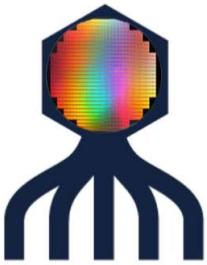


Optical circuit switched time sensitive network architecture for high-speed passive optical networks and next generation ultra-dynamic and reconfigurable central office environments

D2.3 Final modelling of System Architecture and Component/Controller Specifications

Project number:	**101070009**
Project acronym:	**OCTAPUS**
Project title:	Optical circuit switched time sensitive network architecture for high-speed passive optical networks and next generation ultra-dynamic and reconfigurable central office environments
Programme:	HORIZON-CL4-2021-DIGITAL-EMERGING-01-06
Start date of the project:	01/09/2022
Duration:	42 months

Deliverable type:	Report
Deliverable reference number:	101070009/ D2.3/ Final | V.1
Deliverable title:	Final Modelling of System Architecture and Component/Controller Specifications
WP contributing to the deliverable:	WP2
Due date:	31/08/2024
Actual submission date:	06/12/2024

Dissemination level:	Public
Revision:	V1.0

Abstract: OCTAPUS aims to deliver an agile, low-cost and energy-efficient Photonic Integrated Circuit (PIC) technology framework to address the capacity-latency predicament currently faced by incumbent Central Offices (COs) and re-architect the Next Generation Central Office (NGCO) ecosystem, by transparently upgrading its capacity beyond 51.2 Tbps, through an innovative optically-switched backplane and transceiver toolkit, combined with Time Sensitive Networking (TSN) transceiver engines under the overall control of a hierarchical Software Defined Networking (SDN)-driven multi-domain coordinator. The optical and electronic components developed within OCTAPUS will be synergized with OCTAPUS's TSN nodes and SDN control plane to yield an NGCO architecture that can support three different layers of latency (for ultra-real-time, Time Sensitive and Best Effort traffic, respectively), along with deterministic latency guarantees for time-sensitive services, such as mobile fronthaul and Ultra Reliable Low Latency Communications (URLLC). This document presents the final hardware and software specifications of all Data and Control Plane components and software controllers of the OCTAPUS NGCO, along with an extensive simulation-driven modeling of the final OCTAPUS architecture. Specifically, the hardware specifications of the OCTAPUS Silicon Nitride (SiN)- and Indium Phosphide (InP)-based Photonic Integrated Circuits (PICs) are provided, followed by the specifications of the Electronic ICs (EICs) at the Tx/Rx side and the C- and O-band diplexers. The supported features of the TSN Data plane are enumerated and the position of the TSN Centralized Network Controller (CNC) within the hierarchical NGCO Control Plane, coordinated by a multi-domain SDN coordinator, is described, along with the operation of additional domain-specific controllers, namely the OCS (Optical Circuit Switching) SDN Agent and the SDN Diplexer Agent. To evaluate the final proposed OCTAPUS architecture, a simulation-based evaluation study of the effect caused by the crosstalk and insertion losses of the 2x2 switching units comprising the Optical Circuit Switched (OCS) backplane is performed for 4x4 and 8x8 Benes topologies, and the Extinction Ratio (ER) and Insertion Loss (IL) values satisfying the targeted BER are determined. Furthermore, a holistic discrete-event-based evaluation framework is presented capturing realistic aspects of the end-user traffic generated by mobile users and showcasing the effects introduced by different TSN shapers, including the Credit Based Shaper (CBS) and Time Aware Shaper (TAS). An investigation of the performance/complexity tradeoff inherent in the TSN resource allocation algorithms proposed within OCTAPUS is also performed, resulting in the development of a heuristic algorithm achieving an excellent balance in the above tradeoff.

Keywords: Next Generation Central Office, InP TxRx, SiN Phase Change Materials, Optical Circuit Switching, High Speed Drives and TIAs, Glass Diplexers, Time Sensitive Networking, Software Defined Networking multi-domain coordination, discrete event simulation, resource allocation algorithms.

Author List:

Organization	Author
ORAN	Fabienne Saliou, Philippe Chanclou, Georges Gaillard
COSM	Elina Theodoropoulou, George Lyberopoulos
AUTH	Athanasios Papadakis, Sophia Petridou, Stella Markou, George Kalfas, Christos Vagionas, Agapi Mesodiakaki, Marios Gatzianas, Theodoros Moschos
TEEM	Hawraa Atwi, Clément Fresse, Cédric Cassagnettes
NXW	Pietro Piscione, Ahmed K Abdulwahed, Pietro Giardina
COMC	Christopher Alsted, Voica Gavrilut
SOTON	Frederic Gardes, Ipsita Chakraborty, Xingshi Yu, Ioannis Zeimpekis, Thalia Dominguez Bucio
SMART	Grzegorz Sobczak, Mohamad Dernaika
IMEC	Peter Ossieur
IZM	Michail Symeonidis, Sevag Abadian
NVIDIA	Stefanos Dris, Nikos Argyris

Copyright © 2022 the OCTAPUS Consortium. All rights reserved. This document may not be copied, reproduced or modified in whole or in part for any purpose without written permission from the OCTAPUS Consortium. In addition to such written permission to copy, reproduce or modify this document in whole or part, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced.

Disclaimer: *The information, documentation and figures available in this deliverable are written by the OCTAPUS Consortium partners under EC co-financing (HORIZON-CL4-2021-DIGITAL-EMERGING-01-06) and do not necessarily reflect the view of the European Commission. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.*

Table of Contents

TABLE OF CONTENTS **5**
ABBREVIATIONS **6**
1. EXECUTIVE SUMMARY **8**
2. INTRODUCTION **9**
2.1 PURPOSE OF THIS DOCUMENT 9
2.2 DOCUMENT STRUCTURE 9
2.3 AUDIENCE 10
3. SYSTEM AND NETWORK ARCHITECTURE **11**
4. SPECIFICATIONS OF OCTAPUS COMPONENTS, CONTROLLERS AND INTERFACES **13**
4.1 PHY-LAYER OPTICAL TRANSCEIVERS AND ELECTRONICS 13
4.1.1 Optical transceiver and electronic driver/TIA IC specifications *13*
4.2 OCS-BASED NGCO BACKPLANE 17
4.2.1 2x2 PCM cell unit *17*
4.2.2 Controlling the PCM cell units *19*
4.3 OPTICAL DIPLEXERS AND INTERPOSERS 20
4.4 TSN DATA PLANE 21
4.5 OCTAPUS SDN-BASED MULTI-DOMAIN CONTROL PLANE 22
4.5.1 NGCO Network intelligence *24*
5. MODELING AND EVALUATION OF THE OCTAPUS ARCHITECTURE **28**
5.1 OCTAPUS TX/RX PHY-LAYER EVALUATION FOR INTRA-NGCO LINKS 28
5.2 DISCRETE EVENT-BASED EVALUATION FRAMEWORK FOR OCTAPUS ARCHITECTURE 30
5.2.1 Multiple-vehicle scenario *31*
5.2.2 Two-vehicle TSN scenario *32*
5.2.3 Multiple-vehicle scenario results *33*
5.2.4 Two-vehicle TSN scenario results *35*
5.2.4.1 Without shaper 36
5.2.4.2 Time Aware Shaper (TAS) 37
5.2.4.3 Credit Based Shaper (CBS) 41
5.3 OPTIMIZATION-DRIVEN RESOURCE ALLOCATION IN OCTAPUS 43
5.3.1 Joint scheduling and routing of time-sensitive flows: a complexity study *43*
5.3.1.1 Theoretical model 43
5.3.1.2 Proposed algorithms 46
5.3.1.3 Performance evaluation 48
5.3.2 Robust joint scheduling and routing of time-sensitive flows *53*
6. SUMMARY AND CONCLUSIONS **56**
REFERENCES **57**

Abbreviations

3GPP	3rd Generation Partnership Project
AOP	Average Optical Power
BE	Best Effort
CBS	Credit Based Shaper
CMOS	Complementary Metal Oxide Semiconductor
CNC	Centralized Network Controller
CO	Central Office
CQI	Channel Quality Indicator
CUC	Centralized User Configuration
CTLE	Continuous Time Linear Equalizer
DBR	Distributed Bragg Reflector
EAM	Electro Absorption Modulator
EIC	Electronic Integrated Circuit
E2E	End-to-End
EML	Electro-Absorption Modulated Laser
ER	Extinction Ratio
gNB	gNodeB
gPTP	Generalized Precision Time Protocol
HS-PON	High Speed Passive Optical Network
IL	Insertion Loss
InP	Indium Phosphide
ITO	Indium-Tin-Oxide
MZI	Mach-Zehnder Interferometer
NGCO	Next Generation Central Office
NRZ	Non-Return-to-Zero
OCS	Optical Circuit Switching
OOK	On-Off Keying
PCB	Printed Circuit Board
PIC	Photonic Integrated Circuit
PCM	Phase Change Material

PD	Photo Detector
PDK	Process Design Kit
PON	Passive Optical Network
QoS	Quality of Service
RF	Radio Frequency
RIN	Relative Intensity Noise
RU	Remote Unit
SDN	Software Defined Networking
SiN	Silicon Nitride
SINR	Signal to Interference and Noise Ratio
SOA	Semiconductor Optical Amplifier
SOI	Silicon-on-Insulator
SSC	Spot Size Converter
TAS	Time Aware Shaper
TIA	TransImpedance Amplifier
TSN	Time Sensitive Networking
UE	User Equipment
ULS	Uplink Switch
URLLC	Ultra Reliable Low Latency Communications
WAFT	Waveguide Array to Fiber Transposer

1. Executive Summary

This Deliverable is the final report on the activities performed within Work Package #2 of the OCTAPUS project. Motivated by the use cases and network topology previously presented in Deliverable D2.1, the current Deliverable contains the specifications of all major hardware and software components, as well as the associated interfaces, in the OCTAPUS NGCO Data and Control plane, including:

- the SiN based (re)configurable Phase Change Material (PCM)-based OCS backplane connecting the main NGCO blocks (i.e., the Uplink switches and IF cards),
- the InP-based TxRx optical transceivers and associated Electronic ICs,
- the C/O-band optical diplexers enabling the NGCO's Express paths,
- the TSN features supported by the Uplink switches and IF cards,
- the SDN multi-domain coordinator, its interactions with domain-specific (TSN and OCS) controllers, and the employed communication interfaces.

This Deliverable also presents a multi-faceted simulation-based evaluation study of the OCTAPUS architecture capturing many of its unique aspects, including:

- a PHY-layer analysis of the effect of the intervening OCS (in terms of introduced Extinction Raio and Insertion Losses) on the potential degradation of the intra-NGCO link quality, focusing on 4x4 and 8x8 Benes OCS architectures targeting a BER of 10^{-12} at 50 Gbps.
- a detailed, discrete-event-based simulation framework, incorporating all layers of the 5G NR protocol stack and supporting common TSN shapers (i.e., CBS and TAS), to evaluate the mobility-induced time-varying traffic demands and the effect of applying the aforementioned shapers on realistically simulated 5G traffic.
- an optimization-driven framework for efficient resource allocation of Time Sensitive streams and the development of efficient algorithmic heuristic achieving an excellent performance/complexity tradeoff.

The final specifications, recommendations and insights gained from WP2, as described in the current Deliverable, have provided valuable guidance for the more technical development work performed in Work Packages 3,4 so far and also offered a solid foundation for the design and preparation of the lab and field trials planned after M30.

2. Introduction

2.1 Purpose of this document

The present Deliverable D2.3, being the final deliverable of OCTAPUS WP2, describes the outcome of the activities performed in the following WP2 tasks, while also providing an update on the relevant work previously reported in the interim review meeting (held online in April 2024):

- Task 2.3, titled “Subsystems, components specifications and interfaces definition”.
- Task 2.4, titled “Design and specification of TSN and SDN interfaces”.
- Task 2.6, titled “System and component modelling and network simulations”.

Work on the following Tasks was either already completed by April 2024 or any additional work performed does not warrant any updates:

- Task 2.2, titled “System architecture and network topology”.
- Task 2.5, titled “Definition of assembly process flow”.

To this end, the current document provides the final specifications of all hardware and software components comprising the functional blocks of the OCTAPUS NGCO, in both Data and Control Planes, as they have been developed to serve the OCTAPUS Use Cases reported in Deliverable D2.1 [1]. Despite some inevitable overlapping with Deliverable D2.2, titled “Initial Modelling of System Architecture, Component/Controller Specifications and Assembly Process Flow Definition” [2], the current document presents the updated and fine-tuned hardware specifications of the optical transceiver and switching modules (in both electrical and optical domains), the optical diplexers and electronic drivers/TIAs, along with the updated description of the communication interfaces enabling the SDN-based, TSN-compliant Control Plane.

This document also contains an extensive simulation-based evaluation study, updating and expanding upon the work previously presented in D2.2, as follows:

- The intra-NGCO link (Uplink switch to IF card) is further studied in the PHY layer in terms of the effects introduced by the intervening OCS, by focusing on 4x4 and 8x8 Benes architectures and determining, via simulations, the necessary Extinction Ratios and Insertion Losses of the basic 2x2 switching element towards achieving the targeted BER of 10^{-12} at 50 Gbps.
- The discrete event-based simulation framework developed for the evaluation of the overall OCTAPUS network topology and architecture is expanded with the Simu5G library, to allow for more realistic modeling of the 5G NR network stack, and the effect of applying different TSN shapers (i.e., CBS and TAS) is investigated for different traffic scenarios, focusing on time-varying mobility-induced traffic demands with diverse Quality of Service (QoS) requirements and their treatment by the OCTAPUS NGCO.
- The optimization-driven modeling framework for resource allocation (i.e., joint routing and scheduling) of TSN flows within the OCTAPUS NGCO, previously reported in D2.2, is updated to account for different flow periodicity properties and an extensive simulation-based scalability study of newly proposed algorithms is performed to determine the best tradeoff between network performance and computational complexity.

The description of the OCTAPUS components and the evaluation study contained in this document provide an architectural/functional view of the OCTAPUS solution. A more technical discussion of specific hardware components, such as the OCS backplane and WAFT interposers, or the functional decomposition of the SDN and TSN controllers into subcomponents, has been provided in Deliverables D3.1 [3] and D4.2 [4], in M14 and M18 respectively, and will be updated in upcoming Deliverables D3.2 (due M24) and D4.3 (due M36).

2.2 Document structure

The present Deliverable is split into 6 Chapters:

- **Chapter 2** (i.e., the current Chapter) contains an outline of the document’s structure and contents.

- **Chapter 3**, titled *"System and Network Architecture"* provides a brief, high-level overview of the main building blocks (i.e. Uplink switch, IF card, OCS backplane, optical and electronic transceivers, TSN/SDN controllers etc.) of the OCTAPUS NGCO architecture and their role. Although the architecture definition remains unchanged with respect to what was reported in D2.2, this information is essential to establish the proper context of the OCTAPUS solution and refresh the reader's memory. Deliverable D2.1 provides a more expanded treatment of the material presented herein.
- **Chapter 4**, titled *"Specifications of OCTAPUS components, controllers and interfaces"*, describes all hardware and software components of the OCTAPUS NGCO Data and Control planes, as well as their associated interfaces enabling the required interactions among the OCTAPUS components. Detailed specifications of the OCTAPUS transceivers, OCS backplane, optical diplexers and TSN and SDN network controllers are provided.
- **Chapter 5**, titled *"Modeling and evaluation of the OCTAPUS architecture",* presents the simulation-based evaluation studies mentioned above, their generated results and conclusions drawn from them.
- **Chapter 6** contains the summary and conclusions of the deliverable and completes the deliverable.

2.3 Audience

This document is confidential and intended only for members of the OCTAPUS project Consortium and the EU project evaluation services.

3. System and network architecture

The high-level architecture of the OCTAPUS NGCO and its targeted use cases have been described in D2.2 [2] and D2.1 [1], respectively, and no changes have occurred in the interim. However, since some basic knowledge of the OCTAPUS functional blocks and architecture is necessary as a starting point to establish the context within which the different OCTAPUS components will serve their roles, as well as justify some of their specifications, we next provide a brief outline of the main OCTAPUS components, in both the Data and Control planes, as shown in Fig. 3.1. We note that Fig. 3.1 only depicts the OCTAPUS NGCO itself without its service end-points, which are typically located in the metro and access segments (i.e., to the left and right of the NGCO in Fig. 3.1). A description of the different types of X-haul and URLLC services supported by the OCTAPUS NGCO is also provided in D2.1.

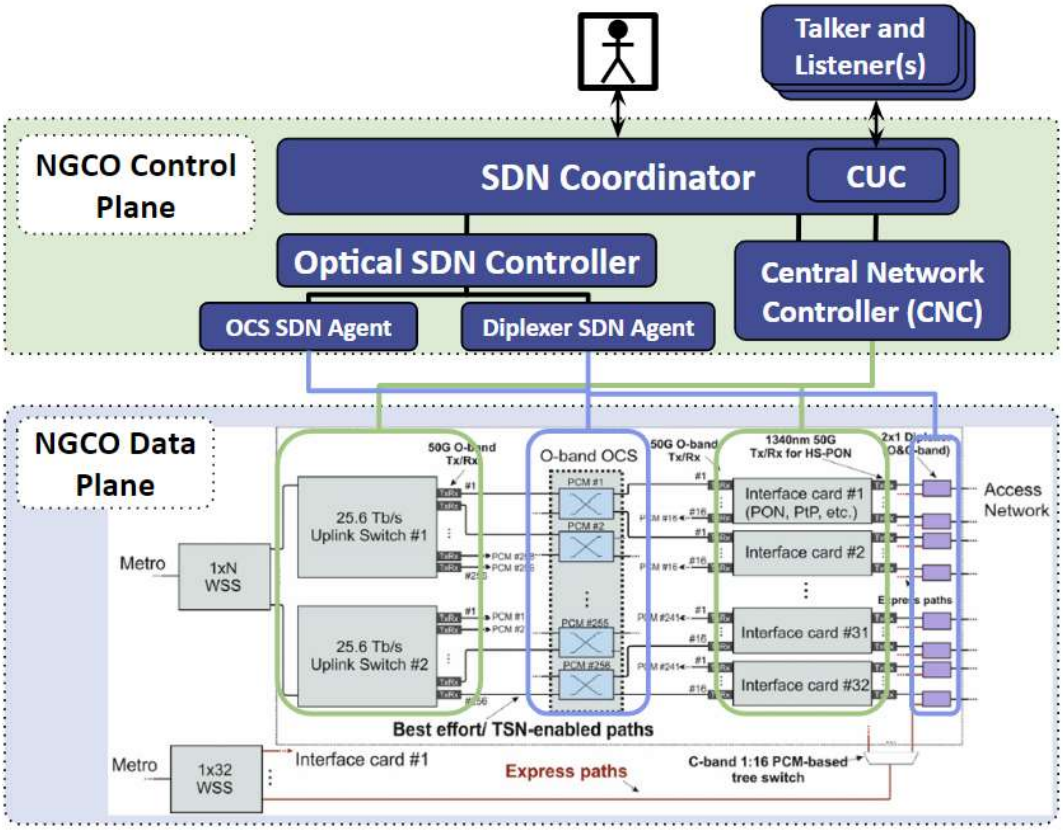


Fig. 3.1: The OCTAPUS NGCO architecture (high-level view).

The OCTAPUS NGCO Data Plane contains the following elements:

- **Uplink Switches (ULS) and Interface (IF) cards**: these entities serve the same role as their legacy CO counterparts, with the Uplink Switches acting as aggregator points providing connectivity to the metro network and the IF cards serving the NGCO's downstream subscribers (e.g., Passive Optical Network (PON) line cards, 4G/5G base stations or disaggregated Remote Units (RUs), etc.). The Uplink switches and IF cards employ high-speed 50 Gbps transceivers at each port and can scale to high port densities, reaching up to 256 ports per ULS and 16 ports per IF.
- **OCTAPUS low-power 50 Gbps transceivers:** as mentioned above, they are employed by the ULS and IF cards and combine O-band InP-based PICs (integrating lasers and EAMs/PDs on the same substrate) with low-power driver and TIA EICs supporting burst-mode operation for High Speed Passive Optical Network (HS-PON) support.
- **Reconfigurable OCS backplane**: this is an SDN-controllable, ultra-fast-reconfigurable all-optical OCS backplane consisting of multiple O-band, low-loss 2x2 PCM-based non-volatile photonic switching devices on a SiN platform. The different switching states of the PCM units, informally referred to as "straight" vs. "cross", are achieved by driving their heating circuits to

change the PCM material state between crystalline and amorphous via proper FPGA-generated control waveforms.

- **Express paths**: this is a limited number of dedicated paths, (e.g., 32 Express paths in Fig. 3.1, one for each IF card) targeting C-band operation for PON support and bypassing the entire NGCO to directly connect the metro network with the appropriate end-node (i.e., Optical Network Unit, 5G RU etc.) on the access side. These paths are intended to serve the most latency-demanding services, where even the packet processing by the ULS and IF cards within the NGCO may violate the service's latency budget. The actual decision of whether a light path should be established through the NGCO proper or the (bypass) Express paths is made via Wavelength Selective Switching (WSS) devices (shown in the left part of Fig. 3.1).
- **O/C-band diplexers and low-loss coupling interfaces to SiN and InP PICs:** low-loss, glass-based optical diplexers are employed at the InP TxRx of the IF cards, (i.e., on the right hand side of Fig. 3.1) to allow O-band PON traffic to pass through the NGCO from the access side (in the upstream traffic) while also routing C-band traffic towards one of the above Express paths, in an all-passive manner. The diplexers are integrated with WAFT-based interposers to provide low loss PIC-to-fiber coupling interfaces.
- **TSN-enabled Uplink switches and IF cards**: the Uplink switches and IF cards incorporate the TSN features of network-wide IEEE 802.1AS-based synchronization and scheduled transmissions via Time-aware shapers (according to IEEE 802.1Qbv), offering important benefits in terms of deterministic latency bounds and flow-prioritization, thus enabling applications with very tight delay budgets. The management of TSN flows requires additional Control Plane entities as described next.

In addition to the above Data Plane entities, the following software-implemented entities constitute the Control Plane of the OCTAPUS NGCO:

- **Centralized User Configuration (CUC):** this software agent collects the TSN traffic requests, and corresponding requirements in terms of End-to-End (E2E) throughput and latency, as they are initiated by the TSN flow end-points, so that a suitable schedule satisfying these requirements can be constructed by the CNC (described below). The CUC is logically embedded into the multi-domain SDN Coordinator described below.
- **Centralized Network Controller (CNC):** it receives the above information from the CUC, along with network topology information, and computes the routed path for each TSN flow (only) as well as the actual transmission schedule for each TSN node, in terms of proper transmission windows of a Time-Aware shaper. Any remaining transmissions windows are can serve standard (i.e., Best Effort (BE)) Ethernet traffic. Once the CNC determines the actual TSN path and schedule, it sends configuration messages to all TSN nodes, including the Uplink switches and IF cards.
- **Multi-domain SDN Coordinator:** this is the "brain" of the OCTAPUS NGCO containing all network intelligence necessary to coordinate higher layer functions (e.g., power/congestion-aware routing for BE traffic, load balancing via OCS reconfiguration etc.) based on network state information obtained through consistent monitoring. More details on the internal hierarchical substructure of the SDN Coordinator will be provided in Chapter 4.

