) is conducted with traffic flows generated following probability distributions characterizing inter-arrival burst and packet time, and burst and packet size.

Figure 50 shows an example of macroscopic (a) and microscopic (b) traffic generated for three flows belonging to different classes: *i*) an isochronous TS flow, *ii*) a non-TS flow with QoS committed, and *iii*) a non-TS BE flow with no stringent requirements.

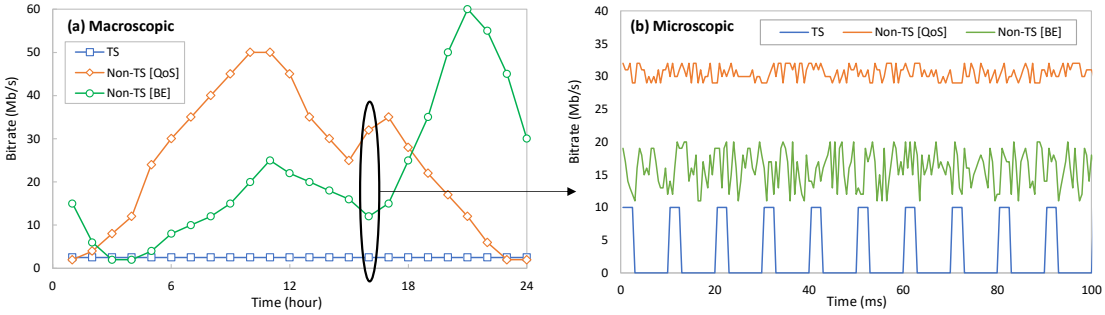

Figure 50. Macroscopic (a) and microscopic (b) flow traffic examples

8.3.2 Queue models

Flow queue models are based on the time-dependent ones presented in D1.1 and used to emulate TS-capable interfaces. Queue service rate is pre-empted at the beginning of a microscopic time period for a duration that depends on the amount and interval length of the existing and/or requested TS flows on that interface, while the remaining time in the period is available for non-TS flows according to their priority.

Starting from the models defined in D1.1, we present a priority queue model that takes into account the pre-emption caused by TS-flows scheduling and characterizes the individual queue that serves every class of non-TS traffic according to its priority i.e., non-TS QoS flows have more priority than non-TS BE ones. The generic queue model for every class is as follows:

$$q'_c(t) = \hat{X}(q_c(t), t) - Y\big(q_c(t), \mu_c(t)\big), \forall c \in \{QoS, BE\}$$

Being $q_c(t)$ the dynamics of the state (i.e., queued traffic in bytes) of class c, $\hat{X}(\cdot)$ the amount of input flow received (b/s), $Y(\cdot)$ the flow leaving the queue (b/s), and $\mu_c(t)$ the server rate assigned to class c at time t.

Given a SF with duration T, which is divided into a number of time slots of duration τ_e, we define a Boolean parameter $z_t \in \{0,1\}$ that is equal to 1 if the time slot t is allocated by a TS flow. Note that this parameter will be provided by TS-FSP after solving its optimization problem. Assuming that μ is the total speed of the interface that models the queue, the equation that defines the server rate of every non-TS class is as follows:

$$\mu_{QoS}(t) = z_t \cdot \min\left(\hat{X}\big(q_{QoS}(t), t\big), \mu\right)$$

$$\mu_{BE}(t) = z_t \cdot \min\left(\hat{X}(q_{BE}(t), t), \max\left(0, \mu - \mu_{BE}(t)\right)\right)$$

In other words, μ_{QoS} takes as much speed as required for QoS traffic conditions (never exceeding μ) if and only if the time slot is free from TS flows. Regarding μ_{BE}, the approach is similar but with the remaining speed after serving QoS needs.

Figure 51 shows both the output traffic (a) and queue state (b) for the non-TS flows in the example of Figure 50.b. The input traffic in that figure is propagated through the queue modelled as explained above. As can be observed, both QoS and BE are blocked during time reserved to TS flow. Then, due to the queued QoS traffic, it takes full interface speed until QoS queue is

empty. At that time, both QoS and BE can be served, always keeping maximum priority to QoS one.

