This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016608.

BEYOND 5G – OPTICAL NETWORK CONTINUUM
(H2020 – Grant Agreement Nº 101016663)

Deliverable D4.3

Report on B5G-OPEN autonomous and zero-touch networking capabilities

Editor L. Velasco (UPC)

Contributors **UPC**

Version 2.0

Date April 15, 2024

Distribution PUBLIC (PU)

DISCLAIMER

REVISION HISTORY

Revision	*Date*	*Responsible*	*Comment*
v0	*December 1, 2024*	*L. Velasco*	*ToC*
v1	*March 29, 2024*	*All partners*	*Contributions received*
v2	*April 15, 2024*	*L. Velasco*	*Frist Integrated version*
v3	*June 22, 2024*	*L. Velasco*	*Final Reviewed Version*

LIST OF AUTHORS

Partner ACRONYM	*Partner FULL NAME*	*Name & Surname*
UPC	*Universitat Politècnica de Catalunya*	*Luis Velasco, Marc Ruiz, Salvatore Spadaro, Morteza Ahmadian, Josep Prat, Jaume Comellas*
TID	*Telefónica I+D*	*Óscar González de Dios*
UC3M	*Universidad Carlos III de Madrid*	*José Alberto Hernández, David Larrabeiti, Alfonso Sánchez-Macián, Farhad Arpanaei*
CNIT	*National Inter-University Consortium for Telecom.*	*Filippo Cugini, Kyriakos Vlachos, Alessio Giorgetti, Andrea Sgambelluri,*
CTTC	*Centre Tecnològic de Telecomunicacions de Catalunya*	*Ramon Casellas, Ricardo Martínez,* Carlos Hernández
Nokia	*Alcatel Lucent Nokia Bell Labs*	*Fabien Boitier, Petros Ramantanis, Isaia Andrenacci*

GLOSSARY

A3C	Asynchronous Advantage Actor Critic
AE	Autoencoder
AI	Artificial Intelligence
API	Application Programming Interface
ASE	Amplified Spontaneous Emission
BBR	Bandwidth Blocking Ratio
BER	Bit Error Ratio
BoDA	Bandwidth-On-Demand Allocation
CMIS	Common Management Interface Specification
CU	Cost Unit
D3QN	Dueling Double Deep Q-Learning
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
DSCM	Digital Subcarrier Multiplexing
DSLAM	Digital Subscriber Line Access Multiplexers
DSP	Digital Signal Processing
DT	Digital Twin
EDFA	Erbium Doped Fiber Amplifier
EEPROM	Electrically Erasable Programmable Read-Only Memory
FEC	Forward Error Correction
FS	Frequency Slots
GCR	Global Concurrent Restoration
GMM	Gaussian Mixture Models
GN	Gaussian Noise
GoF	Goodness-Of-Fit
HT	Holding Time
IAT	Inter-Arrival Time
ILP	Integer Linear Programming
IBN	Intent-Based Networking
IQ	In-Phase and Quadrature
ISP	Internet Service Provider
IPoWDM	IP over Wavelength Division Multiplexing
KNN	K Nearest Neighbors
kSPFF	k-Shortest Paths First Fit
MAS	Multi-Agent System
MDA	Monitoring And Data Analytics
ML	Machine Learning
NBI	Northbound Interface
NCACM	Network Configuration Access Control Model
NetP	B5G-Open Network Planner
NLI	Nonlinear Optical Interference
NN	Neural Network
OA	Optical Amplifiers
OIF	Optical Internetworking Forum
OLT	Optical Line Terminals

ONIE	Open Network Install Environment
OptC	Optical SDN Controller
P2MP	Point-To-Multipoint
P2P	Point-To-Point
PckC	Packet SDN Controller
PDF	Probability Density Function
PCA	Principal Component Analysis
PDL	Polarization Dependent Loss
PoP	Point Of Presence
PRBS	Pseudo-Random Bit Sequence
QoS	Quality of Service
QoT	Quality of Transmission
RF	Random Forest
RL	Reinforcement Learning
ROADM	Reconfigurable Optical Add / Drop Multiplexers
RSA	Routing And Spectrum Assignment
Rx	Receiver
SDN	Software Defined Networking
SLA	Service Level Agreement
SNR	Signal-To-Noise Ratio
SVN	Support Vector Machine
TAPI	Transport API
TD3	Twin Delayed Deep Deterministic Policy Gradient
TIP	Telecom Infra Project
VDM	Versatile Diagnostics Monitoring
VNF	Virtual Network Function
WDM	Wavelength Division Multiplexing
WSS	Wavelength Selective Switches
ZTN	Zero-Touch Networking

EXECUTIVE SUMMARY

Building a framework for an Artificial Intelligence (AI) / Machine Learning (ML) -assisted autonomous and dynamic network supporting real-time operations and Zero-Touch Networking (ZTN) is among the objectives of B5G-OPEN. Autonomous operation is implemented at various levels, from device to network, which requires the development of a distributed knowledge and decision-making engine that massively relies on the use of monitoring data and on the application of AI/ML

This deliverable shows a clear path towards autonomous networking, which includes the following ingredients:

1 **Deep understanding** of the network traffic, not only in terms of volume, but also its dynamicity, as well as on the requirements of services, specifically in terms of bandwidth and latency.
2 **Planning and dimensioning** of the network infrastructure, considering the latest advancements on optical technologies, like Digital Subcarrier Multiplexing (DSCM) and Point-To-Multipoint (P2MP) connectivity.
3 **Predictive maintenance**, as a management tool to reduce overprovisioning and save operational expenditures.
4 The development of **digital twining** solutions, i.e., digital representations of the optical network.
5 The **extensive use of ML**, specifically Reinforcement Learning (RL), which is one of the main techniques that enable truly autonomous ZTN.
6 **Decentralized decision making** precisely based on the deployment of RL that use network telemetry for achieving the defined goals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1 INTRODUCTION

Building a framework for an Artificial Intelligence (AI) / Machine Learning (ML) -assisted autonomous and dynamic network supporting real-time operations and Zero-Touch Networking (ZTN) is among the objectives of B5G-OPEN. Autonomous operation is implemented at various levels, from device to network, which requires the development of a distributed knowledge and decision-making engine that massively relies on the use of monitoring data.