The following Chapters contain more detailed component specifications for each of the above functional blocks of the OCTAPUS NGCO.

4. Specifications of OCTAPUS Components, Controllers and Interfaces

This Chapter contains the hardware specifications of all OCTAPUS components, including i) the OCTAPUS transceivers' optical and electronic (i.e., driver/TIA) components, ii) the PCM-based OCS backplane, and iii) the C/O-band diplexers and WAFT interposers for PIC-to-fiber connections. Additionally, the software functional specifications of the OCTAPUS hierarchical control plane entities, i.e., the SDN multi-domain coordinator (incorporating the CUC) and the TSN CNC controller are provided, along with a discussion of their interactions. Since the rationale for selecting the values of the parameters described below has also been explained in D2.2, the description here will be succinct and, mostly, in the form of tabulated data. Many of the hardware specifications appearing below have been derived based on extensive simulation studies, reported in Deliverable D3.1 [3]. Details on the mask layouts of the optical and electronic components have also been provided in Deliverable D3.1 and will be updated in Deliverable D3.2 (due M24).

4.1 PHY-layer optical transceivers and electronics

A new Process Design Kit (PDK) has been developed by SMART for the design of the OCTAPUS low-power, InP-based Tx and Rx PICs in the 1330-1345 nm O-band range. The PDK provides an extensive set of Building Blocks (BBs), including Distributed Bragg Reflector (DBR) lasers, Electro-absorption Modulators (EAMs) and Photodiodes (PDs).

The OCTAPUS O-band InP PICs target 50 Gbps, On-Off-Keying (OOK), Non-Return-to-Zero (NRZ) operation, with a bandwidth of at least 50 GHz, and an associated output power level above 15 mW. To meet these challenging requirements, the following Tx PIC designs have been finalized within OCTAPUS:

1. A chip design of 4 DBR lasers, each of them followed by EAM in the range of 1330-1345 nm.
2. A chip design of 4 DBR lasers, each of them followed by EAM and Booster Semiconductor Optical Amplifier (SOA).
3. A chip design of variations such as i) 1x4 laser followed by EAMs, ii) laser with Mach–Zehnder interferometer (MZI) + EAMs configuration, iii) cascaded EAMs configuration.

Although all of the above Tx designs have been sent for fabrication, the first two are considered the most mature and will be described in more detail below.

The following Rx PIC designs have also been finalized within OCTAPUS:

1. PDs followed by SOAs, with different variations in terms of SOA length.
2. PD with two cascaded SOAs to act as Preamplifier and Booster SOAs for higher dynamic range and PON splitting ratios.
3. Balanced PDs and SOAs, targeting burst-mode operation.

To achieve the aforementioned low-power, 50 Gbps NRZ operation, tailored EIC driver and TIA designs have also been carried out within OCTAPUS for the Tx and Rx, respectively, to convert the input (differential voltage) signal into a sufficiently large voltage signal to drive the Tx EAM, and convert the low current at the output of the high-speed PD into a (differential) output voltage with sufficient swing. Note that, although the driver output is differential, only one output will be used to drive the EAM thus allowing for fully isolated EAM terminal, which can result in higher available drive. The following subsection contain more details on the specifications of the photonic and electronic ICs on the Tx and Rx sides.

4.1.1 Optical transceiver and electronic driver/TIA IC specifications

The targeted specifications for the InP-based Tx and Rx components are listed in Table 4.1. To meet these specifications, the basic design parameters for the first two Tx chip designs, i.e., DBR laser followed by EAM, and DBR laser followed by EAM and Booster SOA, respectively, are provided in Table 4.2 and Table 4.3, for different wavelengths in the range 1330-1345 nm.

Table 4.1: Target specifications for InP Tx and Rx components.

Components	Parameters	Values
EAM	Extinction ratio (dB)	> 8
	Voltage swing (V)	3-4
	Bias point (V)	1.5-2
DBR laser + SOAs	Output power (dBm)	> 6
	SOA current (mA)	< 80
	SOA voltage (V)	< 2.5
High-speed PD	Dark current (nA)	< 50
	Bias voltage (V)	-5
MMI	Excess loss (dB)	0.5-1
Waveguide	Loss (dB/cm)	< 5
	90° bend loss (dB)	< 0.1 (for 100 μm radius)
	Transition loss (deep to shallow)	< 0.2 dB/transition

Table 4.2: DBR laser + EAM TX design specifications.

Transmission Wavelength (nm)	DBR front	DBR rear	SOA	EAM
1330	Pitch: 0.2068 Length: 200 μm	Pitch: 0.2068 Length: 500 μm	Length: 400 μm	Length: 100 μm
1335	Pitch: 0.2077 Length: 200 μm	Pitch: 0.2077 Length: 500 μm	Length: 400 μm	Length: 100 μm
1340	Pitch: 0.2086 Length: 200 μm	Pitch: 0.2086 Length: 500 μm	Length: 400 μm	Length: 100 μm
1345	Pitch: 0.2095 Length: 200 μm	Pitch: 0.2095 Length: 500 μm	Length: 400 μm	Length: 100 μm

Table 4.3: DBR laser + EAM + SOA Tx design specifications.

Transmission Wavelength (nm)	DBR front	DBR rear	SOA	Booster SOA	EAM

1338	Pitch: 0.2083 µm Length: 200 µm	Pitch: 0.2083 µm Length: 500 µm	Length: 400 µm	Length: 700 µm	Length: 100 µm
1339	Pitch: 0.2085 µm Length: 200 µm	Pitch: 0.2085 µm Length: 500 µm	Length: 400 µm	Length: 700 µm	Length: 100 µm
1340	Pitch: 0.2086 µm Length: 200 µm	Pitch: 0.2086 µm Length: 500 µm	Length: 400 µm	Length: 700 µm	Length: 100 µm
1341	Pitch: 0.2088 µm Length: 200 µm	Pitch: 0.2088 µm Length: 500 µm	Length: 400 µm	Length: 700 µm	Length: 100 µm

The top-level specifications for the Tx driver EICs are presented in Table 4.4. To meet these specifications, the final EIC design parameters for the Tx drivers are provided in Table 4.5, where a Continuous Time Linear Equalizer (CTLE) is included to mitigate any bandwidth limitations on the input side of the driver EICs' Radio Frequency (RF) connection.

Table 4.4: Top-level specifications for TX driver EICs.

Specification	Parameter/Value
Modulation format	NRZ
Baud rate	56 Gbps
Number of amplified channels	4
Process	TSMC 28 nm CMMOS
Connections to modulator	Differential
I/O ring and bondpads	125 µm pitch
Integration scheme	Wirebonding to EAM
Targeted energy efficiency (pJ/bit)	~0.5 (output driver incl. photocurrent monitoring) ~1.2 (full data path, excl. photocurrent modeling)

Table 4.5: Design parameters for Tx driver EICs.

Parameter	Unit	Value			Remarks
		MIN	TYP	MAX	
Output swing, single-ended	V		1.7		Depends on modulator characteristics
EAM bias voltage	V		3.0		Can be set via external pin, limited by breakdown voltage of decoupling capacitor
EAM absorption current	mA		5		Max absorption current generated by EAM
Gain	dB		10		
Rise and fall times	ps		5		
Input time CTLE peaking	dB	0	4	8	Digitally adjustable

Input side CTLE peaking freq.	GHz		25		Intended for 50-56 Gbps operations

The InP-based PD at the Rx side is a p-i-n diode, targeting O-band operation and being optimized for RF work conditions. Although measurement results indicate (see Deliverable D3.1 for more details) an Electro-optic bandwidth at 1310 nm of ~27 GHz for reverse bias values of -5V, a more extensive characterization of the RF PDs is planned to be performed on the 1st run of fabricated PD devices.

The top-level specifications for the Rx TIA EICs are presented in Table 4.6. To meet these specifications, the final EIC design parameters for the Rx TIAs are provided in Table 4.7

Table 4.6: Top-level specifications for Rx driver TIAs.

Specification	Parameter/Value
Detectors	InP photodiodes
Modulation format	NRZ
Baudrate	56 Gbps
Bandwidth	~ 28 GHz
Number of channels	4
Process	TSMC 28 nm CMOS
Connections to detector	Single-ended
Integration scheme	Wirebonding to photodetector
Input referred noise	< 4μA (rms)
Gain	> 65 dBΩ
Targeted energy efficiency	< 2 pJ/bit

Table 4.7: Design parameters for Rx side TIAs.

Parameter	Unit	Value			Remarks
		MIN	TYP	MAX	
Output swing, differential	mV		500		To be re-evaluated, if necessary.
PD cathode bias voltage	V		3		Can be set via external pin, limited by breakdown voltage of decoupling capacitor
PD anode bias voltage	V		0.5		Depends on exact design
Gain	dBΩ Ω		70 3000		

Bandwidth	GHz		35		
Input-referred noise current	μA (rms)		2		Initial target, integrated over bandwidth
Output CTLE peaking	dB	2	5	8	Digitally adjustable
Output CTLE peaking freq.	GHz		25		Intended for 50-56 Gbps operation

4.2 OCS-based NGCO backplane

4.2.1 2x2 PCM cell unit

As shown in Fig. 3.1 of Chapter 3, the OCS backplane has a modular structure consisting of multiple 2x2 PCM cell units. This unit is shown in Fig. 4.1 below, along with the associated switch states/configurations, hereafter referred to as "straight" vs. "cross" for States A and B, respectively, of Fig. 4.1. These units are based on the Silicon Nitride (SiN) and Silicon-on Insulator (SOI) platforms and target operation both in the O-band (for intra-NGCO links) and the C-band (for the optical Express Paths). The specifications of the individual passive elements comprising the PCM-based switches are provided in Table 4.8 and Table 4.9, respectively, for both platforms.

Fig. 4.1 Schematic of 2x2 MZI with two phase shifters in each arm and the associated states.

Table 4.8: Specifications of the passive elements on the SiN platform.

Passive LPCVD SiN platform (layer stack: 2μm-SiO_2/400nm-SiN/3.2μm-SiO_2/Si-substrate)					
Components	**Parameters**	**TE**		**TM**	
		O band	**C band**	**O band**	**C band**
Grating couplers	Coupling angle (degree)	16	15	21	25
	Central wavelength (nm)	1308	1567	1307	1581
	Max. coupling efficiency (dB)	8.2	5.1	16.3	13
	1 dB bandwidth (nm)	22	59	68	64
Strip waveguide	Width (μm)	0.7	1.0	0.7	1.0
	Insertion loss (dB)	0.44	0.35	0.59	0.35
Waveguide bend	Radius (μm)	60	60	60	60

	Insertion loss (dB/unit)	0.02	0.02	0.03	0.02
Waveguide cross	Insertion loss (dB/unit)	0.14	0.23	0.27	0.21
	Crosstalk (dB/unit)	-39	-42	-42	-44
MMI (1x2)	Insertion loss (dB/unit)	0.11	0.10	0.12	0.06
	Imbalance	n/a	n/a	n/a	n/a
MMI (2x2)	Insertion loss (dB/unit)	0.14	0.14	0.15	0.14
	Splitting ratio	n/a	n/a	n/a	n/a

Table 4.9: Specifications of the passive elements on the SOI platform.

Passive SOI platform (layer stack: 2μm SiO_2/220nm Si/2μm SiO_2/Si substrate)					
Components	**Parameters**	**O band**		**C band**	
Grating couplers	Coupling angle (degree)	10		10	
	Central wavelength (nm)	1300-1330		1550-1580	
	Coupling efficiency (dB)	5-6		5.5-6.5	
	1 dB bandwidth (nm)	> 30		> 35	
Strip waveguide	Width (nm)	400		400	
	Insertion loss (dB/cm)	2.5-3.5			
Waveguide bend	Radius (μm)	10	5	10	5
	Insertion loss (dB/unit)	0.01	0.02	0.01	0.03
Waveguide cross	Insertion loss (dB/unit)	0.2		0.12	
	Crosstalk	n/a		n/a	
MMI (1x2)	Insertion loss (dB/unit)	0.278		0.05	
	Splitting ratio (%)	50/50		50/50	
MMI (2x2)	Insertion loss (dB/unit)	n/a		0.33	
	Splitting ratio	n/a		46.69/46.05 (also accounting for insertion loss)	

The actuation of the 2x2 PCM cell unit is based on non-volatile phase shifters that rely on the modulation effect of Phase Change Materials positioned on the waveguide. The lattice state (amorphous or crystalline) of the PCM can be changed by an electrical heat pulse, provided by heaters. The refractive index difference obtained between the amorphous and crystalline state of the material leads to an effective index change of the modes propagating in the SiN waveguide. The non-volatile phase shifter design within OCTAPUS essentially exploits this mechanism to change the PCM lattice state using the Joule heating effect under different temperature conditions. To change the lattice state of PCM from amorphous to crystalline, the heater provides a 560K stable temperature for over 1000 μs. The PCM can also be changed back to amorphous phase by using a short (<10 μs) high temperature pulse and a cooling down process.

4.2.2 Controlling the PCM cell units

As previously mentioned, the operation of the PCM-based phase shifters crucially depends on the excitation of a proper electrical heat pulse. Based on thermal simulations (presented in detail in D3.1 [3]), the switching voltage waveform for the PCM-based phase shifters with ITO based heater using the SiN platform should have the properties presented in Table 4.10.

Table 4.10: Switching voltage pulse amplitude and duration for PCM-based phase shifters.

PCM Phase Shifter	Crystallization	Amorphization
SiN–ITO heater (simulation)	3.6V, <1000μs pulse	6.5V, <1000μs pulse
SOI–ITO heater (simulation)	8V, 500ns pulse	14.5V, 500ns pulse
SiN–polysilicon heater (measured)	12V, 500μs pulse	30V, 1.2μs pulse

Using the aforementioned amorphization and crystallization voltage values for the ITO based heaters, Table 4.11 and Table 4.12 show the simulation-derived specifications of the PCM phase shifters for the SiN and SOI platforms, respectively, at different optical bands, which also influences the length of the ITO heaters and associated resistance, thus leading to the aforementioned control voltage values (the PCM index values used for the simulations in Table 4.12 are derived from measurements performed on the PCM layers after deposition and subsequent to crystallization using a hot plate).

Table 4.11: Simulation-derived specifications for PCM phase shifters on SiN.

Material thickness	Length for 2π phase shift (μm)	Insertion Loss for 2π phase shift (dB)	Effective index (Crystalline)	Effective index (Amorphous)
C-band SbSe	10	<0.05+ 0.7	1.96	1.80
C-band SbS	22	0.1+ 0.66	1.81	1.74
O-band SbSe	6.5	0.1 + 0.4	2.02	1.81
O-band SbS	17.5	0.1 + 0.4	1.82	1.74

Table 4.12: Simulation-derived specifications for PCM phase shifters on SOI.

Material thickness	Length for 2π phase shift (μm)	Insertion Loss for 2π phase shift (dB)	Effective index (Crystalline)	Effective index (Amorphous)
C-band SbSe	33	0.05 + 0.3	2.64	2.59
C-band SbS	52	<0.01 +1.5	2.59	2.56
O-band SbSe	30	0.08 + 3.1	2.86	2.82
O-band SbS	53	<0.01 + 0.7	2.82	2.79

4.3 Optical diplexers and interposers

The operation of the OCTAPUS Express paths crucially depends on the existence of a wavelength diplexer, which allows the separation of two wavelengths (in the C- and O- bands, respectively), with low losses, when they are injected into the same waveguide. Within OCTAPUS, three different diplexer designs, namely i) 2x2 couplers, ii) 1x2 MMIs, and iii) ring resonators, have been explored, with the 2x2 coupler design, depicted in Fig. 4.2 as a 2x2 evanescent coupler with two sine bends and a straight waveguide on each arm, emerging as the most promising design. The C- and O-band wavelengths are injected at the S1 and S2 input arms, respectively, shown in Fig. 4.2, with the objective of acquiring a high output ratio at the E1 arm.

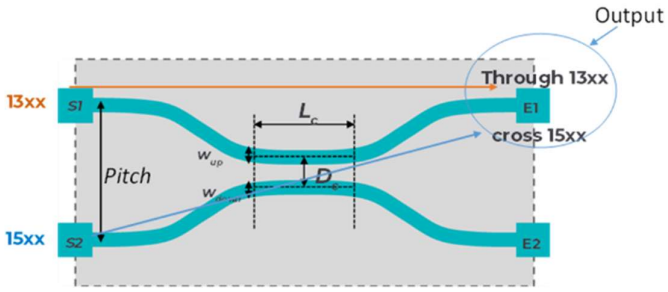


Fig. 4.2: The 2x2 coupler, as the main building block of the OCTAPUS wavelength diplexer.

TEEM has completed 4 groups of simulation and experimental measurement runs of the above design, with different parameter values for L_c, D_c in Fig. 4.2, as will be reported in Deliverable D3.2 (due M24), to determine that the optimal performance is achieved in both C/O bands for $L_c = 801$ μm and $D_c = 5$ μm. Optimal values for the other dimensions shown in Fig. 4.2 have also been selected as follows: W_{up}, W_{down} have been set to 2 μm, the pitch at each end of the coupler is 50 μm and the coupler arm bends have a radius of curvature equal to 2 mm.

For SiN waveguides in C/O bands, both edge coupling and evanescent coupling designs to a WAFT interposer (referred to as EC-WAFT and EV-WAFT, respectively), which is then connected to an optical fiber, have been completed, guided by simulations, as reported in Deliverable D3.1. However, the edge coupling has emerged as the more robust solution that is less susceptible to deviations from optimal parameter values. To this end, the specifications for the edge-coupled WAFT are provided in Table 4.13.

Table 4.13: EC-WAFT design specifications for coupling to SiN waveguides.

Parameter	Value
Coupling type	Edge coupling – taper on SiN
Sides	Single Sided – Right/East side PIC
Pitch Opt. Input/Output	127
Number of fiber channels 8x8 Switch 1x16 OCS	20 (8x In, 8x Out, 4x Alignment)
WAFT Glass Dimensions	6000 μm(length)x3556 μm(width) *thickness 1430 μm (no lid) + (800μm (max value) with lid)
Mode Field Diameter (MFD) on Glass	4.2 horizontal(μm) x 3.2 vertical (μm) @ 1550 nm (waveguide width = 2 μm (standard))

Edge coupling has also been chosen for coupling to InP waveguides, by using a Spot Size Converter (SSC) and waveguide pitch matching to offer seamless coupling compatible with any I/O configuration. The specifications of the EC-WAFT interposers are provided in Table 4.14 below.

Table 4.14: EC-WAFT design specifications for coupling to InP waveguides.

Parameter	Value
Coupling type	Edge coupling with Spot Size Converters
Sides	Single Sided – Right/East side PIC
Pitch Opt. Input/Output	250
Number of fiber channels	15
WAFT Glass Dimensions	6000 μm(length)x5500 μm(width) *thickness 1430 μm (no lid) + (800μm (max value) with lid)
Mode Field Diameter (MFD) on Glass	4.2 horizontal(μm) x 3.2 vertical (μm) @ 1550 nm (waveguide width = 2 μm (standard))

4.4 TSN data plane

Although Ethernet is widely used in data centers and Central Offices and is quickly gaining ground for mobile transport networks via the Ethernet-based eCPRI protocol, it is essentially a best effort protocol that does not provide any guarantees for traffic delivery or E2E latency (other than some basic network separation via suitable VLAN tagging). Since the targeted mobile X-haul and URLLC use cases for the OCTAPUS NGCO (see Deliverable D2.1 [1]) also involve time-sensitive traffic with strict latency QoS requirements, the OCTAPUS NGCO Uplink switches and IF cards, shown in Fig. 3.1 of Chapter 3, are equipped with specific TSN functionality and features to meet these requirements, essentially building on top of Ethernet switching and bridging to implement new traffic control mechanisms. The TSN features currently under implementation in the Uplink switches and IF cards of the OCTAPUS Data plane include:

- Time synchronization among all TSN devices, as the baseline feature upon which more advanced features are built. Synchronization is achieved via the IEEE 802.1AS-2020 [5] Generalized Precision Time Protocol (gPTP), which is a TSN profile of the IEEE 1588 Precision Time Protocol (PTP). PTP relies upon the notion of a Clock Master (CM), whereby multiple CMs can exist in a TSN network but one of them must act as the Grand Master (GM) clock, i.e., a reliable time source distributing accurate timing information throughout the TSN network via PTP. Since current Ethernet switched networks consist exclusively of full duplex Point-to-Point links, synchronization among two neighboring nodes is achieved by having two PTP clocks, one of them acting as the Clock Master and the other as the Clock Slave (CS).
- The Time-Aware Shaper (TAS), as defined in IEEE 802.1Qbv [6], is a scheduling mechanism that exploits the above (g)PTP-achieved synchronization to create Time Division Multiple Access (TDMA)-like reservation schemes, which provide exclusive access for a certain time duration to specified traffic classes within a schedule, thus ensuring bandwidth utilization matches offered load and latency requirements are met. This TSN schedule, which operates in periodic cycles until it is reconfigured on demand by the CNC controller (typically, upon the request for serving a new TSN flow), acts as a "red light/green light" traffic control mechanism for the entire TSN network (including its traffic end-points) and allows coexistence of different traffic classes with distinct QoS characteristics.
- Frame preemption, as defined in IEEE 802.1Qbu [7], is a mechanism to reduce head-of-line blocking for (and, thus, improve latency of) more "urgent" Ethernet frames, whereupon two Ethernet interfaces are created in the MAC layer for express and preemptable traffic (eMAC and pMAC respectively), where preemptable frames may be interrupted by express frames, under certain timing and frame length conditions, and resume their transmission once the express frame's transmission is complete. We note that preemption can be combined with the TAS shaper to reduce the negative impact of the guard band(s) between schedules/transmission windows and increase the utilization of the medium.

In light of the above, and particularly the TAS operation, the careful coordination of frame scheduling and routing in a multi-hop Ethernet network, i.e., the creation of the schedule itself by the CNC, is a

crucial issue, as it entails the entire TSN network topology (and the potential effects of OCS reconfiguration) and requires accurate and up-to-date knowledge of the available network resources. This necessitates a multi-part resource management procedure, which requires the participation of all major OCTAPUS Control plane entities, under the coordination of the SDN-based multi-domain coordinator, as explained in the next Section.

4.5 OCTAPUS SDN-based multi-domain control plane

The main functional blocks of the OCTAPUS NGCO Control plane have been described in Deliverable D2.2 [2] and the corresponding technical details reported in Deliverable D4.2 [4]. Specifically, D2.2 provides the system-level description of the NGCO Control Plane, while D4.2 focuses on the technical aspects such as the software design, implementation, and initial report of the early results. This section aims at reporting the final system-level architecture of the NGCO Control plane, first by summarizing the placement and role played by the NGCO Control plane within the OCTAPUS ecosystem, and then by explaining the minor architectural changes compared to what was reported in Deliverable D2.2. Finally, some details related to the main interactions among the NGCO Control Plane functional elements are also reported.