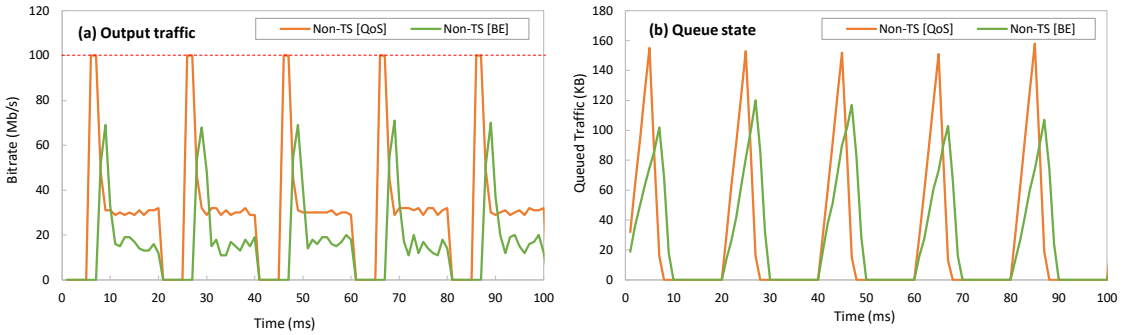

Figure 51. Macroscopic (a) and microscopic (b) flow traffic examples

Let us now show how DT is able to model a more complex queue system consisting in several generators and queues. To this end, Figure 54 shows one of the topologies that are going to be used afterwards for numerical evaluation, that contains two existing flows (QoS and BE). Thus, the DT models a partial topology consisting in a set of traffic generators, queues, and sinks that characterizes the end-to-end path within the domain of the request under evaluation. As a result of this, domain end-to-end KPIs will be provided for this request. Moreover, the existing flows that share network interfaces with the new request are added at the beginning of the shared segment. As a consequence of this, partial KPIs will be provided for these flows.

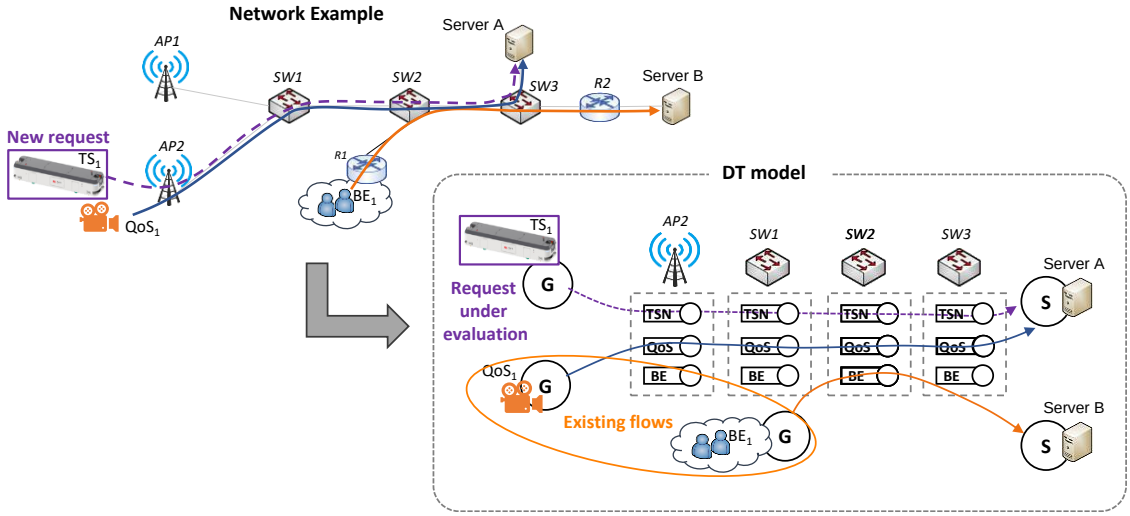

Figure 52. DT Modelling example

8.3.3 KPI estimation

The propagation of the generated traffic for the flows through the defined queuing system results in metrics, such as queued traffic, that are afterwards used to compose *flow* KPIs, such as e2e delay. Without loss of generality, the DT produces two types of KPIs estimation: *i*) *e2e* or nominal values, which are provided only for non-TS requests; and *ii*) *variations* or delta values (Δ), computed for already deployed flows as the KPI increment or decrement for each flow if the new request would be finally deployed, in the network partition defined by the path of the request.