The deliverable shows different approaches to reduce operation overheads by: (i) overprovisioning minimization though planning and orchestration that considers multi-layer complexity; (ii) predictive failure management; and (iii) simplification of network and service operations, through digital twining solutions and near real-time control of network resources.

Because several approaches presented in this deliverable are based on Reinforcement Learning (RL), **Section 2** first provides some background on the topic.

The rest of the deliverable is organized into four main sections:

- **Section 3** focuses on network planning and orchestration, and presents various key mechanisms and approaches aimed at planning the network and enabling autonomous operations through orchestration functions. Network planning is the most classical approach used by network operators to achieve cost savings in deploying network infrastructure. This section explores different approaches for that objective, but first it focuses on understanding traffic patterns, including daily fluctuations and peak traffic periods, along with expected requirements, as such information serves as input for dimensioning the network infrastructure. The first study specifically focuses on metro networks utilizing Point-To-Multipoint (P2MP) connectivity services based on Digital Subcarrier Multiplexing (DSCM). In the provisioning connectivity on top of Elastic Optical Networks (EON), a crucial challenge is the Routing and Spectrum Assignment (RSA) problem that must incorporate QoS constraints of the services, such as maximum tolerated latency and guaranteed bandwidth. Approaches based on deep reinforcement learning (DRL) demonstrate their good performance. Finally in this section, the integration of computing and networking resources is also considered, as it is pivotal for the efficient deployment of network services that meet the diversified requirements of various vertical industries (e.g., Industry 4.0, automotive). The Virtual Network Functions (VNF) placement and the establishment of lightpaths is faced using a DRL approach.
- **Section 4** moves to failure management on the optical layer and elaborates on our proposal (named MESARTHIM) for soft-failure detection, identification, localization and severity estimation. The methodology makes use of a network Digital Twin (DT) that helps in the process, and it shows remarkable anticipation in failure detection and localization by analyzing the estimation of the value of the modelling parameters of the devices. Next, failure recovery and restoration are visited, where evaluation of a DRL-based agent specifically designed for the autonomous restoration of disrupted lightpaths following an optical link failure is presented. Finally, the practical application of failure recovery is shown using the MANTRA architecture defined by the Telecom Infra Project (TIP). The architecture includes a hierarchical control plane with the B5G-OPEN parent controller on top of packet and optical Software-Defined Networking (SDN) controllers. Workflows for activating connectivity services as well as for failure

recovery are introduced and experimental results with real traffic show reduced recovery times.

- In the view of the benefits of using digital twining (DT) solutions for failure management, **Section 5** presents the development of a DT for the optical time domain. The solution, named OCATA, based on concatenating Deep Neural Network (DNN) models suited for the specific lightpath under analysis. The models are designed to run at the receiver (Rx) site, and continuously analyze lightpath's metrics to compress monitored constellation samples and detect potential anomalies.
- **Section 6** concentrates on the near real-time control. First, a solution for optical power optimization with low complexity is presented aimed at refining the knowledge of the network physical parameters. Next, at the packet layer, solutions for the detection of large flows based on hash techniques is shown in P4 scenarios aimed at performing specific operations on those flows at the data plane. Next, solutions based on DRL are presented for the control of the capacity resources in multilayer packet over optical scenarios. An agent running on top of the network nodes makes autonomous decisions near real-time aimed at minimizing the capacity allocated at the optical layer for a given packet flow. This approach consumes local packet telemetry and makes local decisions. The final work presented in this section goes much further in the decentralization of the control for near real-operation, and presents an approach based on *multi-agent systems* (MAS), where agents coordinate among them for the control of optical P2MP connectivity based on DSCM.

Finally, **Section 7** draws the main conclusions of this deliverable.

2 Background on Reinforcement Learning

In general, Reinforcement Learning (RL) is a type of AI/ML strategy in which an agent learns to behave in an environment by trial and error, that is, by making decisions and receiving positive rewards (or penalties as negative rewards). The agent is rewarded for taking actions that lead to desired outcomes and penalized when undesired outcomes occur. Over time, the agent learns to take the best actions in each situation that maximize its rewards (or minimize penalties) [JAN19, NAE20]. RL is effective for a variety of tasks in optical networks, including resource allocation (wavelengths and bandwidth), traffic engineering of flows to minimize congestion and resiliency against failures [POI07, MOM22, NAT20].

The formulation of RL problems require to define:

- A set of states *S* which are representations of the environment at a given point in time. It can be anything from a single number to a complex data structure.
- A set of Actions *A* that can be taken by the agent at a given state *s* from *S*.
- Rewards which are feedback signals that the agent receives from the environment after taking an action in a given state. Rewards can be positive or negative.
- Discounted returns: The discounted return of a state is the sum of all the rewards that the agent expects to receive in the future, discounted by a factor of γ (gamma). This value is a number between 0 and 1 that controls how much the agent values future rewards. A high gamma value means that the agent values future rewards more.
- Finally, the policy π (pi) which maps states to actions.

The goal of RL is to find a policy π that maximizes the agent's expected discounted return in the long term. The agent can do this by trial and error. It tries different actions at different states and observes the received rewards. Over time, the agent learns to take those actions that lead to higher expected discounted returns. There are many libraries with different RL algorithms already coded, both in R and Python [NAT20]. Examples, for open-source programming language R, include *contextual*, *ReinforcementLearning* and *MDPtoolbox*.