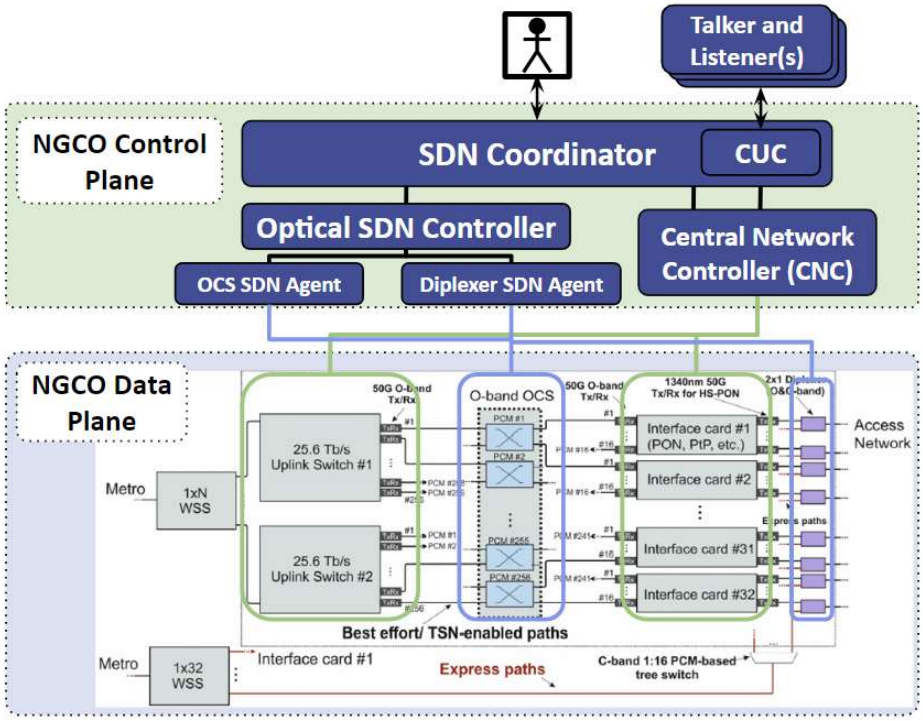


Fig. 4.3: NGCO Control Plane and Data Plane functional architecture.

Fig. 4.3 above is a verbatim copy of Fig. 3.1 (for the reader's convenience) and depicts the NGCO Control Plane and Data Plane functional architecture and their main relationships. The hierarchical structure of the NGCO Control plane is motivated by the different technological domains managed within the NGCO, i.e., the TSN domain and the Optical hardware (OCS, diplexer) domain.

Specifically, the CNC and CUC functional blocks are in charge of controlling the TSN Data Plane, i.e., the Interface Cards and the Uplink switches, by configuring and managing the TSN lifecycle streams within the NGCO.

Compared to the original functional architecture of the NGCO Control Plane reported in D2.2, a minor change implemented since then relates to the Optical Control Plane; the latter now manages and configures not only the OCS backplane within the NGCO, but also the diplexers bypassing the NGCO itself and redirecting the traffic through the Express paths. To this end, a new SDN Agent has been added from the architectural perspective, specifically for the management and control operation on the diplexer. This small architectural change in the NGCO Control Plane now allows the operator (or infrastructure owner) to programmatically turn off the NGCO, or set it in low power mode, when the traffic

has been steered to the Express path, potentially leading to significant energy savings. This operation aligns with the night usage scenario described in Deliverable D2.1, where a single NGCOs suffices to serve the (low) amount of demanded traffic, so that any traffic going through the other NGCOs can be bypassed to this single NGCO, allowing to turn off or powering down the other NGCOs while still meeting the QoS requirements.

On top of both control domains, the SDN Coordinator is the central entity in charge of the whole NGCO Control Plane by relying on the functionalities offered by the Optical and TSN Control Plane. Therefore, the SDN Coordinator has a global and complete view of the NGCO's status, by merging and managing the information coming from both domains.

As already reported in Deliverable D4.2 [4], each functional element of the NGCO Control Plane (i.e., the Optical Control Plane, CNC controller and SDN Coordinator) can be decomposed in smaller functional blocks, with each one of them having a specific role in the overall NGCO Control Plane architecture.

Fig. 4.4 depicts the internal blocks of the Optical SDN Controller, whose roles are summarized below.

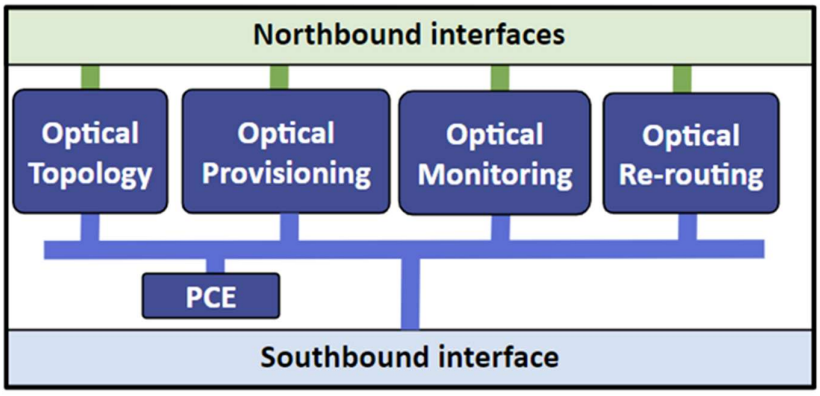


Fig. 4.4: OCTAPUS NGCO Optical Control Plane internal functional blocks.

- Optical Topology: it manages information about the Optical data plane elements (i.e., OCS, diplexers), such as their status and logical connections
- Optical Provisioning: it manages resources for optical switching of the traffic, by controlling the OCS backplane and the diplexer.
- Optical Monitoring: it collects and stores metrics from the OCS backplane and diplexer.
- PCE (Path Computation Engine): it computes the path within the Optical domain.
- Optical Re-routing: its role is to reconfigure the already established optical paths serving the scheduled traffic, aiming at more efficient resource utilization or load balancing.
- NBI (Northbound Interface): this interface exposes the SDN functionalities of the Optical domain or specific control hooks for the Optical domain to the SDN Coordinator.
- SBI (Southbound Interface): this interface interacts with the OCS backplane and Diplexer SDN Agents to communicate the control decisions made by the optical SDN controller.

The functional elements described above communicate with each other through a common bus (blue line in Fig. 4.4) for exposing and requesting the functionalities of the respective building blocks.

The internal decomposition of the SDN Coordinator is depicted in Fig. 4.5.

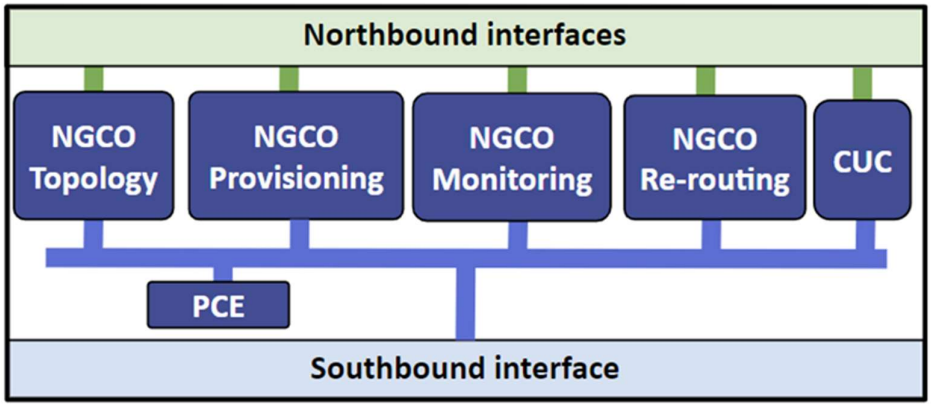


Fig. 4.5: Functional architecture of the SDN multi-domain Coordinator.

The intrinsic functional elements of the SDN Coordinator, as well as the internal and external interfaces for enabling interactions within the SDN Coordinator and across the OCTAPUS NGCO Control plane, respectively, are described next. Comparing Fig. 4.4 and Fig. 4.5 reveals a similar internal substructure for both the Optical SDN controller and the SDN multi-domain Coordinator; however, the latter has a "higher-level" network view and is responsible for the entire NGCO operation.

- NBI: this interface exposes the SDN Coordinator's functionalities to either a network operator or a higher layer network orchestrator.
- NGCO Topology: it manages the information about the NGCO data plane elements, such as their status and logical connections.
- NGCO Provisioning: it manages the NGCO resources for scheduling TSN and Best Effort network traffic.
- NGCO Monitoring: it collects and stores metrics (e.g., bandwidth utilization) from the NGCO.
- NGCO Re-routing: functional element whose main role is to redirect the scheduled traffic with the aim of more efficient resource utilization
- PCE: functional element for computing the path within the NGCO. It is crucial for deciding where the resources should be scheduled for the traffic
- CUC: a controller in the TSN domain that accepts high-level requests for TSN flows from the respective TSN end-points and translates these requests into concrete QoS requirements (in terms of throughput or E2E latency).
- SBI: this interface interacts with the domain-specific SDN Controllers, i.e., the optical SDN controller and TSN CNC controller.

Despite the internal Optical SDN Controller and the SDN Coordinator functional architectures looking similar to each other, it must be underlined that these two entities operate at different network domains. The former operates at the NGCO level, having a complete view of it and being responsible for its overall management and control. The latter operates at the more "limited" optical network level, interacting primarily with the OCS and the diplexer. For example, the NGCO PCE is responsible for computing the path within the NGCO and is supported by the Optical PCE, which computes the path at the optical network level only. With this hierarchical structure, any update or upgrade in the optical network equipment or optical backplane technologu is reflected at the specific Optical Control Plane level, without affecting the general NGCO Control Plane functionalities.

All the functional elements of the SDN Coordinator communicate with each other through a common bus (blue line in Fig. 4.5) for sharing their functionalities among them.

With the above brief summary of the functional architecture of the OCTAPUS NGCO Control plane, we next introduce the NGCO network intelligence.

4.5.1 NGCO Network intelligence

The NGCO Network Intelligence entails the management of TSN flows within the NGCO (i.e., ensuring the proper allocation of network resources within the NGCO to meet the requested QoS), as well as the handling of non-TSN flows (also termed "Best effort" traffic) via the remaining network resources. Another aspect of the NGCO network intelligence is the load-balancing of the NGCO resources, to be explained soon. As described in Deliverable D2.1 [1], the OCTAPUS NGCO is placed at a crucial point

of the end-to-end network architecture, aggregating the traffic from all downstream subscribers and Base Stations or Radio Units (RU) of disaggregated RANs to/from the metro network.

Considering the high-dynamicity of the network flows and traffic demands (as analyzed in Deliverable D2.1), load-balancing among the Uplink switches of the OCTAPUS NGCO is an important quantitative requirement for the NGCO network intelligence. The high-level sequence diagram depicted in Fig. 4.6 shows how the aptly named “make-before-break” approach can be applied to manage traffic in the NGCO and, at the same time, perform load balancing with its available resources. For simplicity, the CUC functional element is embedded within the SDN Coordinator and the Optical SDN Controller and SDN Agent are included in the “Optical_Domain” block within the diagram.

The high-level sequence diagram is composed of two stages, i.e., an initial setup stage and a subsequent operational stage. It must be noted that, from the TSN Control Plane perspective, the OCS backplane and its PCM switches are completely transparent to the CNC, since they are purely physical layer devices that do not support any of the TSN protocols per se, and are not part of the TSN topology. Since the SDN Coordinator is aware of the overall NGCO topology, during the setup stage, the SDN Coordinator informs the CNC about the topology (step 1 and step 2 in Fig. 4.6). At this point, even if the CNC has limited view of the NGCO Topology (being unaware of the OCS configuration), it is fully cognizant of how the TSN data plane devices (i.e., Uplink switches and IF cards) are connected to each other.

After the setup stage, the operational stage is executed whenever it is required to manage one or more TSN or Best Effort flows. For the TSN flows, the requests can be made (as per the IEEE802.1Qdj standard) by either the TSN flow end-points, referred to as Talker and Listener(s), or by a network operator. In Fig. 4.6, it is assumed that the request is made by the Talker and Listener(s) (step 3).

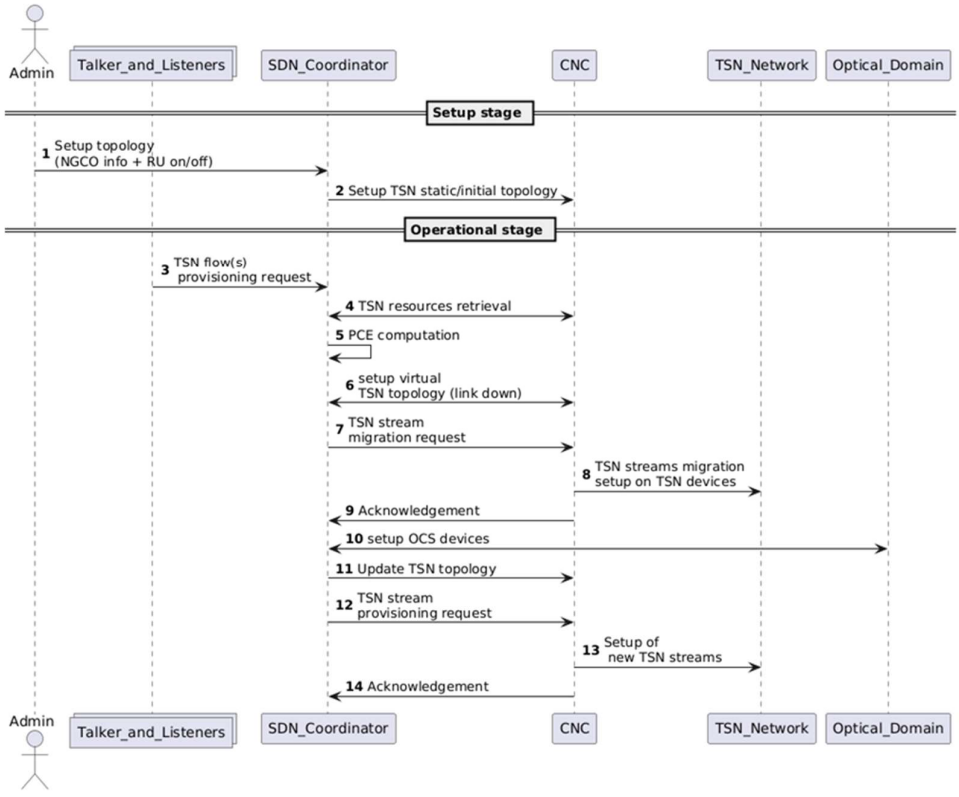


Fig. 4.6: High-level sequence diagram for TSN traffic management.

Regardless of the request’s origin, the SDN Coordinator checks the request’s feasibility against the available NGCO resources (i.e., the resources which have not yet been reserved by the CNC) and computes the path within the NGCO (step 4 and 5). In the sequence diagram of Fig. 4.6, it is assumed that the PCM configuration in the OCS is such that the available NGCO resources are somehow

"scattered" along the different available intra-NGCO links and one or more PCMs must change their configured state (from "straight" to "cross" or vice versa) as described in Section 4.2.2. Without additional measures, this change would disrupt the TSN traffic flow (and the underlying service) running on the specific segment.

Therefore, in the "make-before-break" procedure included in the workflow of Fig. 4.6, information on a virtual TSN topology is first sent to the CNC (step 6), so that the latter is aware of the intended change of the TSN topology (without the actual change taking place ***yet***). The flows running on the particular IF card-to-Uplink switch links that will be torn down by the impending OCS reconfiguration are requested to be moved to another path that will be valid in the ***new*** topology (step 7 and 8), so that the corresponding services will already have available resources for them under the new topology and no disruption will occur. Finally, after a positive acknowledgment from the CNC (step 9), the SDN Coordinator instructs the Optical SDN Controller to perform the actual OCS reconfiguration in the PCMs (step 10). In this way, resources are freed, and the requests for provisioning one or more TSN flows can be satisfied. From a CNC perspective, this is a TSN topology change and, for this reason, at step 11 the updated topology is sent to the CNC. Finally, at step 13 and 14 the requested TSN flow is provisioned within the NGCO and an acknowledgement is sent back to the SDN Coordinator. The advantage of the "make-before-break" is twofold: it allows to find the resources in a NGCO segment if these are "scattered" and it allows the load balancing of the resources. These advantages allow a more efficient usage of resources within the NGCO.

It is important to note that after the provisioning of one or more TSN flows, the SDN Coordinator can collect and store metrics related to the NGCO's status and performance, which are valuable in a highly dynamic environment where different network flows are served by the NGCO. This allows the NGCO Control Plane to take counteractions to either mitigate or solves issues such as performance degradation, unbalanced resource consumption or any other unexpected events. The NGCO Monitoring block in Fig. 4.5 helps in identifying possible issues, while the NGCO Re-routing, Provisioning, and PCE blocks in Fig. 4.5 help to apply the desired counter actions to resolve these issues.

Fig. 4.7 depicts a high-level sequence diagram for the automatic traffic re-routing in the NGCO. The diagram shows the workflow for setting up one or more policy rules conditioned on certain NGCO performance metrics or status indicators (selected by the policy maker), as well as the corresponding actions to be applied if these policy conditions are met.

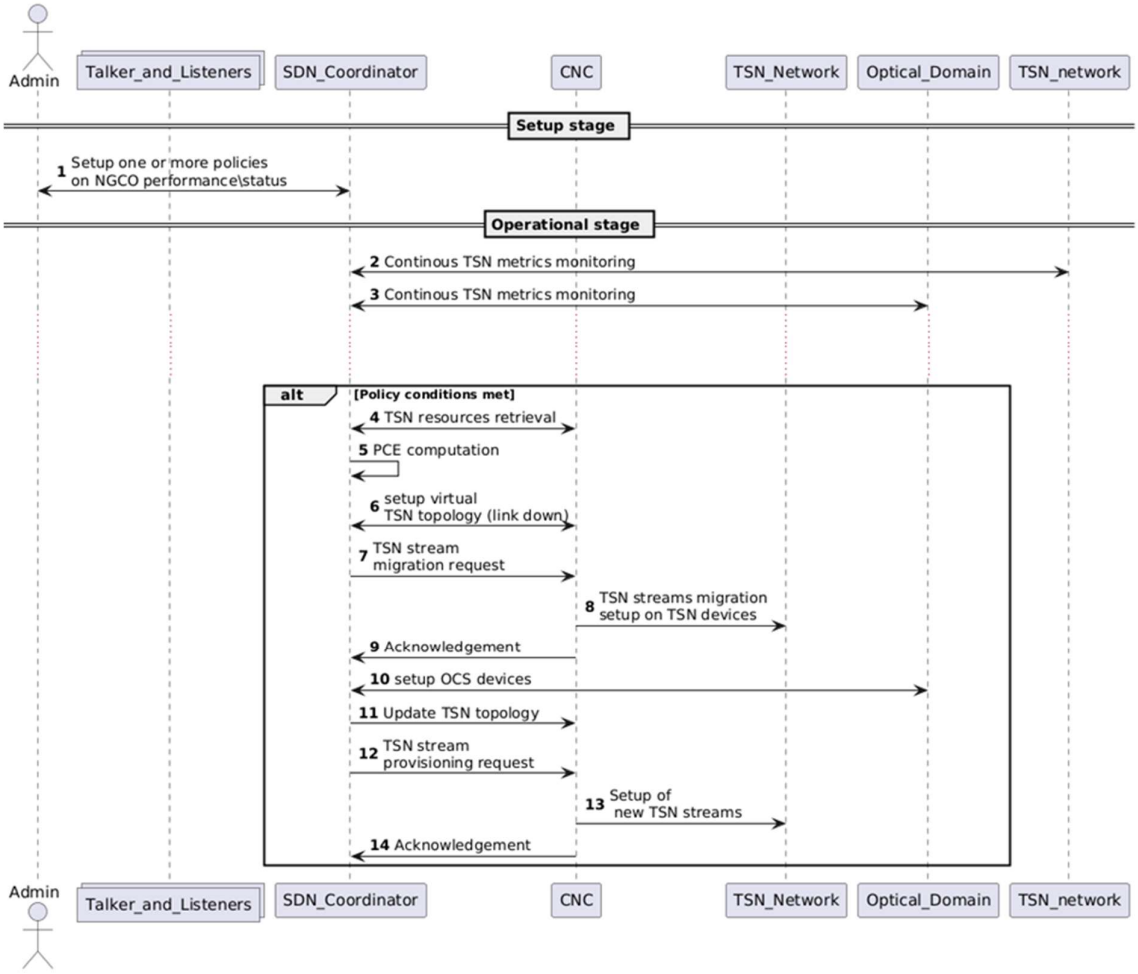


Fig. 4.7:High-level workflow for NGCO traffic re-routing.

At step 1 of Fig. 4.7, the setup stage includes the installation by the (network) administrator of one or more policies, specifying the conditions to be checked and the automatic counteractions to be taken when these conditions are not met. The monitoring can extend to both the optical and TSN control domains (steps 2 and step 3). To provide an example, a policy can consist of monitoring the traffic balance between two NGCO Uplink switches and if the switches are unbalanced according to a determined threshold (i.e., the condition to be checked), then specific counteractions are automatically applied. One of these actions can be the “make-before-break” procedure for re-routing and, thus, balancing the traffic between the Uplink switches (from step 4 to step 14). This allows the automatic prevention of possible network traffic congestion in one of the Uplink switches, where most of the traffic is going under an unbalanced resource allocation scenario. The above policy can be updated according to the high-level operator requirements, setting either more relaxed or stricter conditions to be checked for initiating the automatic re-routing procedure.

5. Modeling and evaluation of the OCTAPUS architecture

5.1 OCTAPUS Tx/Rx PHY-layer evaluation for intra-NGCO links

The effect of crosstalk and loss on the OCTAPUS intra-NGCO link performance is investigated, with the focus being placed on 4x4 and 8x8 Benes topologies using abstract models of the basic 2x2 switching elements interconnected, as shown in Fig. 5.1 and Fig. 5.2.

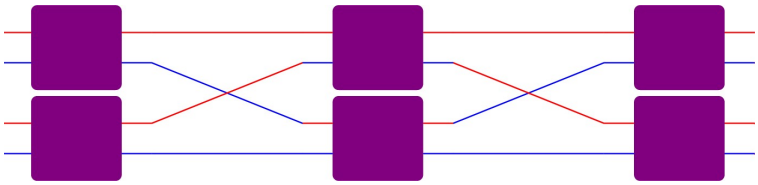


Fig. 5.1: 4x4 Benes network.