As introduced in D1.2, the following list of KPIs can be measured and quantified by the DT for the flow request under evaluation and the rest of established flows:

- **Throughput** (or data rate): Average volume of traffic (in Mb/s or Gb/s) at the input and/or output of a queuing system at a given time.
- **Traffic loss ratio**: Average percentage of traffic that is rejected at the input of a queuing system due to lack of available queueing capacity at a given time.
- **Delay** (or latency): Average of the elapsed time (in ms) that the traffic experiences to traverse a queuing system from input to output at a given time.
- **Jitter**: Standard deviation of the elapsed time (in ms) that the traffic experiences to traverse a queuing system from input to output at a given time.

Figure 53 shows the details of the performance evaluation process that is executed by the DT during the provisioning process of flow request *r* on path E_r. The first step consists in retrieving the set of links (*E'*) and existing flows (*F'*) from the internal network DB that share links and interfaces with request *r* (step 1 in Figure 53). These subsets feed two different processes running in parallel: on the one hand, traffic generators are built and configured to generate traffic according to *r* and *F'* specifications (2), and on the other hand, queues are configured to emulate the network subset *E'* (2'). The outputs of both processes are used by a queuing system composer (3) that concatenates the queues and bonds the generators to the beginning of each flow and/or segment. The propagation of the generated traffic through the composed queuing system (4) creates a set of metrics *X* that, jointly with the available monitoring data *Y* of the existing flows, are used to estimate the KPIs (5) that are finally returned.

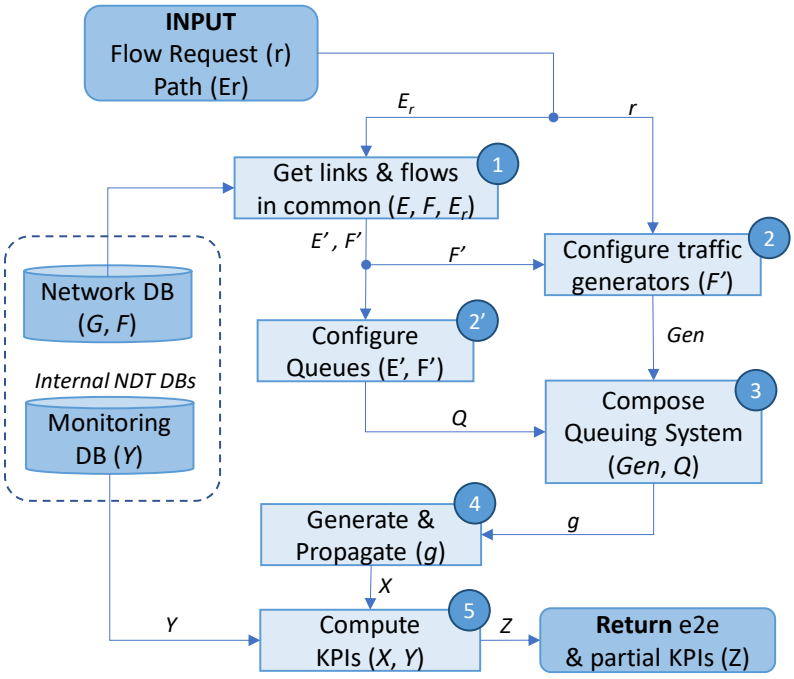

Figure 53. DT main procedure

8.4 ILLUSTRATIVE RESULTS

We firstly focus on evaluating the new models of traffic generator and queue proposed in Sections 8.3.1 and 8.3.2 in comparison with the discrete simulation results presented in Section 7. In particular, we reproduced in the DT the topology and configuration of Test 3 described in Section 7.3.7. The analysis focuses on the maximum QoS traffic that can be admitted with no significant delay impact. In view of Figure 42 to Figure 46, we assume that that maximum traffic is achieved when outage delay increases in 1ms with respect to the baseline observed for low loads. Without loss of generality, we assume that that delay increase is due to a significant increase on queueing delay introduced by buffered traffic. Therefore, we extract from the DT the maximum QoS traffic when queue state presents a significant and persistent amount of buffered traffic, i.e., queueing delay starts affecting e2e delay.