The simplest RL is *Q-learning*, which is a model-free discrete RL method that is able to learn the optimal policy represented by a Q-table of pairs <*s*, *a*>, each containing a *q* value. Being at state *s*, an action *a* is taken—either the one corresponding to the highest *q* value, or chosen randomly. After the action is implemented, it is evaluated by receiving the new state, *s'*, and the gained reward *r* from the environment. The agent then updates the corresponding *q* value in the Q-table. Q-learning works efficiently for problems where both states and actions are discrete and finite. However, it suffers from two main problems: it introduces overestimation, which leads to suboptimal policies, and the Q-table grows with the number of states. It may be appropriate to provide some more information on what is behind the Q-learning algorithm. It updates its value function based on an equation that considers the immediate reward received for an action, plus the maximum future rewards. The Q-value of a state-action pair *(s, a)* is updated as follows:

$$Q(s, a) \;\text{<--}\; Q(s, a) + \alpha\, [\, R(s, a) + \gamma\; max_{a'}\, Q(s', a') - Q(s, a)\,] \qquad (1)$$

where:

- *Q(s, a)* denotes the current estimate of the value of action *a* in state *s*.
- α (alpha) is the learning rate, determining the impact of new information on the existing Q-value.
- *R(s, a)* is the immediate reward received after taking action *a* in state *s*.
- γ is the discount factor, which balances the importance of immediate and future rewards.

- $max_{a'}$ *Q(s', a')* represents the maximum predicted reward achievable in the next state *s'*, considering all possible actions *a'*.

Q-learning excels in scenarios where the agent must learn the value of actions in different states without requiring a model of the environment. This characteristic makes it particularly adept for problems where the environment's dynamics are complex or unknown. Q-learning's straightforwardness and efficiency in learning optimal policies solely from interactions with the environment render it a versatile tool for a wide range of applications [NAE20].

Deep Q-learning (DQN) substitutes the Q-table by a feed-forward DNN that receives a continuous representation of the state and returns the expected *q* value for each discrete action. Because the DNN tends to make learning unstable, countermeasures need to be taken. The most extended measure is to keep a *replay buffer* storing the last *experiences,* i.e., tuple <*s*, *s'*, *r*, *a*> that is used to retrain the DNN. In addition, *double DQN* uses two different DNNs (*learning* and *target*) to avoid overestimation, which happens when a non-optimal action is quickly biased (due to noise or exploration) with a high *q* value that makes it preferably selected. Thus, the learning model is updated using the *q* values retrieved from the target DNN, which is just a simple copy of the learning model that is periodically updated. Finally, *dueling double DQN* (named D3QN in this paper) uses two different estimators to compute the *q* value of a pair <*s*, *a*>: (*i*) the *value* estimator, which can be intuitively seen as the average *q* value of any action taken at state *s*; and (*ii*) the *advantage* estimator, which is the specific state action-dependent component. The sum of both value and advantage components returns expected *q* values. DRL has demonstrated remarkable success in various domains, from mastering complex games to autonomous vehicle navigation and robotics, showcasing its capability to process and act upon large-scale and complex inputs effectively [AIS23, KUM22].

DQN-based methods assume a finite discrete action space. If continuous state and action spaces are required, other approaches such as *Actor–Critic* methods can be used. The main idea behind *Actor–Critic* methods is that two different types of models are trained separately: (*i*) *actors*, who are in charge of computing actions based on states, and *(ii) critics*, who are in charge of evaluating the actions taken by actors, i.e., to compute *q* values. Both actor and critic models can be implemented by means of DNNs. Among all different *Actor–Critic* methods, the TD3 method considers one single actor and two different critic models, where the minimum value from the two critics is used to learn, aiming at reducing overestimation.

3 NETWORK PLANNING AND ORCHESTRATION

This section delves into various key mechanisms and approaches aimed at optimizing network planning and enabling autonomous operations through orchestration functions. At a macro level, network planning endeavors to achieve cost savings in deploying network infrastructure, particularly in terms of devices like optical transceivers. Central to this objective is understanding anticipated traffic patterns, including daily fluctuations and peak traffic periods, along with expected requirements. This information serves as input, leveraging techniques such as ML, to effectively meet traffic demands while appropriately sizing the devices and elements constituting the network infrastructure. The study specifically focuses on metro networks utilizing P2MP connectivity services based on DSCM, capitalizing on its inherent high-capacity and flexibility advantages.

In the (on-line) network operations when provisioning/re-optimizing/restoring connectivity services, the orchestration of different underlying technologies (packet and optical) relies on the autonomous functions. To this end, it is essential supporting well-defined closed-loops functions (*observation-decision-action*) which can be realized through constructing digital network representations, i.e., Digital Twin (DT). In a nutshell, the relevant network and device information is constantly represented and retrieved (e.g., using dedicated telemetry and monitoring platform). The collected data can be conveniently analyzed and processed, and eventually digested by e.g., offline/dynamic trained AI/ML models. In this regard, diverse supervised, unsupervised and reinforcement learning (combined with Deep Learning) can be adopted depending on the targeted network operation. Applied AI/ML algorithms and models can support a broad range of autonomous decisions, such as detecting performance degradations (e.g., optical power anomalies), better fulfilling stringent requirements (e.g., latency, placement of compute network functions) of arriving network services, assisting the SDN controller to re-optimize the allocation of connections to meet Quality of Transmission (QoT) needs or attain enhanced resource utilization, etc. The following details the set of conducted network orchestration studies in B5G-OPEN to highlight the appealing achievements when orchestrating network services in an autonomous way.

3.1 PLANNING OF P2MP CONNECTIVITY

Telecommunication operators seek technologies to build their next-generation optical networks focusing on multiple dimensions, namely resiliency, flexibility, programmability and also scalability and cost-effectiveness. In this sense, P2MP coherent pluggable optics featuring digital subcarrier multiplexing (DSCM) [Wel21] can potentially provide many of such key requirements, in particular, high-capacity and flexibility. Such P2MP network architectures allow to reduce the number of transceivers and intermediate aggregation nodes leading to high-bandwidth and low-latency configurations in a cost-effective way [Bac20,Hos23,Pav22,Sko21,Nap21]. Recently, a new Multi-Source Agreement (MSA) working group called the Open XR Forum [Swe22] has emerged with the goal of fostering collaboration between industry and academia in the development of P2MP coherent pluggable transceiver technology. The forum aims to develop products and services, standardize networking interfaces, and ensure interoperability among vendors. Following [Swe22], the Open XR system concept is a new optical network architecture that allows a single central node to be interconnected with multiple leaf nodes in a P2MP tree topology, whereas passive devices (splitters and combiners) are used in between. Thanks to DSCM, different bandwidth settings can be configured between the central node and each leaf of the tree. This can simplify access-metro topologies in metropolitan area networks (MANs) and has direct applicability in hub-and-spoke traffic scenarios, which are common in MANs and 5G/6G deployments.