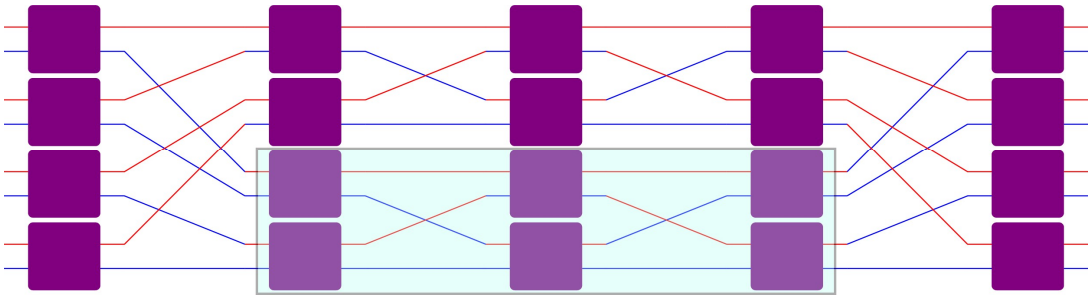


Fig. 5.2: 8x8 Benes network.

The above topologies are based on 2x2 switching elements in order be used as a benchmark for comparing them to the GMZI-based OCS concepts proposed in OCTAPUS and to be presented as soon as measurements of first samples are available. The rationale is to investigate the maximum performance degradation, in terms of crosstalk and insertion losses (IL), caused by an OCS module for a given reference link operating at 50 Gbps (net bit rate) and with transmitter and receiver specs compatible with the OCTAPUS targets and estimated metrics.

The transmitter and receiver impairments (such as laser Relative Intensity Noise (RIN), electronics noise, optical losses, bandwidth limitations etc.) of the reference link are inferred by the IEEE specifications for 53.125 GBaud PAM4 single-mode optical links. To match these specifications with OCTAPUS, the modulation format was adjusted to NRZ. The reference metric is the estimated BER performance as the reference signal passes the OCS topology. The eye diagram of the reference optical link after the receiver analog frontend is shown in Fig. 5.3.

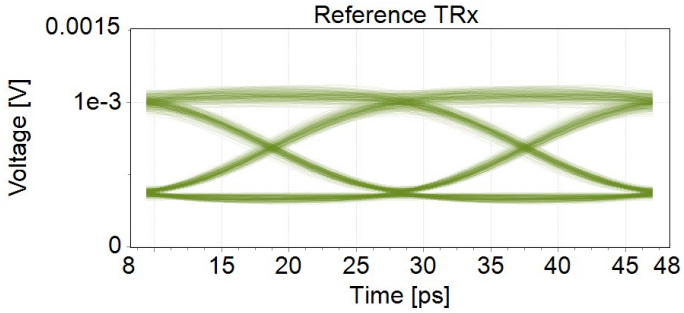


Fig. 5.3: Reference signal used in simulation.

The main crosstalk contributor is the Extinction Ratio (ER) of the 2x2 switching element. The cascade of those elements, combined with their insertion losses, create intra-band interference to the other paths of the OCS. The crosstalk XT is measured as:

$$XT = 10\log\left(\frac{\sum AOP_{aggresors}}{AOP_{victim}}\right) \quad [dB],$$

where

$$AOP_{victim} = AOP_{total} - \sum AOP_{aggresors},$$

AOP_{victim} is the average optical power on the receiver of the link under test without the contribution in power of the aggressors of the switch. In the cases shown below, for a NxN switching matrix, N-1 aggressors are considered for each victim path.

The following Figures present the estimated BER at the receiver as a function of OCS crosstalk and Average Optical Power (AOP) received. To evaluate the maximum tolerance on the parameters of loss and crosstalk, a sweep on the 2 parameters of the basic building block (IL and ER) of the switch is conducted. The range of the sweeps is 15-40 dB on the ER and 0.25-1.5 dB on the IL of each switching element.

Fig. 5.4 indicates that, in the case of the 4x4 Benes switch, the major factor that determines the signal integrity as it passes through the OCS is the extinction ratio of the fundamental switching unit, where the minimum acceptable ER value of 30 dB is required, reaching to 25 dB of total crosstalk in order to achieve the required BER target. It is also observed that the insertion loss of each element affects the overall signal integrity, however not that significantly as the ER. The respective values for the 8x8 Benes switch is 35 dB of ER on the switching element and 22 dB on total crosstalk, as shown in Fig. 5.5.

Fig. 5.6 depicts indicative eye diagrams of the received signals after passing through a 4x4 and 8x8 Benes switch-based OCS for the aforementioned parameters of the switching element.

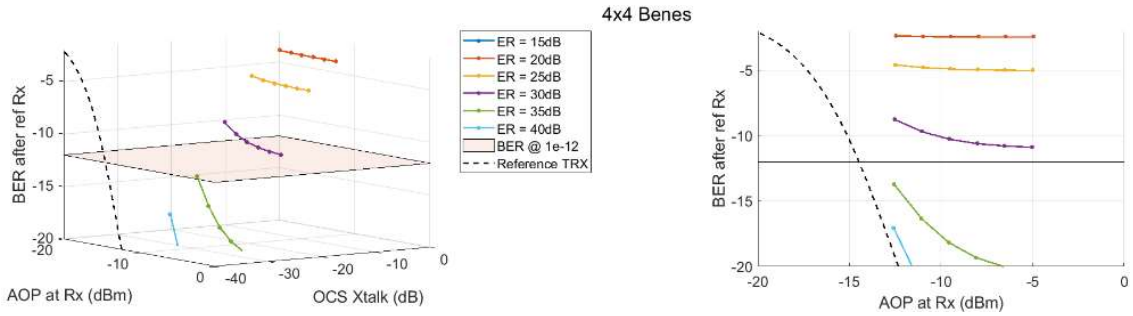


Fig. 5.4: Results of Monte Carlo simulation for 4x4 Benes topology.

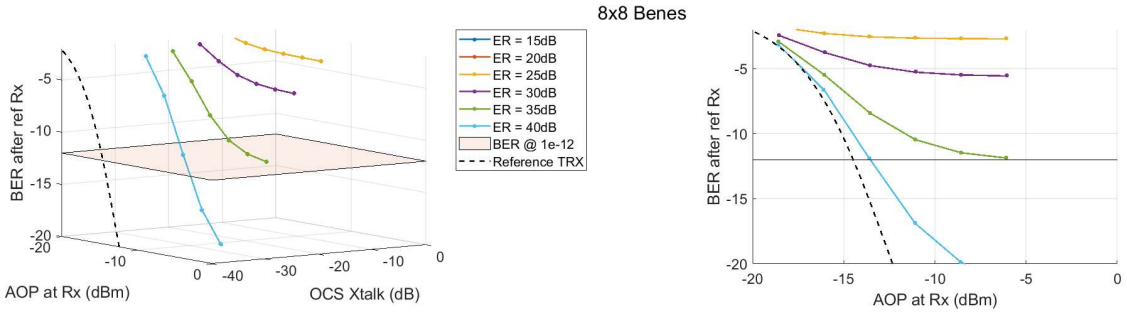
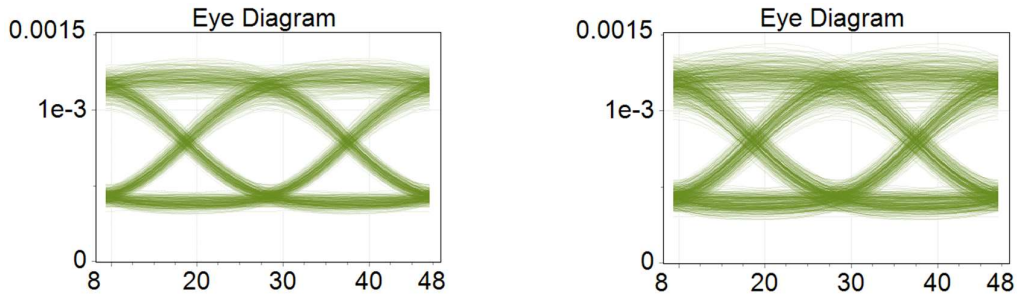


Fig. 5.5: Results of Monte Carlo simulation for 8x8 Benes topology.



(a) Received Signal after 4x4 switch [35 dB ER,1.0 dB IL per switching element]

(b) Received Signal after 8x8 switch [35 dB ER,1.0 dB IL per switching element]

Fig. 5.6: Eye diagrams of the received signals after passing through (a) 4x4 and (b) 8x8 OCS.

5.2 Discrete event-based evaluation framework for OCTAPUS architecture

The OCTAPUS discrete event-based evaluation framework offers a facility to simulate end-to-end (E2E) V2X communication bringing together realistic vehicle mobility and their networking connectivity. For the mobility layer, we exploit SUMO to define realistic mobility scenarios. The mobility layer offers realism to our simulation by incorporating dynamic elements of urban environments, e.g., traffic lights and intersections. We have employed OpenStreetMap (OSM) to replicate road layouts of the Angers city in France for which the precise coordinates of 5G BS are publicly available. We consider both mobile (i.e., vehicles) and static nodes (i.e., BS) and monitor their network connectivity in terms of channel and signal quality, as well as E2e delay, while vehicles are approaching and moving away from the BS. More detailed information can be found in Section 5.2.1 of Deliverable D2.2 [2].

For the network layer, we employ the OMNeT++ simulator by leveraging three specific libraries:

- the Veins library, specifically tailored for Vehicular Ad Hoc Networks (VANETs), to offer components such as vehicles, roadside units, and communication models.
- the Inet library to provide a diverse set of core network components, including nodes, devices, protocols, and applications.
- the Simu5G to bring 5G network functionality by integrating elements like user equipment, gNodeB (gNB) classes, a complete 5G NR protocol stack, and numerous 5G core network components.

The mobility+network layered architecture enables a thorough analysis of the interplay between mobility and network components, providing a holistic view of vehicular traffic generation and network behavior in smart cities, as outlined in Deliverable D2.2.

We utilize the Veins library to facilitate the seamless interaction between OMNeT++ and SUMO, as well as the Inet and Simu5G libraries to enable V2X communication with different technologies. We develop our framework to support either IEEE 802.11p or the 5G NR protocol stack (the latter via the Simu5G library) for vehicle communication facilitating the appropriate configurations for both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) scenarios. However, we focus exclusively on 5G V2I to simulate data exchange between vehicles (treated as User Equipment (UE) components) and BS (treated as gNB components) for radio signal transmission and reception. BS themselves are interconnected via Ethernet-based interface cards (IF cards), ensuring robust and high-speed wired communication (modelled by the Inet library). This configuration allows us to monitor vital 5G quality metrics such as SINR (Signal to Interference plus Noise Ratio) and CQI (Channel Quality Indicator), as well as evaluate E2E packet delay, within a 5G-enabled vehicular communication environment. The aforementioned network components are shown in Fig. 5.7.

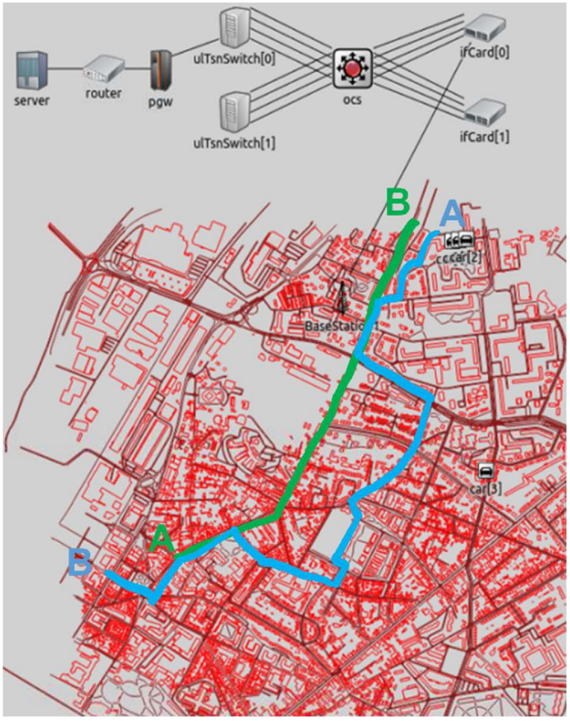


Fig. 5.7: End to end communication framework

5.2.1 Multiple-vehicle scenario

We define two scenarios, namely the multiple-vehicle and the two-vehicle scenario, to validate the functionality of our framework, i.e., enable E2E V2X communication, while considering the dynamics of urban environments. These scenarios are designed to address challenges, such as the impact of concurrent data transmissions on the E2E delay. Such cases are frequent during rush hours or unexpected events (e.g., traffic congestion due to a demonstration) where many vehicles are concentrated in a small geographical area, and become crucial when critical transmissions, e.g., those of Emergency vehicles, should be prioritized.

Following the parameterization described in detail in Deliverable D2.2, in the multiple-vehicle scenario, we deploy several vehicles belonging to all three classes Slow, Fast and Emergency, populated with different densities corresponding to Night, Typical or Busy hours. Vehicular density configurations within our framework are illustrated in Fig. 5.8, Fig. 5.9, Fig. 5.10, respectively. In each case, we monitor three distinct vehicles, one of each class, while they are moving along a predefined route marked in blue in Fig. 5.7. The vehicles begin at point A, near the BS, follow the blue route and finish at point B. The other vehicles are moving randomly across the entire road layout to simulate realistic conditions. In this way, we increase the complexity to observe the impact of traffic density on network delay, evaluate it with respect to the requirements posed by different applications (especially, delay-sensitive ones), and highlight the importance of developing appropriate mechanisms to offer quality of service. This scenario lasts for 10 minutes of simulation time. During this period, each vehicle sends 1 data packet per second to the central server in Fig. 5.7, with each packet being 1000 bytes long, until the vehicle ends its designated route. The BS operates on a component carrier frequency of 2 GHz with transmission power of 40 dBm, while vehicles also use a single component carrier for their communications.

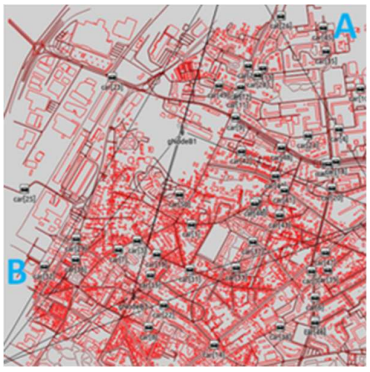


Fig. 5.8: Night hour.

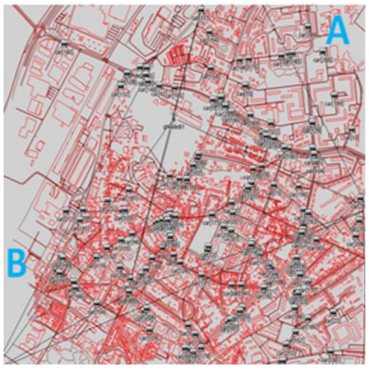


Fig. 5.9: Typical hour.

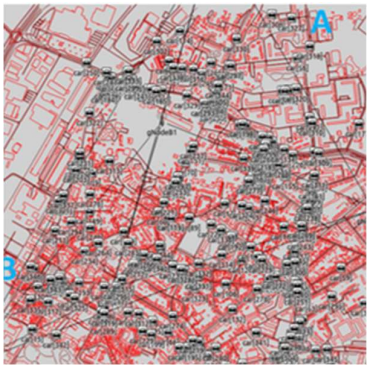


Fig. 5.10: Night hour.

5.2.2 Two-vehicle TSN scenario

In the two-vehicle TSN scenario, a simple road layout shown by the red vertical line in Fig. 5.11 is used to analyze the communication between two vehicles and two 5G base stations, each of them being equipped to handle network traffic generated by the vehicles. Both vehicles start from the top of the road and move towards the bottom, communicating with two 5G base stations (named BaseStation0 and BaseStation1 in Fig. 5.11). The Base Stations receive packets from the vehicles, ensuring real-time communication between them and the rest of the network.

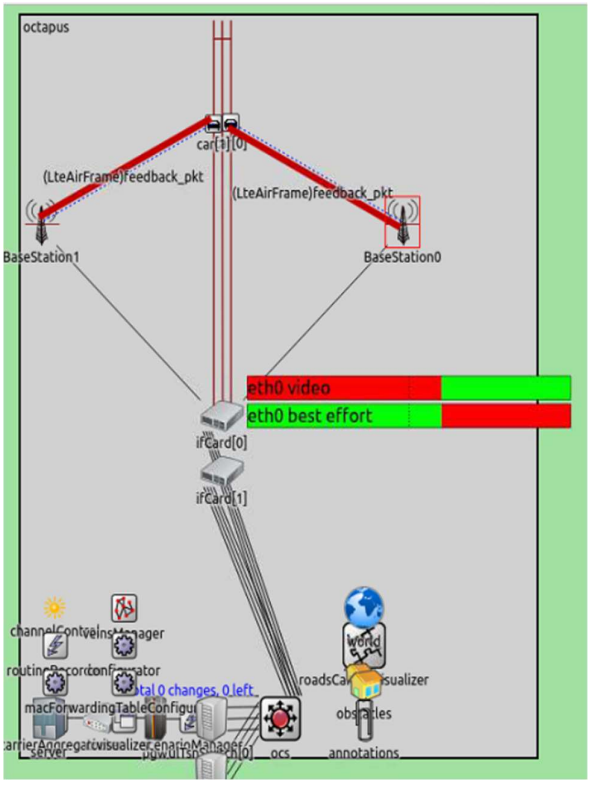


Fig. 5.11: Two-vehicle TSN scenario infrastructure.

The simulation aims to evaluate how the following two traffic shaping mechanisms impact delay between these vehicles and the 5G infrastructure:

- Time-Aware Shaper (TAS), also known as Time-Aware Scheduler.
- Credit-Based Shaper (CBS).

TAS functions by allocating distinct time slots (potentially, of variable duration) for high-priority traffic, such as the video feed from Car #1, ensuring that this traffic is transmitted with minimal interference or delay. In contrast, CBS uses a credit-based system to manage the transmission of traffic, allowing best-effort data from Car #0 to be sent dynamically, depending on the network's current load and the availability of resources. The goal of this analysis is to compare the performance of TAS and CBS in a 5G vehicular network scenario. By doing so, we seek to understand how these traffic shaping techniques influence the overall network performance, particularly in situations where high-priority video streaming must coexist with less critical Best Effort traffic. The results could provide valuable insights into how future 5G networks can be optimized for autonomous vehicles, ensuring that latency-sensitive applications such as video streaming are prioritized without severely impacting other types of data transmission.

5.2.3 Multiple-vehicle scenario results

The results of Fig. 5.12 and Fig. 5.13 refer to more realistic vehicular traffic conditions where the blue, orange, and grey colors correspond to representative Emergency, Fast, and Slow vehicles, respectively, which traverse the blue route in Fig. 5.7 from point A to point B. Fig. 5.12 shows the vehicles' speed over time for the three different traffic density conditions, namely Night, Typical, and Busy (from top to bottom). A warm-up simulation period of 30 s is necessary for the vehicles to be distributed and move within the town. Different speed patterns are produced both due to the vehicles' density and their maximum speed. For Night density conditions (top graph in Fig. 5.12) with as many as 25 vehicles per kilometres of road, Fast and Slow vehicles reach their maximum speed, i.e., 40 km/h and 20 km/h, respectively, while the speed of the representative Emergency vehicle does not exceed 58 km/h. The complete parameterization of the above densities and vehicular capabilities is available in Deliverable D2.2.

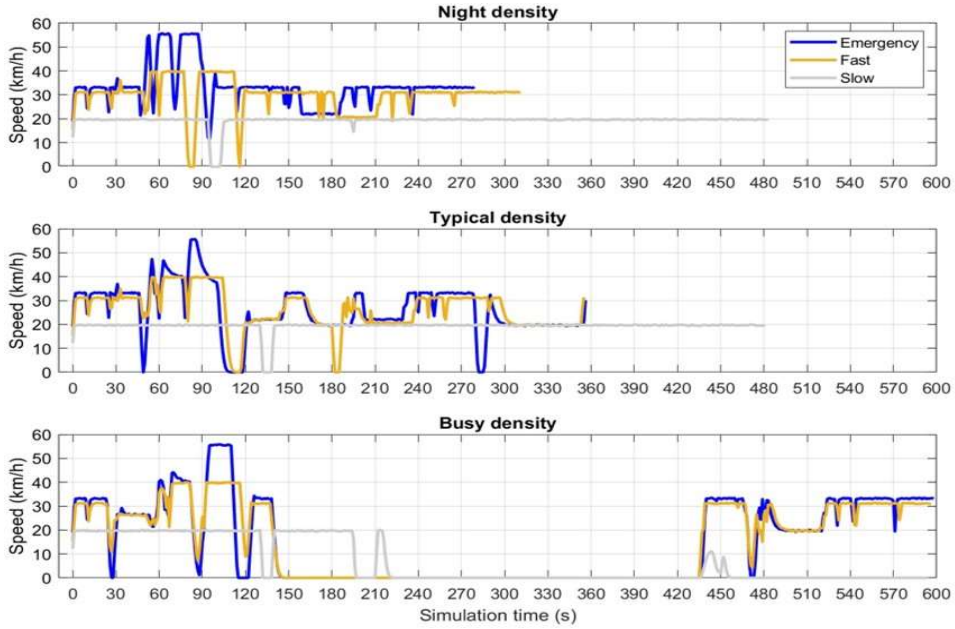


Fig. 5.12: Vehicles' speed over time for different vehicular densities.

Apart from the mobility results of Fig. 5.12, we derive network related metric. Specifically, the graphs of Fig. 5.13 illustrate the relationship between SINR (dashed lines) and E2E uplink delay (solid lines) across the different density scenarios, for three representative vehicles (one of each category).

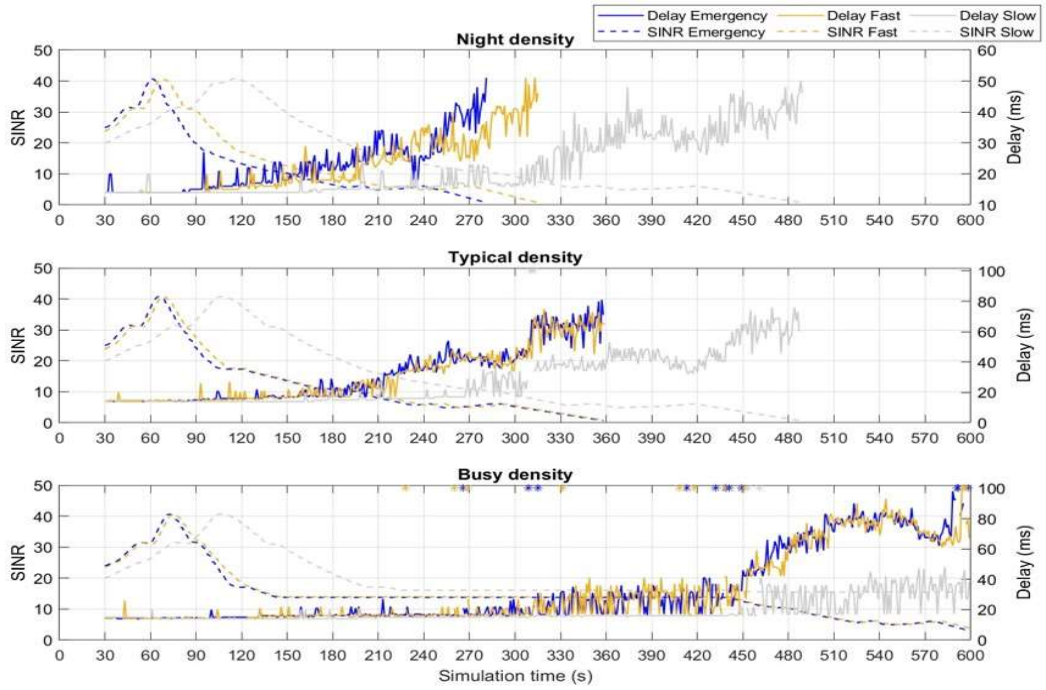


Fig. 5.13: SINR and delay of each vehicle class.