Table 28 shows the values of maximum QoS traffic obtained for both approaches (Matlab-based Scheduler vs DT model) under 4 different configurations combining: *i)* a scheduling policy (either proposed dynamic or benchmarking), *ii)* a load of ISO TSN windows (50% or 75%). Note that the column related to Matlab-based scheduler has been obtained from the graphs in Figure 42 to Figure 46. As can be seen, both approaches provide similar results, which validates the use of DT models for accurate performance estimation.

Scheduler	ISO TSN windows occupancy (%)	Maximum QoS traffic [Matlab]	Maximum QoS traffic [DT]	Error [%]
Benchmark	50%	8.5 Mb/s	7.9 Mb/s	7%
	75%	4.2 Mb/s	3.6 Mb/s	14%
Proposed Dynamic	50%	15 Mb/s	14.1 Mb/s	6%
	75%	13.5 Mb/s	11.2 Mb/s	17%

Table 28. Comparative analysis between Matlab-based scheduler and DT

Next, we focus on the application of the proposed KPI estimation procedure in Section 8.3.3 for more complex scenarios involving several paths from all considered service classes, and combining wireless and wired network segments.

The first scenario is represented in Figure 54, that matches the network modelling example in Figure 52. where a new provisioning request (TS1) is evaluated. The traffic of this request has a 10 ms period and constant generation rate. Moreover, it requires to be served with a maximum delay lower or equal to 0.5 ms, and a maximum jitter lower or equal to 0.1 ms. Regarding the already established flows, QoS flow contains traffic with data traffic at constant generation rate, whereas BE flow transports with video traffic with daily variation rate.

The estimated performance measured at the microscopic level is evaluated for a macroscopic level of 1 day of duration. The results are presented in Figure X7, where the DT estimation (in blue) is compared with the requirements (in red), if any. As can be observed, KPIs are guaranteed during the whole day. Therefore, the provisioning of TS1 flow could be accepted without degrading the performance of existing flows.