In particular, thanks to DSCM, the total bandwidth in the tree may be split into m Nyquist digital subcarriers (SC), typically m = 16, each one operating at 25 Gb/s, for a total bandwidth in the tree of 400 Gb/s, with m × 25 Gb/s (using 16QAM, 3.7 GBaud and dual polarization). In principle, each individual sub-carrier can be treated independently of all others, including modulation, management, and aggregation, and thus can be (software-) routed to different destinations, allowing for a greater degree of flexibility with respect to classical fixed point-to-point (P2P) pluggable transceivers [Nap22,Wel23]. Thus, if properly designed and planned, such DSCM-based P2MP trees can offer an opportunity to design cost-efficient and flexible network topologies by dynamically assigning sub-carriers to individual leaf nodes as needed. In this study, we present a variation of hierarchical clustering algorithms for designing P2MP trees that take advantage of the uncorrelated traffic profiles of end (leaf) nodes. This can lead to significant cost savings in network deployments. The proposed algorithm uses daily traffic patterns to select and group end nodes that can be most efficiently included under the same P2MP tree. This maximizes cost savings in terms of the number of transceivers. The algorithm can reduce the number of trees by up to 18%, which translates to a potential of 40 − 50% total capital expenditure (CAPEX) savings when compared to fixed P2P designs.

3.1.1 Example of a typical MAN scenario

Figure 1 shows an example of a hierarchical MAN topology with three access nodes (ACO1 to ACO3) and two metro central offices (MCO1 and MCO2). The MAN provides services to a heterogeneous number of end nodes, including residential (L4), business (L3), industrial zones (L8), sports zones (L6), and recreation and entertainment areas (e.g., L1 and L2). These areas use different technologies, such as active antenna processing units (AAUs), optical line terminals (OLTs), and digital subscriber line access multiplexers (DSLAMs). The traffic pattern within the network is dynamic and constantly varies across each area. This scenario is very suitable for a P2MP architecture where the aggregated traffic at the ACOs (equipped with 100G low-rate transceivers) is then forwarded to the MCOs (equipped with 400G high-rate P2MP transceivers), as in [Cas23].

The dynamic management of sub-carriers in the P2MP tree can be autonomously handled by either a central SDN controller or agent controllers. These controllers leverage data analytics of traffic patterns, supported by online traffic predictors located at the central controller. As a result, bandwidth-on-demand allocation (BoDA) can be efficiently provided within a reasonable timeframe. Each sub-carrier can deliver a throughput of 25 Gbps (25G), with a maximum transparent reach distance of 500 km [Hos22]. Additionally, the SDN controller is assumed to be capable of activating and deactivating contiguous subcarriers (SCs) in the 400G and 100G modules based on the BoDA strategy for both the downlink and uplink directions.

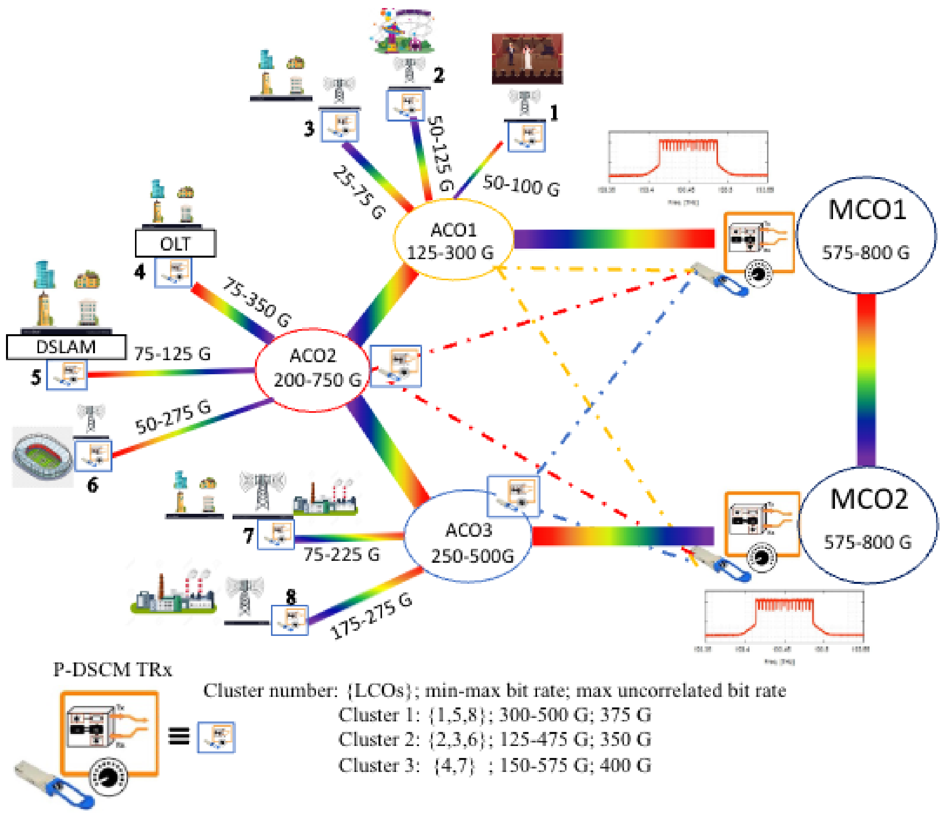

MCO1
575-800 G
MCO2
575-800 G
ACO1
125-300 G
ACO2
200-750 G
ACO3
250-500G
OLT
DSLAM
1
2
3
4
5
6
7
8
25-75 G
50-125 G
50-100 G
75-350 G
75-125 G
50-275 G
75-225 G
175-275 G
P-DSCM TRx
≡
Cluster number: {LCOs}; min-max bit rate; max uncorrelated bit rate
Cluster 1: {1,5,8}; 300-500 G; 375 G
Cluster 2: {2,3,6}; 125-475 G; 350 G
Cluster 3: {4,7} ; 150-575 G; 400 G

Figure 1: Example of a hierarchical MAN with P2MP trees.