We also study how varying vehicular traffic densities impact the mean packet delay averaged over the vehicle's travel duration across different vehicle classes. In particular, Table 5.1 summarizes the mean delay, the standard deviation (SD), and the variance (σ^2) throughout the simulation scenarios along with the number of packets being sent in each case.

Table 5.1 Mean packet delay, SD and variance

Vehicle class	# of packets	Mean delay (ms)	SD (ms)	σ^2
Night density				
Emergency	252	22	8.35	70
Fast	286	22	9.06	82
Slow	460	22	9.8	96
Typical density				
Emergency	329	30	17.94	322
Fast	330	30	17.23	297
Slow	458	32	66.7	4449
Busy density				
Emergency	565	46	117.34	13771
Fast	567	46	117.21	13739
Slow	570	22	59.06	3489

Observing the SINR curves (left y-axis in Fig. 5.13) for different densities we infer that the vehicles' route starts near the BS, matching point A of the blue line in Fig. 5.7, and concludes far away, i.e., at point B. This explains the high SINR values at the beginning of the simulation run in all graphs, where the channel quality is also high, which progressively decreases and becomes minimum at the end of the route. For the Slow vehicle, in particular, the SINR reaches its peak about 40 s later, compared to the Emergency and Fast vehicles, due to its lower speed. For the same reason, the grey curve of the Slow vehicle is broader compared to the blue and orange ones for the Emergency and Fast vehicles, respectively. Another remark concerns the SINR curves of the Emergency and Fast vehicles, which become very similar as the density gets heavier; this confirms that the increased density negates the speed advantage. Heavy traffic is also responsible for the flat SINR values (bottom graph of Fig. 5.13) since the vehicles remain immobile for a long period.

Overall, we notice that the SINR and delay time series (the latter captured in the right y-axis in Fig. 5.13) are correlated, since high values in signal quality, e.g., proximity to the BS and high-quality channel, produce low delays, while signal quality deterioration leads to increases in delay. Apparently, the lighter the traffic (top graph in Fig. 5.13) the lower the E2E delay as indicated by the y-axis. In fact,

for Typical and Busy density, we observe, among others, significant delay spikes (denoted with the star points of the middle and bottom graphs in Fig. 5.13) that reach up to 1 s and are due to high congestion and BS's inability to quickly respond to the vehicle's communication requests. In addition, similar to the SINR curves, the delay curves of the Emergency and Fast vehicles get closer with the increase in the traffic density. However, we should notice that the low levels of delay for the Slow vehicle (bottom graph in Fig. 5.13) is caused by it getting stuck near the BS and, thus, maintaining reliable communication to it. While the delay curves of Fig. 5.13 show its evolution over time for one representative vehicle of each category, a more holistic picture is provided by the statistical data of Table 5.1. Our results show that, under light traffic, vehicles, regardless of their class, experience 22 ms mean delay, which combined with small SD and σ^2 meets the requirements of cutting-edge, safety-critical applications, such as intersection collision avoidance or traffic signal coordination, where a latency of less than 30 ms is necessary for timely decisions.

As anticipated, heavier traffic conditions lead to higher delays. For example, under Typical density, all vehicles experience delays of about 30 ms, while Busy conditions cause delay of 50 ms for high-speed vehicles. An exception is observed for the Slow vehicles under heavy traffic, whose low delay comes along with their inability to conclude their route due to getting stuck in traffic. The aforementioned levels of delay are affordable for a wide range of applications, such as forward collision warning, loss of control alerts and emergency notifications, which exhibit latency requirements of around 100 ms. Our results indicate that vehicles' speed by its own does not provide communication advantages, especially during the rush hours. Given that heavily congested conditions are the most challenging, it is apparent that we need mechanisms that can dynamically match the delay to the application requirements.

5.2.4 Two-vehicle TSN scenario results

In this scenario, the network configuration includes two server applications running on ports 3000 and 3001, respectively. The first car, Car 0, runs a "Best Effort" application that communicates with the server on port 3000, while the second car, Car 1, operates a video streaming application that connects through port 3001. This setup is designed to explore how different traffic types behave under specific network conditions and priorities. Both applications generate packets of 1000 bytes each, transmitted at a rate of 1000 packets per second. The experiment is conducted over 130 seconds, incorporating a 10-second warm-up period to allow the network to stabilize before collecting performance metrics. During the main 120-second measurement period, we focus on several critical metrics, including the cars' distance from Basestation0, E2E latency, incoming data rate and outgoing data rate. These metrics will reveal insights into the reliability and timeliness of packet delivery for each application type.

Fig. 5.14 illustrates the changing distances of the two cars from Basestation0 of Fig. 5.7 over time. The x-axis represents time, while the y-axis shows the distance from the base station, measured in meters. Initially, both cars maintain a relatively constant distance from the base station, with Car 0 positioned slightly farther than Car 1. Around the 40-second mark, both cars begin to approach the Base Station, reaching their closest proximity near the 80-second mark, where their distances from the base station are minimized. Following this point, the cars begin to increase their distance again, both of them moving away from the base station. By the end of the recorded time, around 130 seconds, the cars' distances have returned closer to their initial values, although Car 0 remains consistently farther than Car 1 throughout the simulation. The graph indicates synchronized movement between the two cars, due to similar driving patterns or coordinated vehicle routing in the experiment. The variation in distance from the base station reflects dynamic movement, simulating real-world scenarios where vehicles move toward and then away from a central point over time.

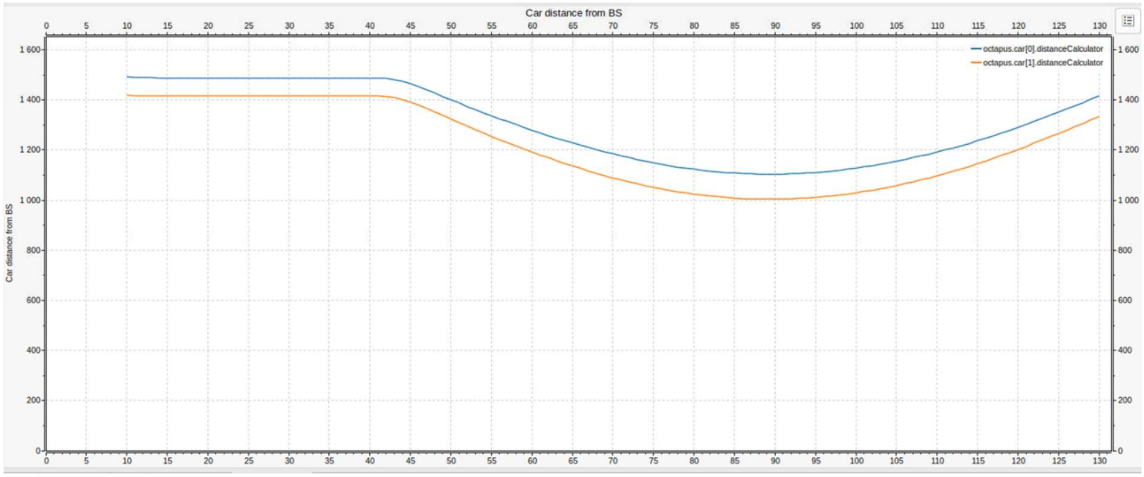


Fig. 5.14: Cars' distance from Basestation 0.

Obviously, any traffic shaping mechanisms, such as TAS or CBS, do not influence vehicles' movement (although they certainly affect other performance aspects, such as network delay). Hence, the trajectory associated to the cars' movement described in Fig. 5.14 holds regardless of whether any shapers are active or not.

5.2.4.1 Without shaper

Table 5.2 provides detailed statistical information on both incoming and outgoing data rates throughout the experiment. Each data rate category (incoming and outgoing) was sampled 2400 times, ensuring a robust analysis of average performance and variability. The mean data rate for both incoming and outgoing traffic is 16.3 Mbps, indicating balanced network usage where data flows into and out of the network at a similar rate on average. The standard deviation of 4.2 Mbps for both directions highlights the variability around the mean, reflecting moderate fluctuations in data rate due to dynamic network conditions.

Table 5.2: Data rate values without any shaper.

Name	Number of packets	Mean	Standard Deviation
Incoming data rate	2400	16.3 Mbps	4.2 Mbps
Outgoing data rate	2400	16.3 Mbps	4.2 Mbps

A graphical representation of these data rates over time is provided in Fig. 5.15, which visualizes the fluctuations and trends in the network's data throughput during the experiment. Notably, this experiment operates with a single queue configuration, as there are no active traffic shapers. This setup allows us to observe the unmodified data flow, providing a clear view of how data rates behave naturally without any prioritization or traffic-shaping mechanisms. Overall, the network demonstrates consistent average throughput with expected variability in instantaneous data rates, indicating a stable network performance under the current conditions.

Table 5.3 presents statistical information on the E2E delay experienced by two applications, named App 0 and App 1, throughout the experiment. With 116,196 measurements for App 0 and 116,045 for App 1, the dataset provides a comprehensive view of delay performance across both applications. The mean E2E delay is 0.75 seconds for App 0 and 0.73 seconds for App 1, indicating that, on average, data packets for both applications reach their destination in a similar timeframe, with only a slight difference between them. The standard deviations of 1.038 seconds for App 0 and 1.019 seconds for App 1 reveal significant variability around the mean delay, suggesting that, although the average delays are relatively low, individual packet delays can vary widely, likely due to network congestion or varying traffic loads.

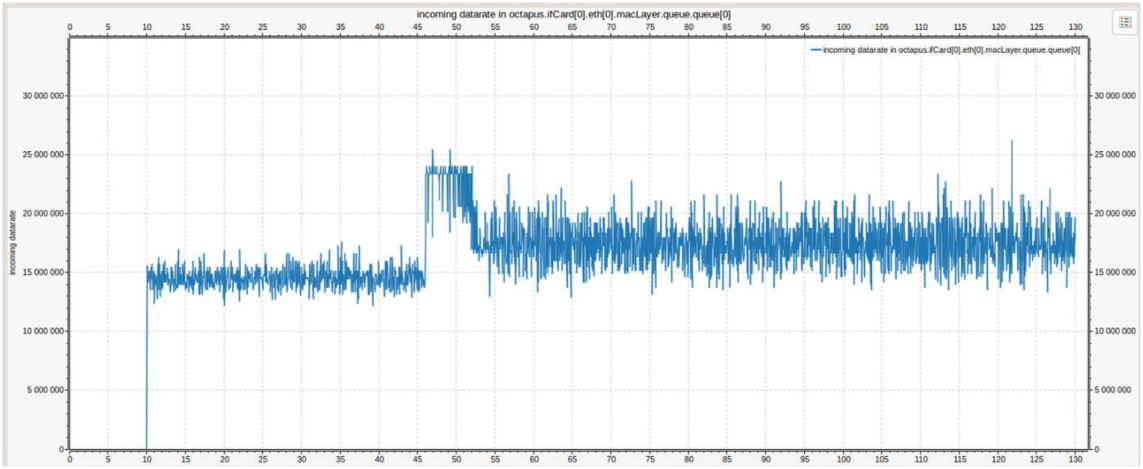


Fig. 5.15: Incoming and outgoing data rate without any shapers.

Table 5.3: End to end delay values without shaper.

Name	Number of packets	Mean	Standard Deviation
End to end delay App 0	116,196	0.75 s	1.038 s
End to end delay App 1	116,045	0.73 s	1.019 s

Fig. 5.16 displays a graphical representation of E2E delay over time for both applications. Since there is no traffic shaper implemented in this experiment, data flows directly without prioritization, resulting in a single-queue setup. This allows us to observe natural variations in delay for each application without interference from shaping policies, providing insight into how delays fluctuate in an unmanaged, First-Come-First-Serve environment. Overall, the network exhibits moderate mean delays with substantial variability, which is typical of a setup with no traffic shaping.

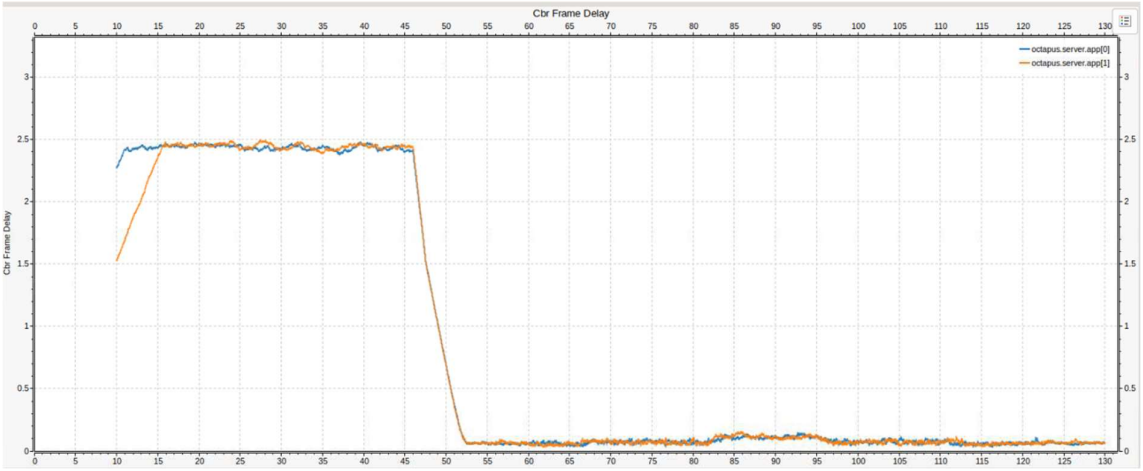


Fig. 5.16: End to end delay with no shaper.

5.2.4.2 Time Aware Shaper (TAS)

An example of the TAS with gate configurations for the first gate (G1), being mapped to Best effort traffic, and the second gate (G2) being mapped to Video traffic is presented in Fig. 5.17, where the notation O indicates an open gate (marked with green colour) and C indicates the gate is closed (marked with red colour). The first gate, G1, does not have any offset configured. In this case, both G1 and G2 start without any offset. G1 is configured with a [6,4] schedule, meaning that G1 remains open for 6 time slots and then closes for 4 slots. On the other hand, G2 is configured with a [4,6] schedule, where its

gate opens for 4 time slots and closes for 6 slots. During slots t1 to t4, both gates are open simultaneously, allowing traffic from both streams to pass through. However, starting from t5, the gate for G2 closes while G1 remains open for 2 more slots, until t6. After this, both gates close from t7 to t10, causing a pause in traffic. At t11, both gates reopen at the same time, allowing transmission for the remaining time slots. This case illustrates a synchronized start between the two traffic streams, with a clear overlap in the first 4 slots where both traffic types are transmitted.

G1 = Best effort, [6,4]
G2 = Video, [4,6] with offset 0

	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13	t14
G1	O	O	O	O	O	O	C	C	C	C	O	O	O	O
G2	O	O	O	O	C	C	C	C	C	C	O	O	O	O

Fig. 5.17: Example of TAS configuration without offset.

Fig. 5.18 presents the effect of a configuration offset. Specifically, G2 introduces an offset of 2, meaning that it starts operating two time slots later than in the first case. G1 maintains the same configuration as before, opening for the first six time slots (t1 to t6). However, due to the offset, G2 now only opens during the first two time slots (t1 to t2), overlapping with G1 during this brief period. After t2, G2 remains closed from t3 to t8, while G1 continues transmitting until it closes at t7. At t7, both gates are closed simultaneously, halting all traffic for the next two time slots (t7 to t8). The offset helps reduce direct contention between the two gates by staggering their operation in time more effectively, compared to the first case, where both gates had a longer overlap. This configuration creates a more efficient transmission schedule, although there are still brief periods where both gates are closed, leading to interruptions in both traffic streams. The offset thus provides better traffic separation, though some improvements can still be made to fully eliminate the overlapping closed periods.

G1 = Best effort, [6,4]
G2 = Video, [4,6] with offset 2

			t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	T12
G1			O	O	O	O	O	O	C	C	C	C	O	O
	Offset t1	Offset t2	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12
G2	O	O	O	O	C	C	C	C	C	C	O	O	O	O

Fig. 5.18: An example of TAS configuration with offset.

In our scenario, the simulation is set up with time slots, labelled t1 to t10, which control when the gates for each traffic type (i.e., Best Effort vs video) open or close. Best Effort traffic is assigned a gating cycle of [8, 2], meaning that its transmission gate remains open for the first eight time slots (t1 to t8) and closes during the last two (t9 and t10). Video traffic, on the other hand, is given a schedule of [2, 8], with an offset of eight time slots. This causes the video transmission gate to stay closed for the first eight slots and only open for the last two, during t9 and t10. Fig. 5.19 depicts the exact scheduling of both traffic types, showing the time slots where gates are open and closed. Best-effort traffic has priority early in the cycle, as its gate is open for most of the time, while video traffic is delayed until the end of the cycle.

Best effort G0 [8,2] 0 offset
Video G1 [2,8] 8 offset

									t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
G0									O	O	O	O	O	O	O	O	C	C
	Offset t1	Offset t2	Offset t3	Offset t4	Offset t5	Offset t6	Offset t7	Offset t8	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
G1	O	O	O	O	O	O	O	O	C	C	C	C	C	C	C	C	O	O

Fig. 5.19: TAS configuration for Best Effort and Video traffic types.

Table 5.4 presents an analysis of the incoming data rate values, also depicted in Fig. 5.20, for two cars, Car 0 and Car 1, when the TAS is applied as configured above. Both cars have similar data profiles,

with 1350 measurements for Car 0 and 1349 measurements for Car 1, which provides a reliable dataset for examining incoming data rates under TAS conditions. The mean incoming data rate for both cars is 7.8 Mbps, suggesting balanced and equal bandwidth allocation for each vehicle. This consistent average rate across both cars indicates that TAS is effectively distributing the available bandwidth equally, contributing to predictable network behaviour. The standard deviation of 2.5 Mbps for both Car 0 and Car 1 suggests moderate variability around the mean. This implies that, while the average data rate is stable, individual measurements may vary, potentially due to the inherent burstiness of network traffic or varying network conditions.

Table 5.4: Incoming data rate values with TAS.

Name	Number of packets	Mean	Standard Deviation
Incoming data rate Car 0	1350	7.8Mbps	2.5Mbps
Incoming data rate Car 1	1349	7.8Mbps	2.5Mbps

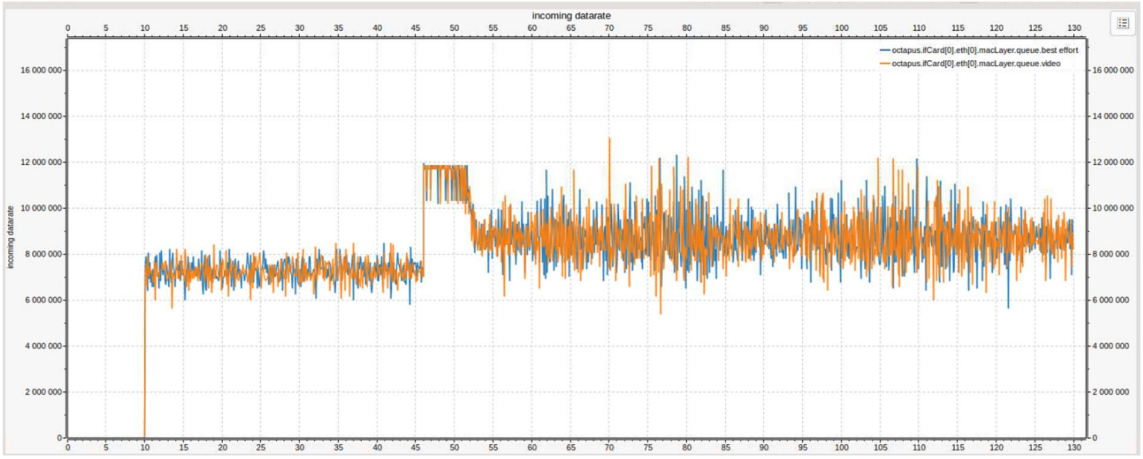


Fig. 5.20: Incoming data rate with TAS.

Table 5.5 in combination with Fig. 5.21 provide the analysis of the outgoing data rates for two different car configurations under the TAS. Car 0, with a mean outgoing data rate of 143.3 Mbps and a standard deviation of 338 Mbps, operates under a Best Effort queue configuration. The 8 seconds open and 2 seconds close schedule, without any initial offset, likely contributes to the variability seen in its outgoing data rate, as this setting does not prioritize data transmission, leading to sporadic bursts and potential delays. In contrast, Car 1 demonstrates a significantly higher mean outgoing data rate of 415.9 Mbps and a standard deviation of 484 Mbps, operating under a Video queue configuration. This queue is designed for prioritizing video traffic, characterized by a more aggressive gate management system of 2 seconds open and 8 seconds close, combined with an 8-second offset. This configuration facilitates a more stable and efficient flow of data, particularly suited for video applications that require consistent bandwidth. The larger standard deviation for Car 1, while indicative of some variability, suggests that the TAS effectively allocates resources to meet the higher demands of video streaming, resulting in overall improved data rates compared to Car 0.

Table 5.5: Outgoing data rate values with TAS.