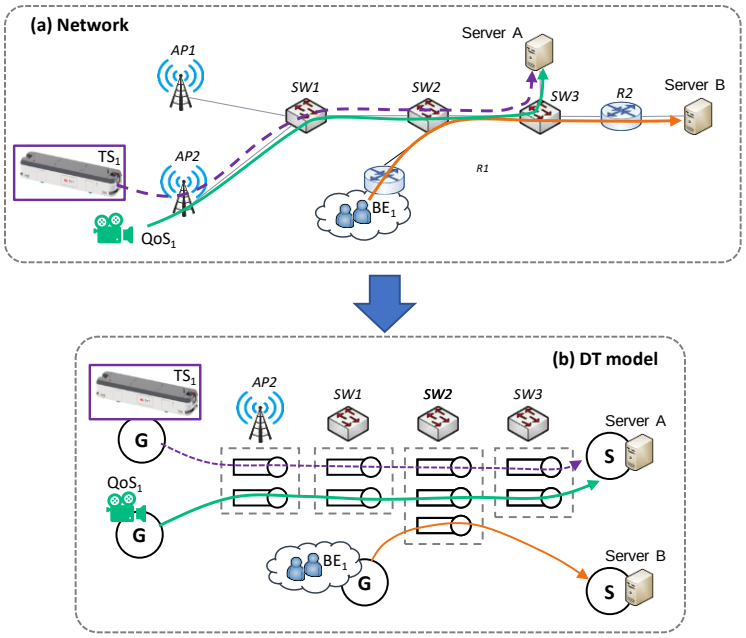

Figure 54. Modelling for TS1 request evaluation

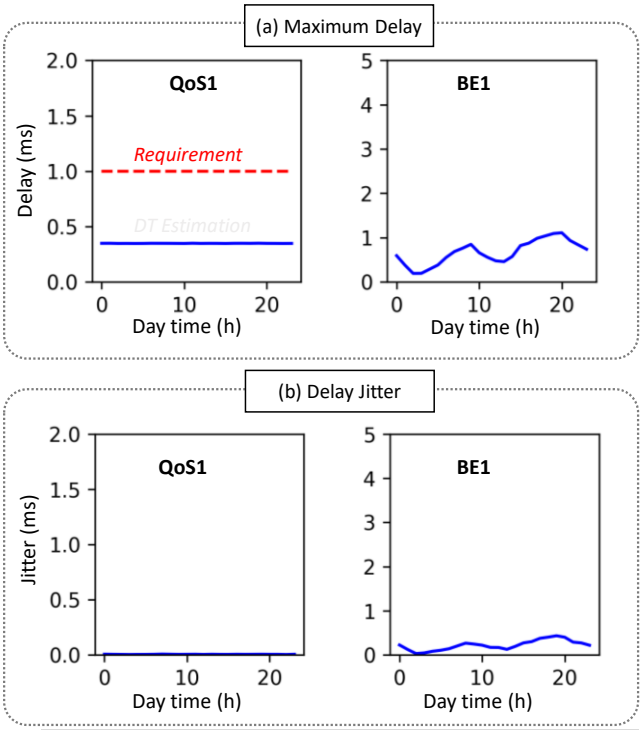

Figure 55. Expected performance of non-TS flows under TS1 provisioning

The second scenario assumes that the previous TS1 request was established, and evaluates the request of a new non-TS QoS service (QoS2). This request contains data from sensors, with daily variable generation rate and 1ms of maximum delay requirement. The topology and model can be seen in Figure 56, and the performance evaluation in Figure 57. In view of the latter, we can conclude that provisioning QoS2 will violate its own maximum delay requirements, as well as for other competing QoS traffic flows. Therefore, in this case, the DT will provide relevant

information to allow the connectivity manager to not accept that request in the provisioned path. This can trigger further actions such as find an alternative path excluding the links that provided the delay degradation (this is an output of the DT to potentially exploit in further work).