3.1.2 Clustering example

Following Figure 1, we consider a hypothetical traffic pattern for all the leaf nodes L1-L8, which are situated within a specific area. For example, L8 is a leaf node in an industrial area with a minimum traffic of 175 Gbps (7 x 25 Gbps) and a peak traffic of 275 Gbps (11 x 25 Gbps) over a given time period. For simplicity, we assume that the upstream and downstream traffic are symmetrical. We further assume that the SDN controller is aware of the traffic pattern and have clustered the leaf nodes into three clusters. The AI/ML algorithm presented in the next section ensures that the leaf nodes are clustered together in a way that their peak traffic times are uncorrelated. This ensures that the aggregated traffic on any given link never exceeds 400 Gbps at any time of the day.

For example, in Cluster 1, which comprises leaf nodes 1, 5, and 8, the cumulative throughput could be as high as 500 Gbps if the peak traffic times of all three nodes coincide. However, this never happens because the leaf nodes are located in different heterogeneous areas, such as recreation and entertainment, business and residential, and industrial. As a result, the maximum aggregated traffic measured is 375 Gbps. Similarly, in the example, the maximum measured bitrates are 350 Gbps and 400 Gbps for Clusters 2 and 3, respectively, which are both less than the sum of the peak rates.

The next section provides an overview of AI/ML clustering techniques and how they can be adapted to find the most appropriate sets of clusters with uncorrelated traffic patterns. This can help to reduce the number of P2MP trees necessary in a hierarchical MAN.

3.1.3 On clustering leaf nodes with uncorrelated traffic profiles

We start from the hypothesis that leaf nodes exhibit different daily traffic profiles, and this can be leveraged by the inherent flexibility of DSCM transceivers to build P2MP trees in a cost-efficient manner. Essentially, the 16 x 25Gb/s channels can be assigned to clients in a dynamic fashion moving channels from one client to another as needed, according to dynamic traffic demands, thus allowing to provide extra resources to specific clients during his/her peak hour, and reassigning those resources to other clients on different times of the day, when their peak hours occur. A simplified version of this idea is shown in Figure 2, where three clients are assigned a different number of subcarriers in the morning and at night.

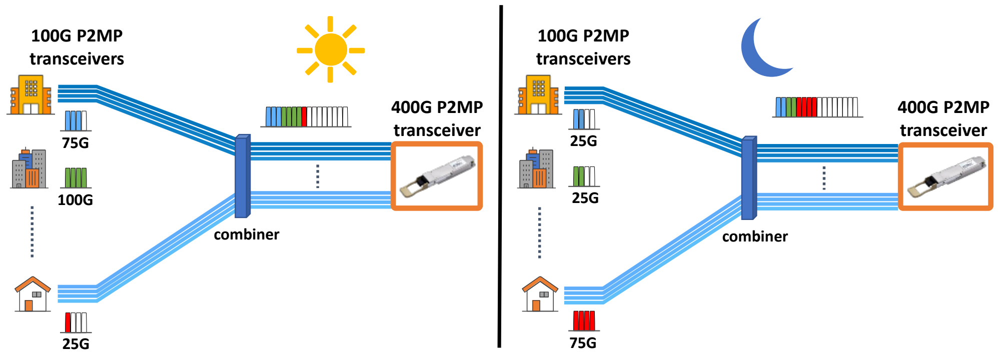


Figure 2: Dynamic allocation of subcarriers at different times of the day (left) in the morning (right) at night.

Clustering is a well-known unsupervised ML method for grouping elements based on a similarity or distance metric. It is used to reveal subgroups of similar structure within a set of unlabeled data, where each individual cluster has some homogeneity compared to the rest of the data. Essentially, clustering algorithms aim at partitioning data into groups such that the elements within the group are highly related or show minimal distance within the group, while the distance between groups is maximized.

Depending on the dataset and the application considered, the definition of a given distance is critical for the algorithm to partition the data. The most common distance metrics used in clustering techniques include cosine similarity, Euclidean, and Hamming distance, but others can be defined for specific uses. Regarding the algorithms themselves, there are several different clustering techniques; the most popular ones are hierarchical clustering and K-means, but there are others more sophisticated like partitioning Clustering, density-based clustering, etc.

Several measurement-based research studies conducted in the past have identified multiple aspects in traffic patterns. Some of them observed different behaviors between weekdays and weekends (see, e.g., [Xu16]), or the peak and valley times of the day. Typically, aggregated traffic profiles reveal the peak hour in the morning, a valley around lunchtime, and another peak after lunch; the exact time at which the peaks and valleys occur depends on the habits of the countries. In all cases, the network is usually quiet at night.

In [Xu16], the authors deeply studied the traffic patterns observed in 9,600 cell towers in the city of Shanghai, China. Among other interesting observations, they identified four different patterns of use, namely Residential, Office, Transport, and Comprehensive profiles, the latter being related to shopping centers and other leisure areas. The profiles account for 16%, 25%, 32% and 27% for Residential, Office, Transport and Comprehensive profiles respectively. These hourly traffic patterns are reproduced in Figure 3 as 1 x 24 row-vectors, representing different hours of the day.