Name	Number of packets	Mean	Standard Deviation
Outgoing data rate Car 0	1565	143.3 Mbps	338 Mbps
Outgoing data rate Car 1	2162	415.9 Mbps	484 Mbps

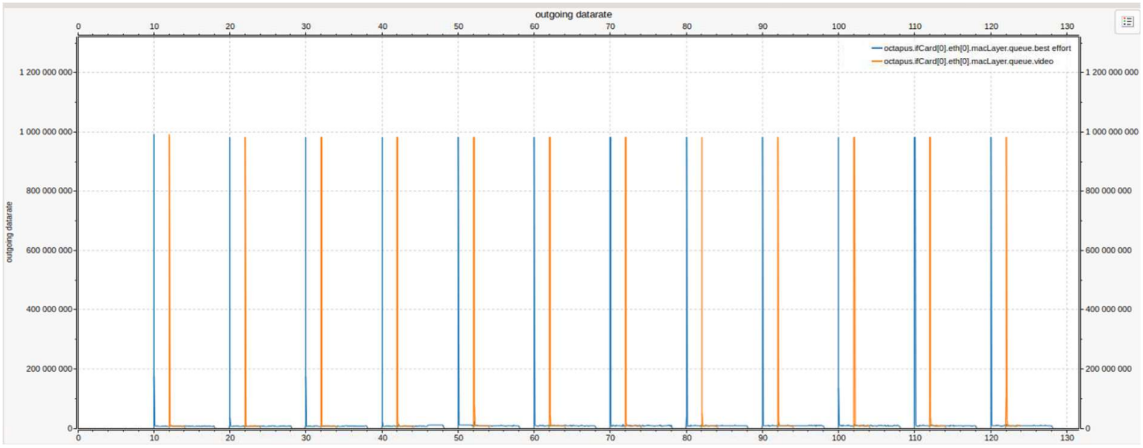


Fig. 5.21: Outgoing data rate with TAS.

Table 5.6 shows the E2E delay values for the two applications under TAS. Application 0 shows a mean E2E delay of 0.99 seconds with a standard deviation of 1.16 seconds. This low mean delay shows that TAS effectively prioritizes traffic for this application, allowing for efficient data transmission and minimizing latency. Application 1 experiences a much higher mean E2E delay of 4 seconds and a standard deviation of 2.8 seconds. These results demonstrate the effectiveness of the TAS in managing delays for lower-demand applications while indicating a need for improved resource management for more latency-sensitive applications. Tailoring the TAS settings or prioritization mechanisms may enhance the performance of Application 0, ensuring more consistent and acceptable end-to-end delay values across varying application types. This differentiation in performance emphasizes the importance of adapting the TAS to the specific requirements of each application to optimize overall network performance.

Table 5.6: End-to-end delay values with TAS.

Name	Number of packets	Mean	Standard Deviation
End to end delay App 0	115,868	0.99 s	1.16 s
End to end delay App 1	114,982	4 s	2.8s

The plotted dots shown in Fig. 5.22, representing the delays for both traffic types, exhibit a clear periodic pattern aligned with the gating schedule. Best Effort traffic (indicated by blue dots for Application 0 across all graphs) can be transmitted during the first eight slots and experiences low delays during this time, as reflected by the clustering of data points at lower delay values. In contrast, Video traffic (shown with orange dots for Application 1) faces an initial delay, as it only begins transmitting in the last two time slots. This results in slightly higher delay values, representing the waiting period for the video traffic's gate to open. Due to the TAS configuration, we observe that the maximum delay for Application 1 is 8 seconds, corresponding to the gating schedule for video traffic, while Application 0's maximum delay is 2 seconds. This reflects the prioritization of Best Effort traffic for the majority of the cycle's duration, ensuring low delays and efficient use of network resources for applications that can tolerate less stringent timing. In contrast, video traffic, while subject to some delay, transmits reliably at the end of each cycle, ensuring consistent delivery even for time-sensitive applications like video streaming. The results of the simulation demonstrate how TAS can effectively balance different types of network traffic by assigning specific time slots to each flow. This scheduling approach helps manage network resources efficiently, preventing congestion and allowing for predictable delays tailored to each traffic type, thus optimizing the network for both low-latency and time-sensitive applications.

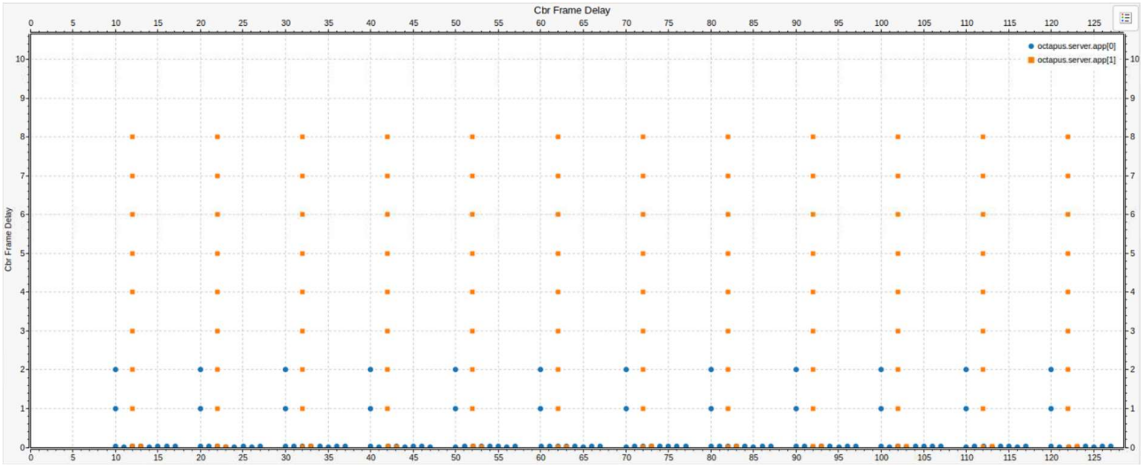


Fig. 5.22: End to end delay with TAS.

5.2.4.3 Credit Based Shaper (CBS)

The Credit-Based Shaper (CBS) operates by using a "credit" system to regulate the flow of time-sensitive traffic, ensuring predictable and bounded latency in mixed-traffic networks. Each high-priority traffic class accumulates or depletes credits over time, which determines when it can transmit data. For this configuration, the idle slope—which controls the rate at which credits are accumulated during idle periods—is set to 6 Mbps for Application 0 and 2 Mbps for Application 1. This configuration means that Application 0 can accumulate credits more quickly, allowing it to transmit at a higher rate compared to Application 1. When credits reach a determined threshold, the traffic is allowed to be transmitted, and as it does so, credits decrease. When transmission pauses, credits build up again based on the idle slope. This approach enables CBS to control bandwidth allocation, preventing network congestion and ensuring that time-sensitive data is delivered with minimal delay, even in environments with competing lower-priority traffic.

The outgoing data rates for two different car configurations under the CBS, as shown in Table 5.7 and visualized in Fig. 5.23, highlight distinct patterns influenced by the assigned idle slopes and traffic priorities. Car 0, configured with a higher idle slope of 6 Mbps, achieves a mean outgoing data rate of 5.4 Mbps with a standard deviation of 1.5 Mbps. This higher mean data rate aligns with the configuration, which allows Car 0 to accumulate credits more quickly, resulting in a steady and more substantial transmission rate compared to Car 1. The moderate standard deviation indicates relatively stable transmission with minor fluctuations, which is typical in CBS settings optimized for higher data flow. In contrast, Car 1, with an idle slope set to 2 Mbps, has a mean outgoing data rate of 1.8 Mbps and a standard deviation of 0.5 Mbps. This lower data rate reflects the slower accumulation of credits due to the lower idle slope, limiting the transmission rate for Car 1. The small standard deviation suggests consistent data flow with minimal variability, indicating that CBS effectively controls the traffic for lower-priority applications, ensuring bandwidth consistency.

Table 5.7: Outgoing data rate values with CBS.

Name	Number of packets	Mean	Standard Deviation
Outgoing data rate Car 0	1299	5.4 Mbps	1.5 Mbps
Outgoing data rate Car 1	1299	1.8 Mbps	0.5 Mbps

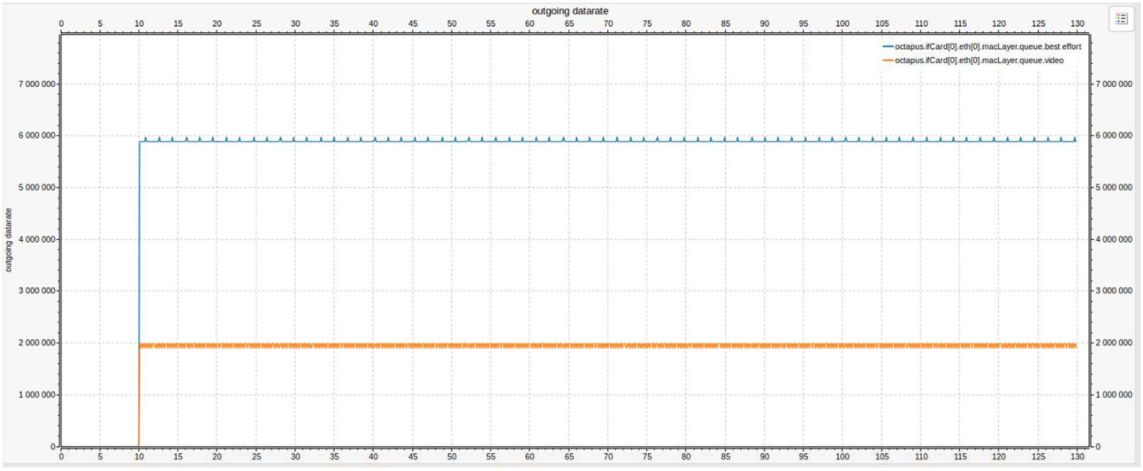


Fig. 5.23: Outgoing data rate with CBS.

The analysis of E2E delay values for the two applications, as presented in Table 5.8 and visualized in Fig. 5.24, highlights substantial differences in delay performance, reflecting varied application requirements and likely resource allocations. Application 0 shows a mean end-to-end delay of 17.9 seconds with a standard deviation of 9.4 seconds. This relatively lower mean delay and moderate variability suggest that Application 0 benefits from a more stable network path or prioritized handling, resulting in quicker and more predictable delivery times. However, the standard deviation indicates occasional spikes, which could slightly impact delay-sensitive scenarios. In contrast, Application 1 has a significantly higher mean end-to-end delay of 52.2 seconds with a standard deviation of 25.6 seconds. This higher delay suggests that Application 1 experiences more congestion or lower priority within the network, possibly due to less frequent access to transmission resources or more stringent data requirements. The high standard deviation reveals considerable variability, indicating inconsistent delivery times that could lead to performance issues for real-time or delay-sensitive tasks.

Table 5.8: End to end delay values with CBS.

Name	Number of packets	Mean	Standard Deviation
End to end delay App 0	81670	17.9 s	9.4 s
End to end delay App 1	27223	52.2 s	25.6 s

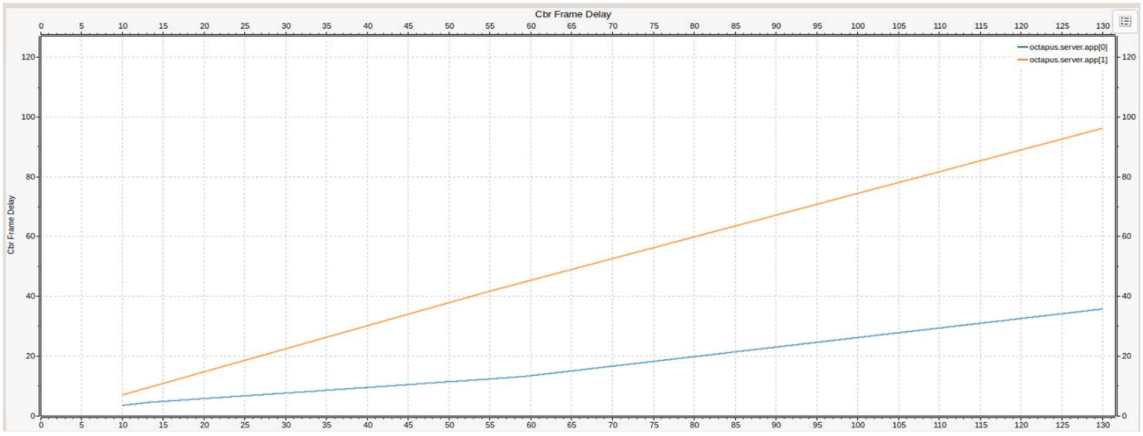


Fig. 5.24: End to end delay with CBS.

5.3 Optimization-driven resource allocation in OCTAPUS

In this section, we expand upon the resource allocation algorithms being executed in the OCTAPUS NGCO Control plane, as described in Deliverable D2.2 [2], related to the creation (by the CNC controller) of TSN-compliant transmission schedules and the selection of routed paths among the multiple connections between the Uplink switches and IF cards, under a diverse set of service-driven QoS requirements in terms of throughput and E2E latency. Deliverable D2.2 provided Mixed Integer Linear Problem (MILP)-based formulations for the following two types of problems:

- The scheduling only problem, where the routed path for each TSN flow has been selected *a priori* (potentially, by an external entity) and is provided as an input to the algorithms.
- The joint routing + scheduling problem, where the algorithm also determines the optimal routed path for each TSN flow.

In both cases, a multi-objective optimization formulation was employed, aiming to maximize the number of admitted TSN flows while also minimizing the *flowSpan*, i.e., the time of latest arrival of a TSN flow to its destination. An extensive simulation-based study was performed providing important insights, albeit also indicating the high computational complexity of the involved algorithms. The latter posed a severe challenge in terms of the algorithms' scalability, especially considering the dynamic nature of TSN flow requests arriving at the NGCO, which necessitates the frequent execution of the above algorithms.

In the current Deliverable, we pick up from the last point above and investigate ways to reduce the computational complexity of the joint routing + scheduling problem by decomposing it into two sub-problems to be solved in sequence: i) a routing only sub-problem, and ii) a scheduling only sub-problem, using the routed paths determined from the solution of the first sub-problem. Two variations of this decomposition are proposed, and their performance is compared to the optimal (but computationally expensive) joint routing + scheduling algorithm via an extensive simulation study, which results in the development of a heuristic approach providing an excellent performance/complexity tradeoff. We also present a multi-objective variant of the joint routing + scheduling algorithm, which will only be employed in specific circumstances to improve the robustness of the algorithms.

The formulated MILP problems are still solved using IBM's CPLEX optimization suite [8],which offers many attractive features such as a simulation time limit (upon which CPLEX returns the best solution, if any, it has found so far) and an accompanying upper/lower bound as an indicator of the solution's "quality" if the simulation time limit is reached before the optimal solution is found.

We next describe the model formulation, which has been slightly modified from Deliverable D2.2 to include additional periodicity constraints, and present the associated lower-complexity algorithms for the individual sub-problems.

5.3.1 Joint scheduling and routing of time-sensitive flows: a complexity study

5.3.1.1 Theoretical model

The model presented below follows the work presented in [9] but introduces some extensions in subsequent Sections. We consider a switched Ethernet network containing Ethernet switches and endpoint Ethernet NICs (referred to as "hosts"). Although we assume full-duplex Ethernet links, we treat the network topology as a **directed** graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and assume that it operates in cycles of duration T_{cycle}, with the transmission schedule being periodically repeated with period T_{cycle}. Each directed link $(u,v) \in \mathcal{E}$ introduces a propagation delay $d_{u,v}^{prop}$, as well as a transmission delay $d_{u,v}^{trans}$, the latter depending on the link line-rate and the size of the transmitted Ethernet frame. Also, each node $u \in \mathcal{V}$ has an associated processing delay d_v^{proc} incurred by processing the received Ethernet frames and deciding where to forward them.

Since the routing decision is now performed by the CNC controller, we introduce the notion of a unidirectional TSN stream request r, characterized by the tuple $r = \langle id_r, src_r, dst_r, T_r, F_r, sz_r, D_r^{E2E} \rangle$, where:

- id_r is a globally unique request identifier.
- $src_r \in \mathcal{V}$ is the stream's source node.

- $dst_r \in \mathcal{V}$ is the stream's destination node.
- T_r is the period of request r, in the sense that the frame arrival process of the corresponding stream is periodic (in time) with this period. Without loss of generality, and assuming a sufficiently fine temporal granularity (say, at the ns scale), we assume that T_r is integer.
- F_r is the number of Ethernet frames generated by the stream at the beginning of each period. **All** of these frames must be transmitted from src_r within the period where they were generated. Hence, stream r effectively generates F_r frames in every interval of duration T_r.
- sz_r is the size (in bytes) of each frame generated by stream
- D_r is the maximum allowed E2E delay for all frames, i.e., the difference between the time of arrival at the stream destination dst_r from the time of transmission at the stream source src_r

We next use the terms "stream" and "request", for index r, interchangeably. We denote with $\mathcal{R}$ the set of all TSN stream requests. Since each stream is associated with a periodic frame arrival process, we can select an extended interval duration, such that all streams are periodically repeated within this interval. This is easily achieved by selecting $T_{cycle} = LCM(T_r: r \in \mathcal{R})$, where LCM is the Least Common Multiple of the integer periods of the above streams. Under this selection, it follows immediately that each stream is periodically repeated T_{cycle}/T_r times within the cycle duration T_{cycle}, and the cycle is then periodically repeated ad infinitum. Similarly, since stream r generates F_r frames in a interval of duration T_r, it follows that the stream will also generate a number $F_r^{cycle} = F_r \frac{T_{cycle}}{T_r}$ of frames within the cycle duration T_{cycle}. For each stream r, we denote with $\mathcal{F}_r = \{f: 0 \leq f \leq F_r - 1\}$ the set of frames generated within each period of duration T_r and with $F_r^{cycle} = \{f: 0 \leq f \leq F_r^{cycle} - 1\}$ the set of frames generated within the entire cycle of duration T_{cycle}

We assume that, for each stream request r, the Ethernet frames can only be forwarded along the paths specified in the set $\mathcal{P}_r$, which effectively captures constraints imposed by external entities (e.g., the OCTAPUS SDN Coordinator) due to administrative, security or network intelligence-based policies such as these mentioned in Section 4.5.1. Each path $p \in \mathcal{P}_r$ starting at src_r and terminating at dst_r is essentially an ***ordered*** sequence O_p of nodes in $\mathcal{V}$, i.e., $O_p = \langle v_1, \cdots, v_{N_p} \rangle$ such that $v_1 = src_r$ and $v_{N_p} = dst_r$, where N_p is the number of nodes in path p. We denote with $pred_p(v)$, $succ_p(v)$ the predecessor and successor, respectively, of node v in path p. To provide an example of this notation, for a path $O_p = \langle 2,3,8,5 \rangle$, it holds $pred_p(5) = 8$, $pred_p(8) = 3$, $pred_p(3) = 2$ and similarly $succ_p(2) = 3$, $succ_p(3) = 8$, $succ_p(8) = 5$. By convention, the first node in a path has no predecessor and the last node in the path has no successor, i.e., $pred_p(2) = \emptyset$ and $succ_p(5) = \emptyset$

We require all packets of a given stream r to travel along the same path $p \in \mathcal{P}_r$, so that all packets within the stream experience the same E2E delay and the jitter among them is eliminated. For each path $p \in \mathcal{P}_r$ and node v contained in this path (i.e., $v \in O_p$), we denote with $D_{p,u}$ the ***cumulative*** delay along path p, starting from the path's beginning up to node u, including the processing delay at u, **as well as** the transmission delay on the link $(u, succ_p(u))$. Under this notation, the E2E delay along path $p \in \mathcal{P}_r$ is equal to $D_{p,v_{N_p}} = D_{p,dst_r}$. For convenience, we also use the notational shortcut $D_p = D_{p,v_{N_p}}$.

We introduce the following decision variables:

- Boolean decision variables $\gamma_{r,p}$ indicate whether the packets of request r are actually forwarded along path $p \in \mathcal{P}_r$.
- Floating point decision variables $t_{r,f}^{in}$ denote the actual time when frame f of request r (i.e., $f \in \mathcal{F}_r^{cycle}$) is injected into the network (equivalently, its transmission beginning at node src_r).
- Floating point decision variable $flowSpan$ denotes the highest interval duration until an injected frame reaches its destination, with the reference time being the beginning of the period where the frame is generated. Hence, it holds $flowSpan = \max_{r \in \mathcal{R}, f \in \mathcal{F}_r^{cycle}} \left(t_{r,f}^{in} - \lfloor f/F_r \rfloor T_r\right)$. We note that this definition is slightly different than the one used in Deliverable D2.2.
- Boolean decision variables $\rho_{r_1,f_1,p_1,r_2,f_2,p_2}^{u,v}$ indicate whether frame f_1 of stream request r_1 (i.e., $f_1 \in \mathcal{F}_{r_1}^{cycle}$) is scheduled before frame f_2 of r_2 (i.e., $f_2 \in \mathcal{F}_{r_2}^{cycle}$) on the common link (u, v) of paths $p_1 \in \mathcal{P}_{r_1}$ and $p_2 \in \mathcal{P}_{r_2}$.

To better illustrate the semantics of $\rho^{u,v}_{r_1,f_1,p_1,r_2,f_2,p_2}$, Fig. 5.25 provides a graphical example showing that, for any links that belong to allowable paths of different streams, the frames of these streams cannot have their transmissions overlapping, in time, along any common links, as this would cause a conflict.

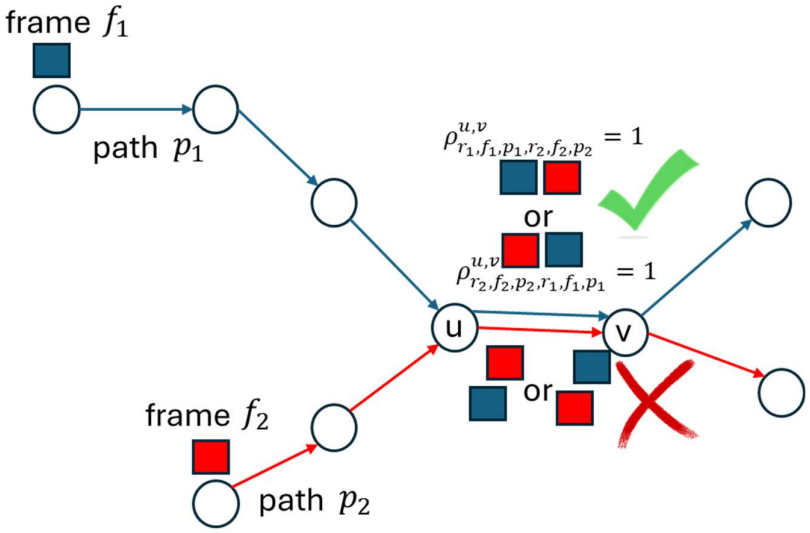


Fig. 5.25: Two paths (shown in blue and red) of different streams sharing a common link cannot have temporally overlapping frame transmissions on this link.