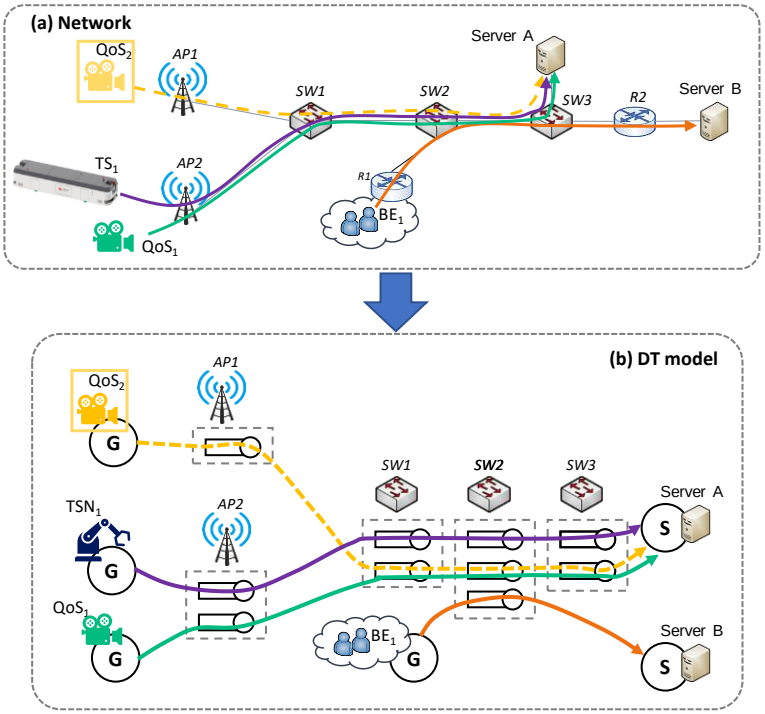

Figure 56. Modelling for QoS2 request evaluation

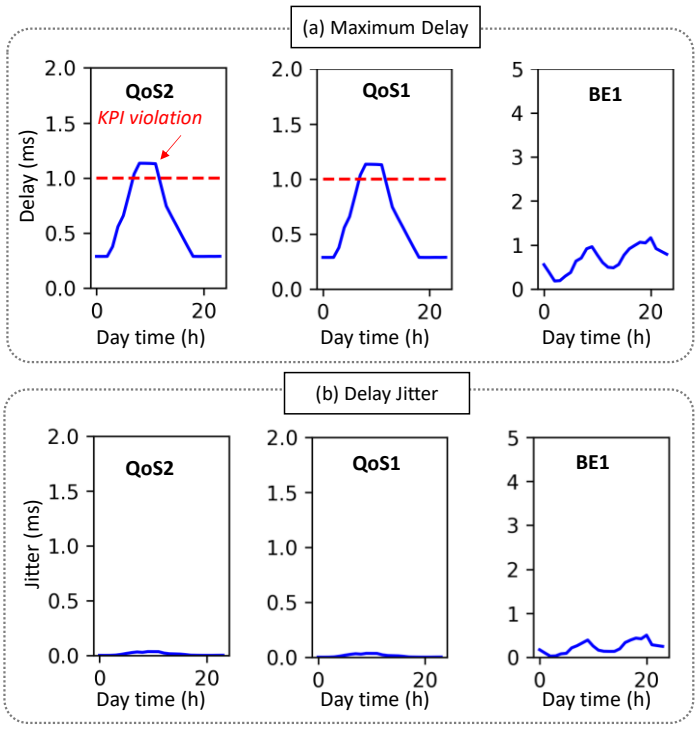

Figure 57. Expected performance of non-TS flows under QoS2 provisioning

9 CONCLUSIONS AND OUTLOOK

This document updated deliverable D1.2 with the activities carried out during the second year of the TIMING project related to activities in WP1. Specifically, the document has provided updated specifications and preliminary performance evaluations of the TIMING components, including the Wi-Fi and Ethernet TSN nodes, the TSN controllers, the connectivity manager, the scheduling algorithms, and the TSN models for the digital twin. Additionally, a novel component has been introduced, extending the control plane architecture. This new component, called Time Sensitive Flow Scheduler Planner (TS-FSP), is devoted to network-wide resource allocation. The final specifications and performance evaluation of the components of the TIMING solution, related to activities in WP1, will be reported in the forthcoming document D1.5.

10 REFERENCES

[5G-19], “A 5G Traffic Model for Industrial Use Cases”, November 2019. Available at: https://5g-acia.org/whitepapers/a-5g-traffic-model-for-industrial-use-cases/

[5G-21], “Integration of 5G with Time-Sensitive Networking for Industrial Communications”, February 2021. Available at: https://5g-acia.org/whitepapers/integration-of-5g-with-time-sensitive-networking-for-industrial-communications/

[ENP] E-lighthouse Network Planner. Online. https://e-lighthouse.com/products/networkplanner

[MAT21] "Physical Layer Abstraction for System-Level Simulation". Matlab documentation. Matlab R2021

[RAM01] Kavita Ramanan, Alexander L. Stolyar "Largest Weighted Delay First Scheduling: Large Deviations and Optimality," The Annals of Applied Probability, Ann. Appl. Probab. 11(1), 1-48, (February 2001)

[SEI18] Ó. Seijo, Z. Fernández, I. Val and J. A. López-Fernández, "SHARP: A novel hybrid architecture for industrial wireless sensor and actuator networks," 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 2018, pp. 1-10

[VIN04] A. Vinko Erceg, Laurent Schumacher, Persefoni Kyritsi, Molisch, D. S. Baum, A. Y. Gorokhov, and C. Oestges, “TGn Channel Models,” 2004.

[WLA99] “IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, High-speed Physical Layer in the 5 GHZ Band,” *IEEE Std 802.11a-1999, Supplement to IEEE Std 802.11-1999.*

[WLA21] “IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks--Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Ame,” *IEEE Std 802.11ax-2021 (Amendment to IEEE Std 802.11-2020)*, 2021.