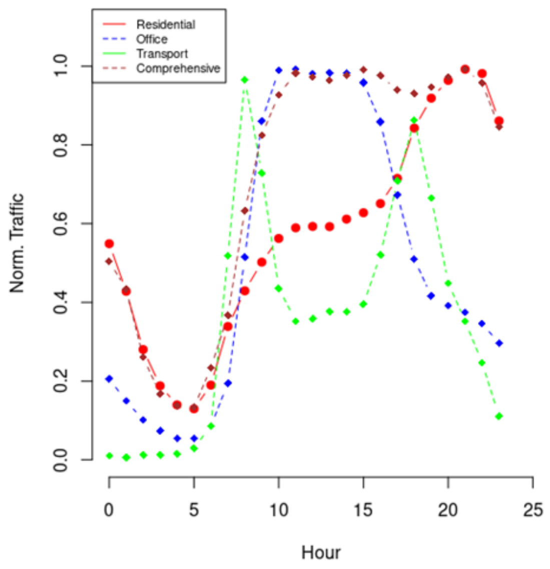


Figure 3. Normalized traffic profiles: Residential, Office, Transport and Comprehensive.

As observed, different traffic profiles exhibit peak and valley times at different times of the day. For example, the residential profile has its peak time between 8 p.m. and 23 p.m. while the office traffic profile shows the peak between 9 a.m. and 4 p.m. Interestingly, the transport traffic pattern has two peaks, one before office hours plus another one after work.

As shown, the four profiles have different peaks and valleys, and combining them wisely (using clustering) will result in a reduction of P2MP trees and equipment. In general, selecting the optimal groups/clusters whose sum is below 16 x 25G SCs (for 400 Gb/s P2MP tree configurations) can be thought of as an extension of the classical bin-packing/Knapsack problem, which is known to be NP-complete (i.e., NP stands for nondeterministic polynomial-time complete) and very hard to solve for computers as the problem grows in size. Instead, the next subsections define two clustering techniques to find groups of uncorrelated traffic profiles while satisfying the requirement that the number of SCs demanded is always below the maximum *m*=16 throughout the day.

3.1.4 CA1: Clustering of uncorrelated traffic sources

In Clustering Algorithm no. 1 (CA1), we are mostly interested in designing a clustering-based algorithm that takes into account only daily traffic profiles that benefit from the dynamic behavior of coherent P2MP pluggable transceivers. In this sense, the correlation between daily profile traffic patterns can be used as a distance metric, where two nodes with uncorrelated traffic patterns will be considered of short distances, while nodes with correlated peak times shall be avoided or penalized with a high distance value. To this end, let $d_{traff}(v_i,v_j)$ be the distance between the leaf nodes v_i and v_j with traffic patterns t_i and t_j as:

$$d_{traff}(v_i, v_j) = \frac{1 + corr(t_i, t_j)}{2} \tag{2}$$

Essentially, the minimum distance between nodes nodes v_i and v_j (null distance) occurs when their traffic patterns are totally uncorrelated (i.e. correlation = -1) and the maximum distance (distance = 1) occurs when their traffic profiles are totally correlated (correlation = 1). Here, the traffic pattern for a node is a 1 x 24 row vector whose elements represent the hourly traffic volume offered throughout the day.

3.1.5 CA2: Clustering of uncorrelated traffic sources and short geographical distance

In a second version, the clustering algorithm (CA2) must find groups of nodes that favor uncorrelated traffic patterns, but the nodes participating in the same group must be as close as possible in terms of geographical distance. To take into account both aspects, we define a new distance metric $d_{tot}(v_i,v_j)$:

$$d_{tot}(v_i, v_j) = \alpha\, d_{tot}(v_i, v_j) + (1-\alpha) d_{geom}(v_i, v_j) \qquad (3)$$

Here, $d_{tot}(v_i,v_j)$ takes into account both distance metrics: traffic-based and geographical distance. The latter one is computed based on the GPS coordinates of the nodes, normalized among all nodes, that is, the two furthest apart nodes have a geographical distance equal to one. With this approach, α is a weighting value (α in [0,1]) that allows the network planner to give more importance to traffic (α close to 1) or to the location (α close to 0) of the nodes participating in each P2MP tree. In scenarios where the network operator has some freedom to interconnect distant-apart nodes, then α can be closer to 1 allowing the clustering algorithm to better exploit uncorrelated traffic patterns among the total number of nodes.

3.1.6 Simulations and results

As shown in Figure 3, different traffic profiles have different busy moments and quiet hours; in general, the peak-to-average values range between 2 and 3. Following this behavior, we have simulated 1,000 nodes with random traffic profiles and uniformly distributed random peak values *U(10, 100)* Gb/s plus a percentage of random traffic values (10% and 30% of peak traffic) to create a diverse dataset with different traffic profiles and demand volumes. Figure 4 shows an example of the resulting traffic offered by four simulated nodes (in Gb/s) at different times of the day for both 10% and 30% traffic variability with respect to the normalized profile. As shown, the 30% case shows more traffic fluctuations than the 10% case and should be considered a worst-case scenario of nodes with high traffic fluctuations.

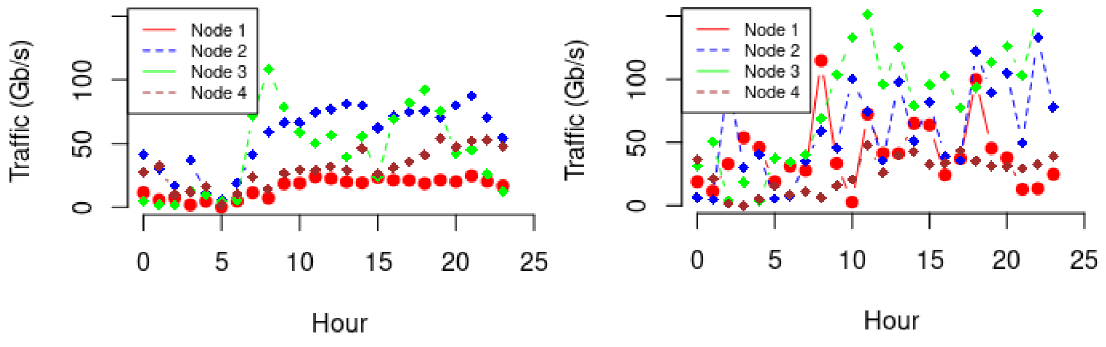


Figure 4. Daily traffic profiles with low traffic variability (10%) and high-traffic variability (30%).