Based on the above notation, the following constraints arise immediately. First, as mentioned above, all frames of a given stream must be forwarded along the same single path belonging to the set of allowable paths for this stream, so that it must hold

$$\sum\nolimits_{p\in\mathcal{P}_r} \gamma_{r,p} = 1, \quad \forall r \in \mathcal{R}. \tag{1}$$

Additionally, all frames of a stream generated at the beginning of its period must also be scheduled for transmission within the same period. The following constraint guarantees that this condition is met:

$$\lfloor f/F_r \rfloor T_r \le t^{in}_{r,f} \le \left(\left\lfloor \frac{f}{F_r} \right\rfloor + 1\right) T_r - \sum\nolimits_{p\in\mathcal{P}_r} \max_{(u,v)\in O_p} \left(\gamma_{r,p} d^{trans}_{u,v}\right), \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{2}$$

Once generated, all frames must reach their destination within an interval of duration less than the maximum allowed E2E latency, which can be written as

$$t^{in}_{r,f} - \left\lfloor \frac{f}{F_r} \right\rfloor T_r + \sum\nolimits_{p\in\mathcal{P}_r} \gamma_{r,p} D_p \le D^{E2E}_r, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{3}$$

We further require that all frames within the same stream are scheduled for transmission in the order in which they are generated, i.e.,

$$t^{in}_{r,f} + \sum\nolimits_{p\in\mathcal{P}_r} \max_{(u,v)\in O_p} \left(\gamma_{r,p} d^{trans}_{u,v}\right) \le t^{in}_{r,f+1}, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{4}$$

Finally, by definition of the flowSpan, it must hold

$$t^{in}_{r,f} - \left\lfloor \frac{f}{F_r} \right\rfloor T_r \le flowSpan, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{5}$$

As indicated by Fig. 5.25, there must be a temporal precedence on any common links between two paths, i.e., if frame f_1 of request r_1 and frame f_2 of request r_2 need to traverse link (u, v) that is common in both paths p_1, p_2, then either the transmission of f_1 on link (u, v) must complete before transmission of f_2 begins, or the reverse condition should apply. Under the introduced notation, the former case corresponds to the requirement

$$t^{in}_{r_2,f_2} + D_{p_2,pred_{p_2}(u)} + d^{proc}_u \ge t^{in}_{r_1,f_1} + D_{p_1,u}, \tag{6}$$

where $t^{in}_{r_1,f_1} + D_{p_1,u}$ is the time when frame f_1 completes its transmission on link (u, v) (recall that $D_{p_1,u}$ **includes** the transmission delay on the subsequent link of path p_1, i.e., the link (u, v) itself) and $t^{in}_{r_2,f_2} +$

$D_{p_2,pred_{p_2}(u)} + d_u^{proc}$ is the time when frame f_2 arrives at node u. The reverse scenario yields the following requirement

$$t_{r_1,f_1}^{in} + D_{p_1,pred_{p_1}(u)} + d_u^{proc} \geq t_{r_2,f_2}^{in} + D_{p_2,u}. \quad (7)$$

Some thought reveals that both above conditions can be captured in the following two constraints:

$$t_{r_1,f_1}^{in} - t_{r_2,f_2}^{in} + M\rho_{r_1,f_1,p_1,r_2,f_2,p_2}^{u,v} + M\left(2 - \gamma_{r_1,p_1} - \gamma_{r_2,p_2}\right) \geq D_{p_2,u} - D_{p_1,pred_{p_1}(u)} - d_u^{proc}, \quad (8)$$

$$\forall r_1, r_2 \in \mathcal{R}\ s.t. r_1 \neq r_2, \forall f_1 \in \mathcal{F}_{r_1}^{cycle}, \forall f_2 \in \mathcal{F}_{r_2}^{cycle}, \forall p_1 \in \mathcal{P}_{r_1}, \forall p_2 \in \mathcal{P}_{r_2}\ \forall (u,v) \text{ common in } p_1, p_2,$$

$$t_{r_2,f_2}^{in} - t_{r_1,f_1}^{in} + M\left(1 - \rho_{r_1,f_1,p_1,r_2,f_2,p_2}^{u,v}\right) + M\left(2 - \gamma_{r_1,p_1} - \gamma_{r_2,p_2}\right) \geq D_{p_1,u} - D_{p_2,pred_{p_2}(u)} - d_u^{proc}, \quad (9)$$

$$\forall r_1, r_2 \in \mathcal{R}\ s.t. r_1 \neq r_2, \forall f_1 \in \mathcal{F}_{r_1}^{cycle}, \forall f_2 \in \mathcal{F}_{r_2}^{cycle}, \forall p_1 \in \mathcal{P}_{r_1}, \forall p_2 \in \mathcal{P}_{r_2}\ \forall (u,v) \text{ common in } p_1, p_2,$$

where M is a sufficiently large constant. Essentially, by setting $\gamma_{r_1,p_1} = \gamma_{r_2,p_2} = 1$, (8) reduces to (7) when $\rho_{r_1,f_1,p_1,r_2,f_2,p_2}^{u,v} = 0$ (in which case, (9) holds trivially for a sufficiently large M), whereas (9) reduces to (6) when $\rho_{r_1,f_1,p_1,r_2,f_2,p_2}^{u,v} = 1$ (in which case, (8) again holds trivially). Equations (8), (9) essentially capture the noWait constraint described in D2.2.

To provide the final form of the MILP formulation, we must handle the expression $\max_{(u,v)\in O_p}\left(\gamma_{r,p} d_{u,v}^{trans}\right)$ appearing in (2), (4) to convert it into a linear constraint. This is achieved by introducing the auxiliary variable $\xi_{r,p} = \max_{(u,v)\in O_p}\left(\gamma_{r,p} d_{u,v}^{trans}\right)$ such that (2), and (4) are converted, respectively, to the following linear constraints:

$$\lfloor f/F_r \rfloor T_r \leq t_{r,f}^{in} \leq \left(\left\lfloor \frac{f}{F_r} \right\rfloor + 1\right) T_r - \sum_{p\in\mathcal{P}_r} \xi_{r,p} \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}_r^{cycle}, \quad (10)$$

$$t_{r,f}^{in} + \sum_{p\in\mathcal{P}_r} \xi_{r,p} \leq t_{r,f+1}^{in}, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}_r^{cycle}. \quad (11)$$

By definition of $\xi_{r,p}$, we need to impose the additional constraint

$$\xi_{r,p} \geq \gamma_{r,p} d_{u,v}^{trans}, \forall r \in \mathcal{R}, \forall p \in \mathcal{P}_r, \forall (u,v) \in p. \quad (12)$$

Hence, the final MILP formulation becomes

$$\begin{aligned} &minimize \quad flowSpan, \\ &s.t.\ (1), (3), (5), (8) - (12). \end{aligned} \quad (13)$$

Since the problem in (13) is NP-hard, an optimal solution to (13) obtained by CPLEX does not scale well in terms of network size. In the following Section, we examine ways to reduce the computational time of solving (13) by decomposing it into smaller complexity subproblems. We also note that, depending on the provided parameters of the requested streams, the problem in (13) may be infeasible. In this case, an attractive feature of any algorithm is that it quickly detects this infeasibility. We take this ability into consideration as we next present the proposed algorithms.

5.3.1.2 Proposed algorithms

The first algorithm to be evaluated is named *TrueJoint* and entails the solution (via IBM CPLEX) of the joint routing and scheduling problem in (13), to be used as a benchmark tool for the subsequent algorithms. To reduce the computational complexity of this problem, we also apply the heuristic algorithm proposed in [9], which we refer to as *Stage2S*, by breaking the joint routing + scheduling problem into two subproblems, namely a pure routing problem and a pure scheduling problem, as follows:

1. We first determine the routed paths for each request by solving the following, reduced complexity, MILP:

$$\begin{aligned} &minimize \quad FS, \\ &s.t. \sum_{p\in\mathcal{P}_r} \gamma_{r,p} = 1, \quad \forall r \in \mathcal{R}, \\ &s.t. \quad \sum_{p\in\mathcal{P}_r} \max_{(u,v)\in O_p}\left(\gamma_{r,p} d_{u,v}^{trans}\right) \leq FS, \ \forall r \in \mathcal{R}, \forall f \in \mathcal{F}_r^{cycle}, \end{aligned} \quad (14)$$

where only the decision variables $\gamma_{r,p}, FS$ are considered. The problem in (14) follows a shortest weighted path-like approach, where the routed paths are selected such that their total delay (technically, the delay terms captured in the left hand side of the second constraint in (14)) is less than an upper bound FS (to be minimized) and, therefore, as small as possible. However, no scheduling constraints are imposed per se on the actual frames of each requested stream in (14). In fact, it is quite possible that the variables $\gamma^*_{r,p}$ arising as the solution to (14) will contain paths with common edges, so that appropriate scheduling ***must*** be performed at a later stage.

2. We now use the solution $\gamma^*_{r,p}$ from the previous step (i.e., the selected routed paths) to perform the actual scheduling according to the constraints of Section 5.3.1.1. Hence, we solve the following MILP problem, again via CPLEX:

$$\begin{aligned} &minimize \quad flowSpan, \\ &s.t.\ \gamma_{r,p} = \gamma^*_{r,p}\ , \quad \forall r \in \mathcal{R}, \forall p \in \mathcal{P}_r, \\ &s.t.\ (3), (5), (8) - (12), \end{aligned} \tag{15}$$

where only the decision variables $t^{in}_{r,f}, flowSpan, \rho^{u,v}_{r_1,f_1,p_1,r_2,f_2,p_2}$ are considered. Clearly, with the routed paths being selected via the solution of (14), solving (15) according to the same scheduling constraints as in (13) addresses the scheduling subproblem.

The two stages described above justify the name *Stage2S* and warrant the following remarks:

- The routing problem of (14) always admits a solution. Hence, $\gamma^*_{r,p}$ are always available and the problem in (15) is well-posed; however, it may also be infeasible as we discuss next and is later confirmed by the simulation results.
- Removing the need to solve for $\gamma_{r,p}$ in (15) significantly reduces the computational time of solving (15). Similarly, the solution to (14) can be computed very quickly since the search space is much smaller compared to (13). However, the possibility arises that, due to the treatment of routing and scheduling at separate stages, the selected paths determined by the solution to (14) make the problem in (15) infeasible.

The last statement above indicates a potential weakness of *Stage2S*, in the sense that it may "declare" the combination of problems (14), (15) to be infeasible whereas the original joint problem in (13) actually has a solution. This weakness, which is the "price" to pay for the significant reduction in computational complexity, is addressed within our work in OCTAPUS in the following two ways:

- If the combination of (14), (15) leads to an infeasible problem, we can enumerate the different routed path selections and, for each of them, solve (15) until a solution is found or the problem is determined to be infeasible for all possible routed path selections. In the latter case, it immediately follows that the problem in (13) is also infeasible. This modified algorithm is referred to as *Stage2M* and will be described in pseudocode below.
- Additionally, a slightly modified formulation of the problem in (13) can be provided where we don't require all requested streams to be admitted into the network but only wish to maximize the number of the admitted streams while also minimizing their flowSpan. This formulation allows us to always provide a solution to this new multi-objective problem (i.e., the modified version of (13) is never infeasible) and will be described in Section 5.3.2.

Coming back to the proposed algorithm *Stage2M,* its logic is quite simple: first, run *Stage2S*. If a solution is found, there is no need to do anything else. Otherwise, (i.e., if (15) is infeasible), we solve (15) for each different path selection (i.e., we skip solving (14) and, instead, apply the arbitrary path selection to determine $\gamma^*_{r,p}$ in (15)) until either a solution is found or the problem in (15) is infeasible for all possible path selections. The pseudocode description of *Stage2M* is provided in Fig. 5.26.

Input: network topology, request set $\mathcal{R}$ and associated QoS requirements, set of allowable paths $\mathcal{P}_r$
Output: decision variables $\gamma_{r,p}$, $t^{in}_{r,f}$ $flowSpan$, $\rho^{u,v}_{r_1,f_1,p_1,r_2,f_2,p_2}$ or detecting infeasible problem

1. run Stage2S. If solution found, **return**
2. Otherwise, enumerate all possible path selections $(p_1, p_2, \cdots, p_R) \in \bigotimes_{r\in\mathcal{R}} \mathcal{P}_r$
3. **for each** $(p_1, p_2, \cdots, p_R) \in \bigotimes_{r\in\mathcal{R}} \mathcal{P}_r$
4. solve (15) using $(p_1, p_2, \cdots, p_R)$ as the routed paths
5. if solution found, **return**
6. **end for**
7. *// problem is infeasible if code reaches this line*

Fig. 5.26: Pseudocode of *Stage2M*.

5.3.1.3 Performance evaluation

We evaluate the TSN resource allocation algorithms proposed in Section 5.3.1.2 when applied to the network topology shown in Fig. 5.27, which has been purposely selected to resemble the OCTAPUS topology, with nodes 1-10 and 28-38 being the TSN traffic endpoints, nodes 11-14 and 23-27 acting as traffic aggregators in the metro and access segments, respectively, and nodes 15-18 and 19-22 representing the OCTAPUS Uplink switches and IF cards, respectively. Of course, all algorithms in Section 5.3.1.2 can be applied in arbitrary topologies. All links in Fig. 5.27 are bidirectional; however, since nodes 1-10 and 28-38 are the potential sources and destinations, respectively, of the TSN streams, we have only indicated the direction of the links towards the destinations.

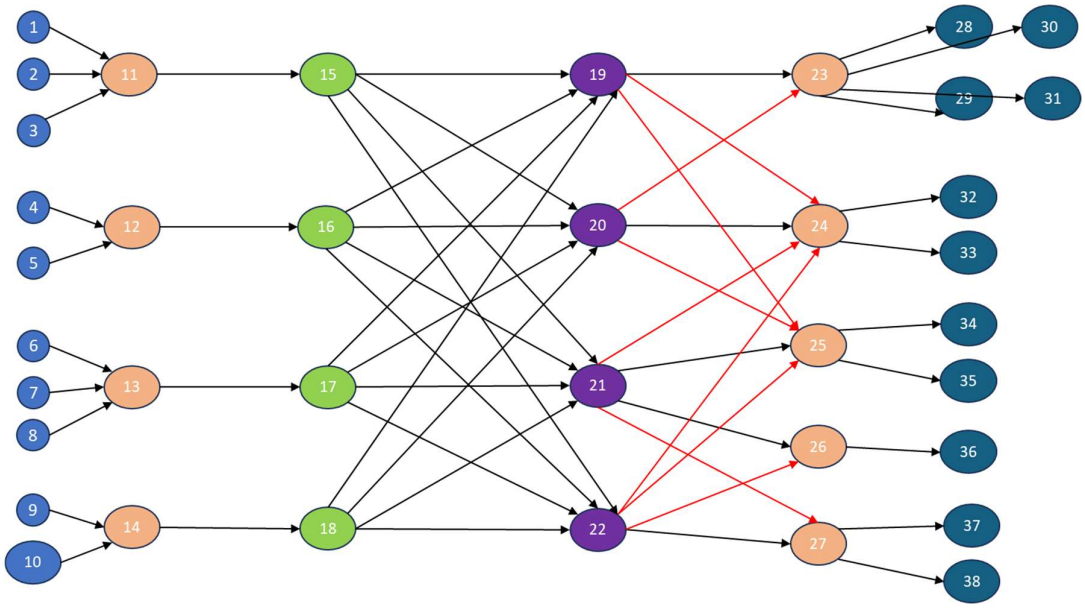


Fig. 5.27: Illustrative network topology for comparison of TSN algorithms.

The various simulation parameters for the execution of algorithms *TrueJoint*, *Stage2S*, *Stage2M* have been collected in Table 5.9, which shows that many parameters have been randomly selected to explore the algorithms' ability to handle TSN stream requests with non-uniform QoS requirements. Although the number of frames per period may initially seem too small, the duration of the period has also been selected low enough to yield an average throughput value well in excess of current TSN applications. For example, a TSN stream sending 2 Ethernet frames of size 1000 bytes every 30 μs results in an average throughput of 2*1000*8/30μs=533.33 Mbps. The specific ranges of the values of the parameters in Table 5.9 have been selected for illustration purposes only and can be arbitrarily modified. For the parameter values which are randomly selected in Table 5.9, we refer to the tuple of actually selected values as "a problem instance" and, for each pair of values $(|\mathcal{R}|, T_r)$ in Table 5.9, we select 100 random instances to allow for a statistical description of the results.

Table 5.9: Summary of employed simulation parameters for TSN resource allocation.

Simulation parameter	Value
Transmission rates	1, 10 Gbps
Request period T_r	30, 60, 120 μs
Number of requests $\|\mathcal{R}\|$	5, 10 (for $T_r = 120$ only)
Number of frames per period	2, 3, 4 (if $T_r = 30, 60, 120$, respectively)
Ethernet size	Randomly selected in [200, 300, 400, 600, 1000] bytes
TSN stream source node	Randomly selected in nodes 1-10
TSN stream destination node	Randomly selected in nodes 28-38
Max E2E latency	Randomly selected in [100, 150, 200] μs
Fiber length	Varying per link in the range 10 m (intra-NGCO links) to 10 km (access/metro connections)

We begin the discussion by quantitatively examining *Stage2S*'s weakness of "artificially" creating an infeasible problem in (15) due to the decoupling of the routing and scheduling problems. As shown in Fig. 5.28, for the selected range of parameters in Table 5.9, there always exist some "truly infeasible" problems, since even *TrueJoint* cannot solve all 100 instances for a given combination of number of requests and period. The fact that the bars in Fig. 5.28 always have the same height for *TrueJoint* and *Stage2M* proves that *Stage2M* "works hard" to determine if a problem is infeasible and never falsely identifies a solvable problem as infeasible (since in the worst case scenario, *Stage2M* will search ***all*** possible path combinations to find a solution, if one exists). The same is not true for *Stage2S*, which wrongly identifies many problem instances as infeasible, whereas *TrueJoint* and *Stage2M* provide solutions for them. As intuition suggests and Fig. 5.28 confirms, this undesirable effect is exacerbated for lower period values under a given number of requests, or for an increasing number of requests under a given stream period value. This follows from the fact that lower period values essentially force (for a fixed number of requests) the algorithms to "fit" the scheduled Ethernet frames into a smaller duration interval, which makes the problem more likely to be infeasible. A similar explanation is valid for the case of increasing number of requests with a fixed period.

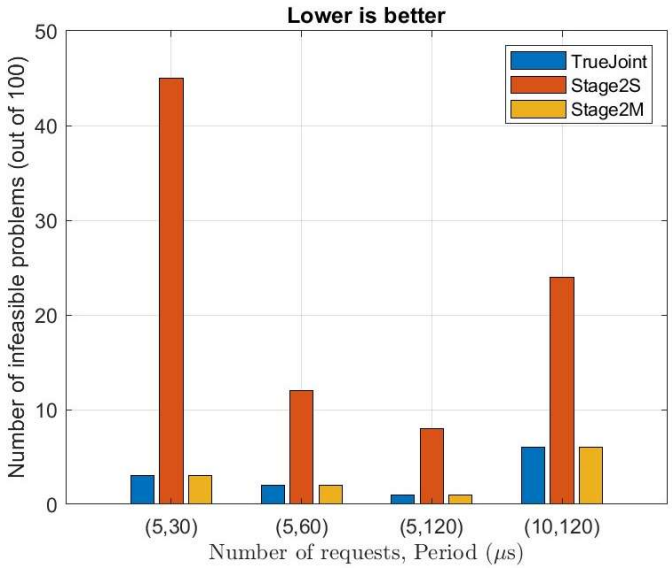


Fig. 5.28: Number of infeasible problems (out of 100 instances) identified by *TrueJoint*, *Stage2M*, *Stage2S* for different TSN stream configurations.

We also investigate the computational savings gained by *Stage2S*, *Stage2M* when splitting the original joint problem into the two subproblems. Fig. 5.29 provides a statistical summary of the algorithms' execution times in the form of a boxplot. For the reader's convenience, we next provide a brief outline of the boxplot's semantics: each configuration of $(|\mathcal{R}|, T_r)$ is mapped to one column in Fig. 5.29, along with the corresponding box. Each box is described by a closed rectangle, whose horizontal lines denote (from bottom to top) the 25%, 50% (i.e., median) and 75% quantiles of the underlying distribution of values. There are also narrower horizontal lines (referred to as "whiskers" and connected to the box via dashed vertical lines) which identify the tails of the distribution. Extreme outlier values are denoted with the + sign.

Fig. 5.29 presents the statistics of the algorithms' execution time (denoted in the vertical axis in a logarithmic scale of seconds) where, for visual convenience, the boxplots of *TrueJoint*, *Stage2S*, *Stage2M* are colored with blue, magenta and black, respectively. It is immediately apparent that the execution time of all algorithms increases as the number of requests increases but *Stage2S*, *Stage2M* scale much better (i.e., computation time increases slower with number of requests) than *TrueJoint*. Considering the median boxplot values in Fig. 5.29, we conclude that *Stage2S*, *Stage2M* offer higher computational savings as the number of requests increases (which is the regime where *TrueJoint* scales poorly), reaching up to 30X less computation time than *TrueJoint*. We note that Fig. 5.29 only considers the instances leading to a feasible problem; however, the overall statistical distribution for *TrueJoint* and *Stage2M* does not change significantly when the infeasible problem instances are included in the boxplot. This lack of variation is due to the fact that, according to Fig. 5.28, there are only very few truly infeasible problems (i.e., at most 6 out of a 100 instances for 10 requests of period 120 μs), which implies that even if it takes a lot of time to identify these problems as infeasible by *Stage2S* (and *Stage2M*), these eventualities are too rate to have a significant impact.

We also investigate the "quality" of the solutions produced by *Stage2S*, *Stage2M* using *TrueJoint* as a benchmark tool. To this end, and considering only the feasible problems, we normalize the flowSpan values of the solutions returned by *Stage2S* and *Stage2M* with respect to the corresponding solution returned by *TrueJoint* as (we only show the *Stage2S* normalization but *Stage2M* is handled similarly)

$$Normalized\ flowSpan_{Stage2S} = \frac{flowSpan_{Stage2S}}{flowSpan_{TrueJoint}}. \tag{16}$$

By definition, the normalized flowSpan values for *Stage2S*, *Stage2M* are always greater or equal to 1, where the optimal performance is achieved for a normalized value of 1 and lower normalized values indicate superior performance.

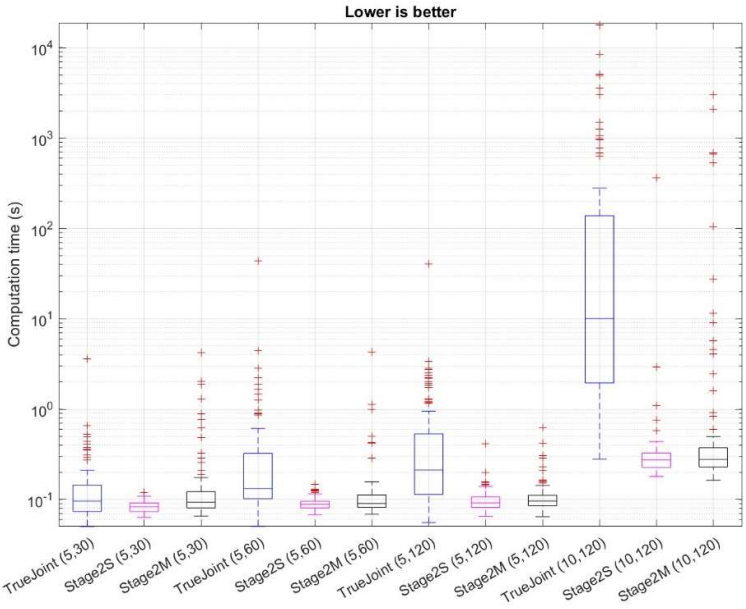


Fig. 5.29: Computation time of *TrueJoint*, *Stage2S*, *Stage2M*.