The scenarios, algorithms and simulations have been programmed using R.

3.1.7 Simulation #1: without geographical coordinates

In this first simulation, we find the best combination of nodes that minimize the number of P2MP trees, taking into account only the traffic profile (i.e., $\alpha = 1$) regardless of their location. The 1,000 simulated nodes are grouped into 171 clusters (i.e., P2MP trees), that is, an average of 5.85 nodes per cluster.

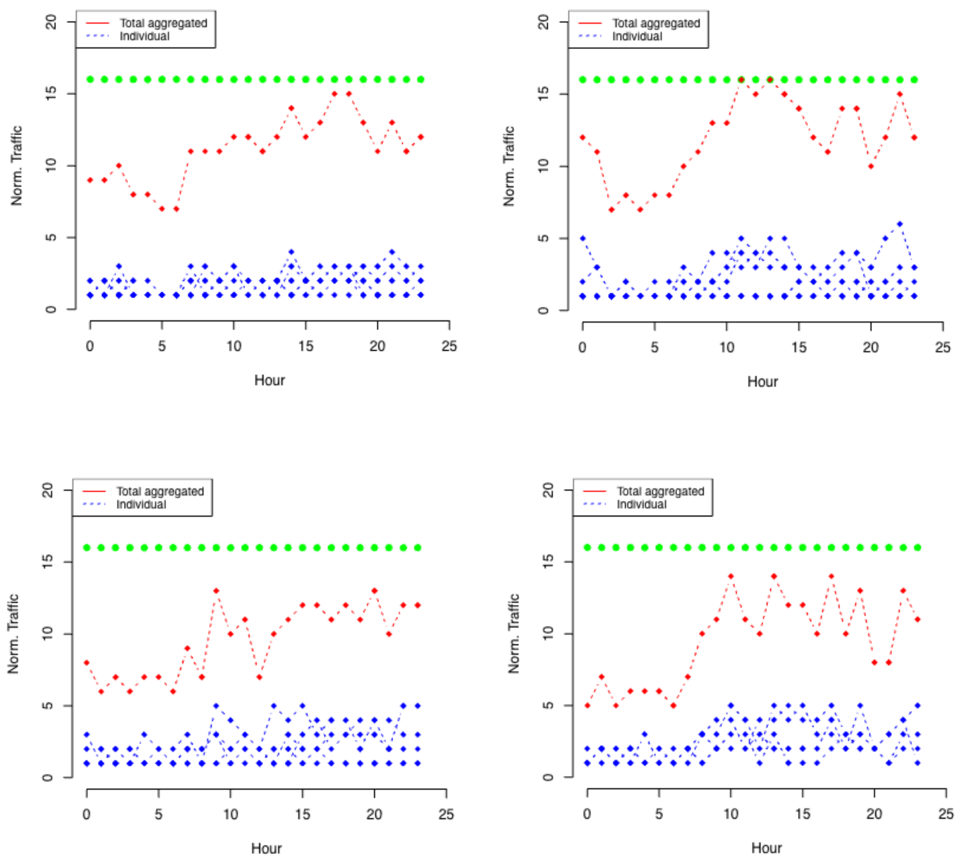


Figure 5. Daily traffic profiles for clusters/trees no. 1, 2, 3, and 4 for simulation case 1.

Figure 5 shows four different clusters/trees. Essentially, the blue lines represent the traffic of each leaf node in the P2MP tree (not in Gb/s but in the number of subcarriers of 25G) at different times of the day; the red line is the sum or aggregated number of subcarriers for all nodes in the same P2MP tree arriving at the hub node. As shown, the uncorrelated nature of individual flows is leveraged to reach the maximum number of SCs at different times of the day. As shown, the summation of the number of subcarriers of individual leaf nodes does not exceed the limit of 16 subcarriers offered by the hub node at any time of the day. The first cluster depicted in Figure 5 comprises seven P2MP leaf nodes of the simulation. The second, third, and fourth clusters include 7, 5, and 4 leaf nodes respectively. Remark that the figure shows examples of only 4 out of 171 clusters or trees decided by the clustering algorithm for the 1,000 nodes.

Table 1 further details the number of SCs required by each P2MP leaf node in cluster/tree no. 2 at different times of the day Figure 5 (top) and the total sum of SCs needed to satisfy all traffic demands. As shown, the demands for 25G SCs vary significantly from one hour to the next, highlighting the benefits of having dynamic bandwidth allocation of 25G modules over different times. For example, in this cluster, at hour h12 (i.e., 12 noon), node 3 needs to jump from 3 SCs up to 5, while node 1 can reduce from 4 to 3. On the other hand, node 7 typically needs only 1 SC except at some hours of the day when it requires 2 SCs, but these are valley hours for other nodes. In all cases, the total number of SC never exceeds *m*=16.

Table 1 shows the number of P2MP trees created by CA1 for different values of the ranges of peak traffic and the variability of the traffic with respect to the peak. The comparison is against a largest-fit heuristic assignment algorithm which assigns nodes to clusters by sorting them from largest to smallest demands and filling the tree until the maximum number of SCs is reached. As shown, our clustering algorithm CA1 always creates a number of P2MP trees that is considerably smaller than such a largest-fit heuristic, with tree savings up to 18% in many cases.