The normalized flowSpan values for *Stage2S* and *Stage2M* are shown in Fig. 5.30, again in boxplot format and using the same group coloring scheme as in Fig. 5.29. Clearly, Fig. 5.30 shows outlier

problem instances where *Stage2S*, *Stage2M* return a solution that is ~4 times higher (i.e., worse) than the optimal *TrueJoint* flowSpan. However, considering again the median values in Fig. 5.30, we conclude that, for all configurations, the median normalized values of *Stage2S* and *Stage2M* are close to 1.3; this indicates a 30% performance drop compared to *TrueJoint*, albeit with the significant computational savings reported in Fig. 5.29.

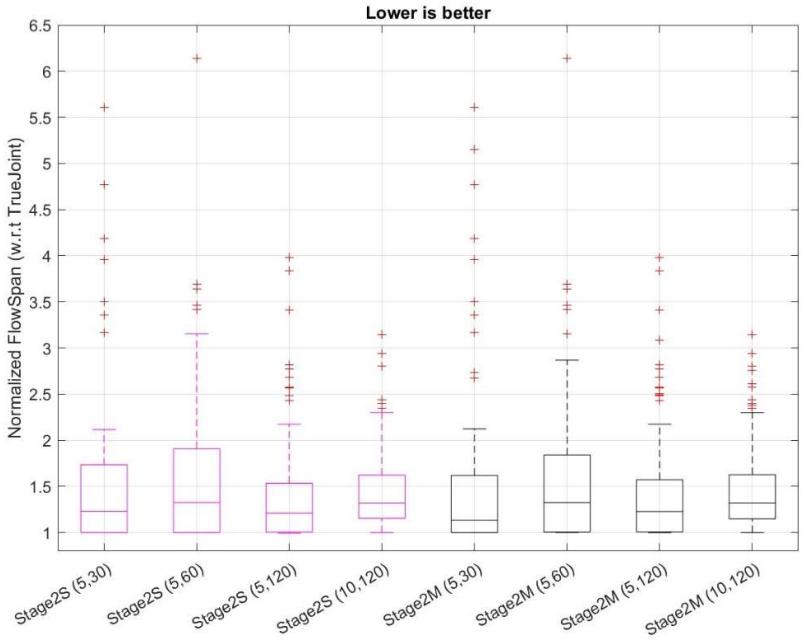


Fig. 5.30: Normalized flowSpan of *Stage2S*, *Stage2M* compared to *TrueJoint*.

It should also be stated that, for static environments where the number of TSN stream requests does not change over time (i.e., there are no new requests arriving into the network), any valid solution returned by *Stage2S*, *Stage2M* is "good enough", since it allocates the necessary resources for the Ethernet frames of the streams. It is mostly in dynamic environments where a low flowSpan value is advantageous, since such a solution leaves "more time" available for future TSN requests. The last statement also indicates the benefit of proactively allocating resources for TSN streams of known QoS requirements, even if these streams are not active yet (i.e., essentially allocating resources for a "worst-case" scenario), as the pooling of TSN requests allows for a potentially lower flowSpan value as opposed to sequentially performing the resource allocations for the streams.

So far, the above results indicate that *Stage2M* offers a viable, performance-wise, alternative to the computationally intensive *TrueJoint* algorithm, with huge computational savings and without *Stage2S*'s disadvantage of incorrectly labeling a solvable problem as infeasible. However, the issue remains that *Stage2M* must enumerate all possible path selections until it determines a problem to be infeasible, which raises some concerns on whether the event of encountering an infeasible problem, rare though it may be (as shown in Fig. 5.28), will lead to unacceptably high computation times. To this end, Fig. 5.31 plots the amount of elapsed time until *TrueJoint*, *Stage2S*, *Stage2M* determine a problem to be infeasible. It is clear from Fig. 5.31 that *TrueJoint* can identify an infeasible problem within a few seconds (typically, below 1 sec and at maximum 10 sec), whereas *Stage2M*, as expected, may need up to 10000 sec until it tries all possible path selections. *Stage2S* can detect an infeasible problem even faster than *TrueJoint*, although it may mis-identify an infeasible problem, as shown in Fig. 5.28.

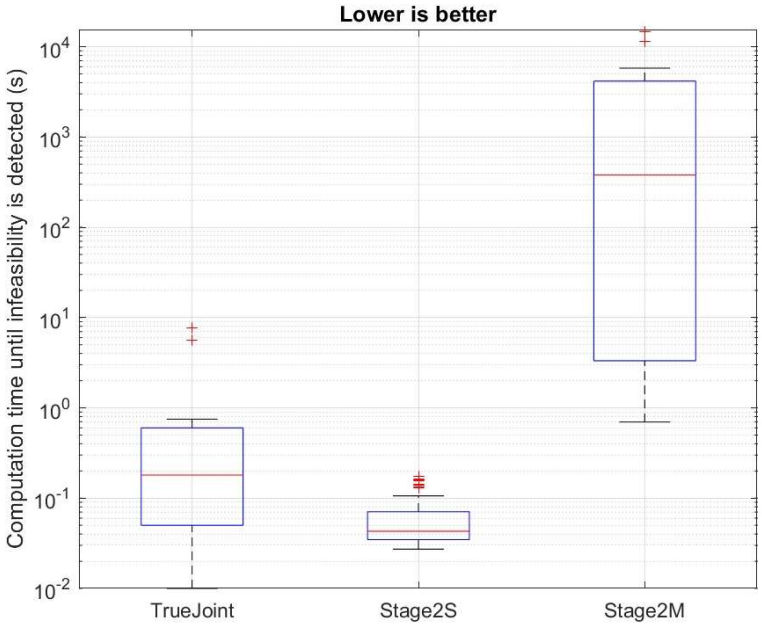


Fig. 5.31: Elapsed computation time until problem infeasibility is detected by *TrueJoint*, *Stage2S*, *Stage2M*.

The discussion above indicates that algorithms *TrueJoint*, *Joint2S*, *Joint2M* possess the qualitative features shown in Table 5.10. Ideally, we are seeking for an algorithm that possesses "Excellent" or "Very good" properties in all 3 columns of Table 5.10.

Table 5.10: Qualitative properties of proposed algorithms.

Algorithm	Scalability	Ability to detect infeasible problem fast	Solution quality (in terms of flowSpan) for feasible problems
TrueJoint	Poor	Excellent	Excellent
Stage2S	Very good	Excellent	Good
Stage2M	Very good	Poor	Very Good

Although none of the algorithms possesses all 3 attractive features on its own, an obvious hybridization emerges as follows: *Stage2M*'s real weakness is its inability to quickly detect an infeasible problem. Since *TrueJoint* can perform this detection in a few seconds (as shown in Fig. 5.31), the following scheme, termed *Hybrid*, combines the best of both worlds and achieves very good properties across all 3 columns of Table 5.10:

- As the first step of *Hybrid*, run *TrueJoint* for a duration of time T_{test} (say, 10-15 seconds). If problem is detected to be infeasible, there is no need to run *Stage2M*. The only alternative is to perform some form of TSN stream admission (since not all of them can be admitted; otherwise the problem would be feasible), like the one described in Section 5.3.2.
- If *TrueJoint* has not detected an infeasible problem at the end of T_{test}, then stop its execution and run *Stage2M* instead, since we are reasonably confident that the problem is feasible and we wish to use a lower complexity algorithm.

Despite the favorable properties of the above *Hybrid* scheme, it does not take any corrective measures when the problem in (13) is infeasible, other than quickly detecting the infeasibility and returning an appropriate error message. From the NGCO owner's or operator's point of view, this is not an ideal behavior since, even though it is not possible to serve all TSN requests concurrently under the given QoS requirements, it ***may still be possible*** to successfully serve a ***subset*** of these requests. Hence, to increase the NGCO Control Plane's robustness and to allow the system to fail more "gracefully" by serving as many TSN requests as possible, we next present a multi-objective optimization-based

formulation which allows only ***some*** of the TSN requests to be served, instead of the "all or nothing" approach followed by *TrueJoint*, leading to an alternative algorithm named *SubJoint*.

5.3.2 Robust joint scheduling and routing of time-sensitive flows

The system model for the new robust *SubJoint* algorithm is similar to that of *TrueJoint*, as described in Section 5.3.1.1, albeit with some important modifications. First, we need to introduce additional decision variables χ_r denoting the indicator variables of whether request $r \in \mathcal{R}$ is ***actually*** admitted into the network or not. All decision variables $\gamma_{r,p}$, $t^{in}_{r,f}$, $flowSpan$, $\rho^{u,v}_{r_1,f_1,p_1,r_2,f_2,p_2}$ from Section 5.3.1.1 retain their semantics for the *SubJoint* model as well. We also need to modify some of the constraints in Section 5.3.1.1 for *SubJoint*, since it is now possible to admit only a subset of the requests in $\mathcal{R}$. Specifically, equation (1) is now replaced by

$$\sum\nolimits_{p\in\mathcal{P}_r} \gamma_{r,p} \le 1, \quad \forall r \in \mathcal{R}, \tag{17}$$

since an inadmissible TSN stream will not be forwarded along ***any*** of its allowable paths in $\mathcal{P}_r$. Hence, we impose the obvious implication $\chi_r = 0 \Longrightarrow \gamma_{r,p} = 0$ for all $p \in \mathcal{P}_r$. Without loss of generality (or optimality), we also impose the following convention: any TSN streams $r \in \mathcal{R}$ ***not actually*** admitted into the network will have an injection value of zero for all of their streams' frames, i.e., we impose the rule $\chi_r = 0 \Longrightarrow t^{in}_{r,f} = 0$ for all $f \in \mathcal{F}^{cycle}_r$. This necessitates that (2), (3), (5) of Section 5.3.1.1 are converted into (18)-(20), respectively, as

$$\lfloor f/F_r \rfloor T_r \chi_r \le t^{in}_{r,f} \le \left(\left\lfloor \frac{f}{F_r} \right\rfloor + 1\right) T_r \chi_r - \sum\nolimits_{p\in\mathcal{P}_r} \max_{(u,v)\in O_p} \left(\gamma_{r,p} d^{trans}_{u,v}\right), \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r, \tag{18}$$

$$t^{in}_{r,f} - \left\lfloor \frac{f}{F_r} \right\rfloor T_r \chi_r + \sum\nolimits_{p\in\mathcal{P}_r} \gamma_{r,p} D_p \le D^{E2E}_r, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r, \tag{19}$$

$$t^{in}_{r,f} - \left\lfloor \frac{f}{F_r} \right\rfloor T_r \chi_r \le flowSpan, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{20}$$

All other constraints in Section 5.3.1.1 apply verbatim for the current model. We, again, handle the expression $\max_{(u,v)\in O_p} \left(\gamma_{r,p} d^{trans}_{u,v}\right)$ appearing in (18), (4) by introducing the auxiliary variable $\xi_{r,p} = \max_{(u,v)\in O_p} \left(\gamma_{r,p} d^{trans}_{u,v}\right)$ such that (18), and (4) are converted, respectively, to the following linear constraints:

$$\lfloor f/F_r \rfloor T_r \chi_r \le t^{in}_{r,f} \le \left(\left\lfloor \frac{f}{F_r} \right\rfloor + 1\right) T_r \chi_r - \sum\nolimits_{p\in\mathcal{P}_r} \xi_{r,p} \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r, \tag{21}$$

$$t^{in}_{r,f} + \sum\nolimits_{p\in\mathcal{P}_r} \xi_{r,p} \le t^{in}_{r,f+1}, \quad \forall r \in \mathcal{R}, \forall f \in \mathcal{F}^{cycle}_r. \tag{22}$$

By definition of $\xi_{r,p}$, we need to impose the additional constraint

$$\xi_{r,p} \ge \gamma_{r,p} d^{trans}_{u,v}, \forall r \in \mathcal{R}, \forall p \in \mathcal{P}_r, \forall (u,v) \in p. \tag{23}$$

The total number of TSN streams ***actually*** admitted into the network is $\sum_{r\in\mathcal{R}} \chi_r$. Hence, the final multi-objective MILP formulation for *SubJoint* becomes

$$\begin{aligned} &Lex_{minimize}\ (-\textstyle\sum_{r\in\mathcal{R}} \chi_r\ , flowSpan), \\ &s.t.\ (17), (19) - (22), \end{aligned} \tag{24}$$

where $Lex_{minimize}$ performs lexicographic minimization using the value of $-\sum_{r\in\mathcal{R}} \chi_r$ as the primary minimization objective (notice that the minus sign essentially turns minimization of $-\sum_{r\in\mathcal{R}} \chi_r$ into a maximization of $\sum_{r\in\mathcal{R}} \chi_r$, as intended) and the value of $flowSpan$ as the secondary objective, to be employed as a tie-breaker among solutions producing the same value for the primary objective.

Equation (24) yields a new MILP problem, which is solved by *SubJoint* using CPLEX. We note that, under the convention that inadmissible TSN streams have an "artificial" injection time of 0 for their frames (as also captured in (21)), the problem in (24) always has a feasible (albeit, not very interesting) solution of admitting no TSN streams at all. Hence, (24) is never infeasible, which addresses a limitation of *TrueJoint*. At the same time, (24) is still an NP-hard problem, so that its worst-case complexity is same as for *TrueJoint*; hence, it may seem at first that any robustness gained by *SubJoint*'s extended model

(in the sense that a feasible solution always exists) is counterbalanced by its high computational complexity. However, *SubJoint*'s worst-case high computational complexity does not become a detriment for its practical application, for the following reasons:

- *SubJoint* is ***only*** executed when the problem in (13) is determined by *TrueJoint* to be infeasible. As shown in Fig. 5.28, this is a quite rare event, as only 12 out of the 400 problem instances captured in Fig. 5.28 were actually infeasible.
- More importantly, a simulation-based study described below indicates that it is possible to use a relatively small simulation time limit for *SubJoint* and still get a solution of the same "quality" as what would be achieved if *SubJoint* was allowed to run for a much longer time. Essentially, *SubJoint* determines a "good enough" solution quite fast and then spends most of its time in attempting to find an improvement for it, which is usually not worth the extra effort.

To further clarify the last point above, we have executed *SubJoint only* for the 12 infeasible problem instances (as identified by *TrueJoint),* solving (24) with CPLEX. As mentioned, CPLEX allows a simulation time limit to be set, such that if the optimal solution has not been found up to that point, CPLEX will terminate the execution of *SubJoint* and return the best (though, not necessarily optimal) solution it has found so far.

To determine the effect of this simulation time limit on the solution quality returned by *SubJoint*, we have executed *Subjoint* for the following six time limits: 30 sec, 60 sec, 300 sec, 600 sec, 3000 sec, 6000 sec. The results, shown in Fig. 5.32, Fig. 5.33, demonstrate most convincingly the last point above. For all 12 infeasible instances, the solution obtained by *SubJoint* after 30 sec of simulation time could not be improved even when setting the simulation time limit to 6000 sec! Hence, the following heuristic approach combining *TrueJoint*, *Stage2M*, and *SubJoint* to produce a solution of "good" quality within a time limit of 1-2 minutes has emerged:

- Run *TrueJoint* for a duration of time T_{test} (say, 10-15 seconds)
  - If problem is detected to be infeasible, then run SubJoint with a simulation time limit of 30-60 sec.
  - Otherwise, if *TrueJoint* has not detected an infeasible problem at the end of T_{test}, then stop its execution and run *Stage2M*.

Based on the numerical results presented so far, this approach offers an excellent performance/complexity tradeoff for the TSN resource allocation problems that are relevant to the OCTAPUS NGCO.

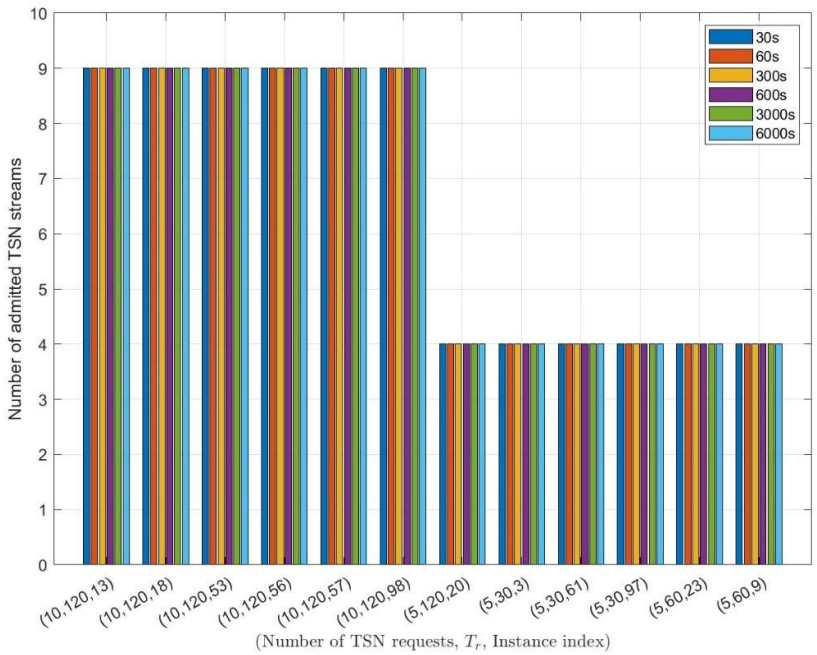


Fig. 5.32: Primary objective value (i.e., number of admitted flows) returned by SubJoint for different simulation time limits.

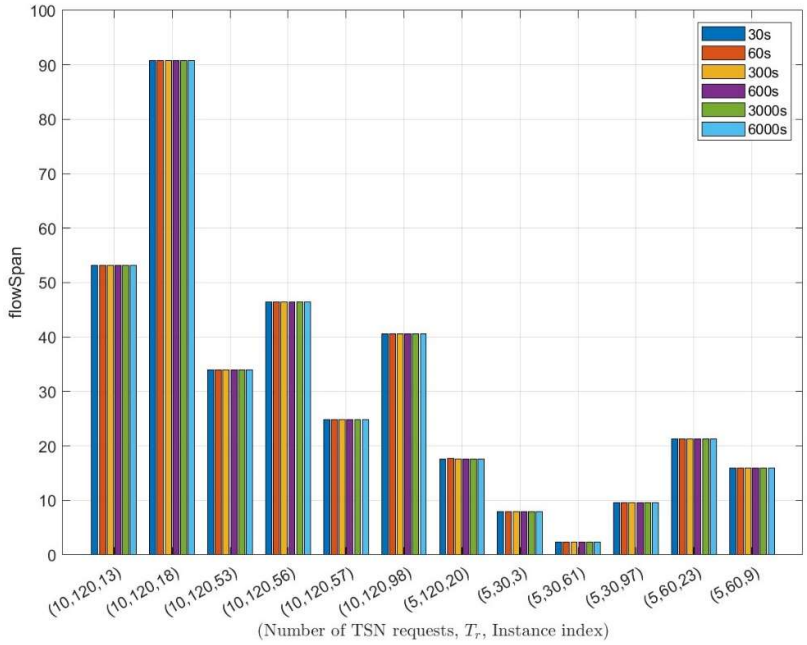


Fig. 5.33: Secondary objective value (i.e., flowSpan) returned by *SubJoint* for different simulation time limits.

6. Summary and Conclusions

This document is the final deliverable of Work Package 2 of the OCTAPUS project, focusing on the final specifications of all major components in the Data and Control planes of the OCTAPUS solution. Drawing upon the work reported in the previous Deliverable D2.2, this document reasserts the overall OCTAPUS architecture and provides the optical, electrical and mechanical specifications for the following OCTAPUS hardware components: i) the InP-based 50 Gbps transceivers and corresponding electronic drivers and TIAs, ii) the SiN-based optical backplane exploiting PCM technologies, and iii) the C/O optical diplexers responsible for implementing the OCTAPUS express paths. Information on the supported TSN features in the Uplink switches and IF cards are also provided. The hierarchical architecture of the OCTAPUS NGCO Control plane is presented, and the interactions between the SDN multi-domain coordinator and specific domain SDN controllers (such as TSN and OCS) are described.

This Deliverable also contains an extensive simulation-based study to model and evaluate the proposed OCTAPUS architecture, in anticipation of the transceiver hardware to be delivered as well as to predict the network's performance under realistic traffic conditions and operational link parameters. Specifically, at the PHY layer, the necessary Extinction Ratios and Insertion Losses of 2x2 switching units (comprising 4x4 and 8x8 Benes) to satisfy a BER requirement of 10^{-12} (at 50 Gbps) for the intra-NGCO links are determined. Moving higher up in the OSI layers, the study of the application of the CBS and TAS shapers/schedulers in a realistic urban environment with cellular-based time-varying traffic loads is performed via discrete event-based simulation, drawing important conclusions from the physical user mobility and need for careful configuration of the TAS scheduler. As a parallel line of investigation, the optimization-driven TSN flow resource allocation algorithm described in D2.2 is revisited, where the model is generalized and a carefully constructed heuristic algorithm is proposed to provide the best tradeoff between network performance and computational complexity.

The information contained in this Deliverable, in terms of architecture, functional decomposition and component specifications, has provided important guidelines for the hardware and software development work undertaken in Work Packages 3, 4 (still in progress) and led to a series of publications, to be reported in the next Project Periodic Report, while also indicating a sequence of intermediate testing and validation steps to be performed in preparation for the lab and field trials planned at the final stages of the project.

References

[1] OCTAPUS Deliverable D2.1, “Initial Report on Use Cases, System Requirements, KPIs and Network Architecture”.

[2] OCTAPUS Deliverable D2.2, “Initial Modelling of System Architecture, Component/Controller Specifications and Assembly Process Flow Definition”.

[3] OCTAPUS Deliverable D3.1, “Report on the development of 1st gen. photonic circuits including SiN PCM-based switches, lasers, EAM and PDs and 1st gen. high-speed and low-power driver and receiver electronics”.

[4] OCTAPUS Deliverable D4.2, “Initial report on the development of the network control/TSN technologies for high-bandwidth low-latency communications”.

[5] “IEEE Standard for Local and Metropolitan Area Networks – Timing and Synchronization for Time-Sensitive Applications”, https://standards.ieee.org/ieee/802.1AS/7121/ [Accessed Feb. 2024].

[6] “IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amend. 25: Enhancements for Scheduled Traffic”, https://standards.ieee.org/ieee/802.1Qbv/6068/ [Accessed Oct. 2024].

[7] “IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amend. 26: Frame Preemption”, https://standards.ieee.org/ieee/802.1Qbu/5464/ [Accessed Feb. 2024].

[8] IBM ILOG CPLEX optimizer, https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer [Online, accessed Oct. 2024]

[9] G. Sharma, W. Tavernier *et al.* “End-to-End No-wait Scheduling for Time-Triggered Streams in Mixed Wired-Wireless Networks”, *Journal Network and System Manage*ment **32**, 65 (2024).