Table 1 Simulations CA1 in 1,000 node networks

Peak (Gb/s)	**Variability**	**CA1**	**Largest-fit**	**Savings**
U(10,100)	30% of peak	172 trees	183 trees	6.0%
U(50,100)	30% of peak	220 trees	250 trees	12.0%
U(10,150)	30% of peak	257 trees	278 trees	7.6%
U(10,100)	80% of peak	250 trees	278 trees	5.8%
U(50,100)	80% of peak	333 trees	397 trees	16.1%
U(10,150)	80% of peak	440 trees	520 trees	18.2%

As shown in Table 1, when traffic variability is very high, the number of P2MP trees required is between 250 and 440, more than double than in case with less variability (172 to 257 trees). Still, the largest savings in the number of trees is achieved when traffic demands are highly variable (80% variability in the table) since the clustering algorithm outperforms at finding uncorrelated traffic profiles and better squeezes the dynamicity of P2MP technology.

3.1.8 Simulation #2: with GPS coordinates

In this second scenario, we introduce GPS coordinates to the nodes, forcing the clustering algorithm to also take into account the location of the nodes to find a balance between the nodes with uncorrelated traffic patterns and their geographical distance. The peak traffic for each node is randomly selected from a uniform distribution *U(10,100)* Gb/s and the variability is 80% of the peak in all cases. In this case, networks with 1,000 nodes are simulated, again compared with the largest-fit heuristic explained in the previous section.

The nodes are randomly located in a grid with different sizes, as specified in Table 2. This table also shows the number of trees obtained by CA2 and the savings with respect to the largest-fit heuristic for different values of α. The α parameter has a critical impact on savings since, as observed, it allows a maximum of about 5.6% savings regarding the number of P2MP trees down to 0% savings if the clustering algorithm weights zero (i.e., uncorrelated traffic patterns is not relevant) and only the minimum geographical distance is considered.

Table 2 Simulations CA2 in 1,000 node networks

Peak (Gb/s)	Variability	α	CA2	Largest-fit	Savings
U(10,100)	80% of peak	1	162 trees	172 trees	5.8%
U(10,100)	80% of peak	0.75	163 trees	173 trees	5.2%
U(10,100)	80% of peak	0.5	164 trees	172 trees	4.6%
U(10,100)	80% of peak	0.25	169 trees	172 trees	1.7%
U(10,100)	80% of peak	0	172 trees	172 trees	0%

3.1.9 CAPEX savings: P2MP trees vs fixed P2P transceivers

Next, we compare the P2MP architecture with CA2 dimensioning against a hypothetical dimensioning employing P2P pairs of transceivers interconnecting access nodes directly to aggregation nodes. In such a fixed P2P setting, no dynamic allocation/sharing of bandwidth can be leveraged, and each P2P link must be dimensioned to the peak hour.

Following the cost model of [Hos22], we consider the following normalized cost units (CU) for each P2P transceiver: C_{10G}=1 CU, C_{100G}=5 CU, C_{200G}=8 CU, C_{400G}=12 CU for 10G, 100G, 200G and 400G fixed P2P transceivers respectively. For the cost of P2MP trees, we assume that high bitrate 400G P2MP transceivers have the same cost as in fixed P2P, that is 12 CU, while low-bitrate transceivers cost one half, that is 6 CU, in line with [Hos22].

Figure 6 shows an evolution of total cost (in CU) for interconnecting 1,000 nodes, either using P2MP or P2P technologies. The bars represent the total cost when each leaf node offers a traffic volume uniformly distributed *U(10,50)* (30 Gb/s on average), then *U(50,100)* (75 Gb/s on average) and finally *U(100,150)* (125Gb/s on average). These 30, 75, and 125 Gb/s represent traffic values of ACOs serving small neighborhoods (say 10,000 households offering 3 Mb/s on average per household [Her19], and their evolution in the short term (after three years) and medium term (after five years). As shown, the total cost of using P2MP technology is always smaller than that of P2P technology.

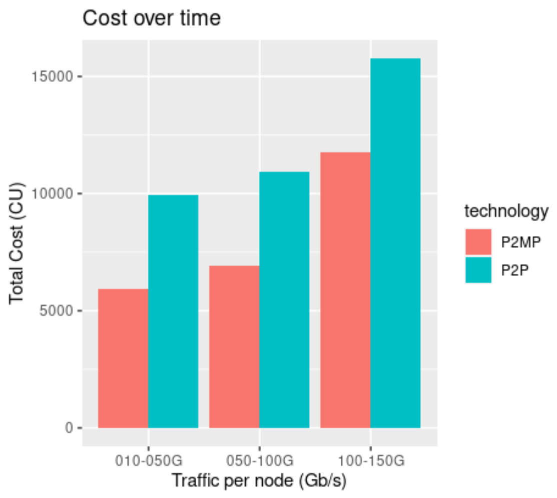


Figure 6. Total cost evolution in the short and medium term for 1,000 nodes: P2MP vs P2P technology.

Thus, in conclusion, we have shown the applicability of Machine-Learning based clustering techniques to find groups of nodes whose aggregated traffic demands are best suitable for m x 25 Gb/s P2MP topologies featuring digital subcarrier multiplexing. The clustering algorithm takes into account both uncorrelated daily traffic patterns and distance between the nodes and finds groups of nodes for building point-to-multipoint trees with low computational complexity. This algorithm has been tested on different simulated scenarios showing that the clustering algorithm allows about 10% savings in the number of trees with respect to other tree assignment heuristics.

Finally, when the tree clustering dimensioning algorithm is compared against a network plan based on fixed P2P transceivers, important equipment reductions are achieved thanks to the dynamic bandwidth assignment allowed in point-to-multipoint trees. This implies capital expenditure savings between 40 and 50% approximately with respect to fixed P2P transceivers dimensioned to support peak traffic.

3.2 Reinforcement Learning Based Routing For Packet-Optical Networks With Multilayer Measurements

Path optimization for traffic flows is a method available to enhance the quality of experience (QoE) perceived by users. Network automation facilitates this goal by monitoring and collecting telemetry information and network states for both optical and packet-based data. Advanced AI/ML, or other intelligent algorithms are then applied to evaluate network performance. Subsequently, decisions are made, and actions are taken over the network to prevent or correct possible performance issues that affect perceived QoE. This process is commonly referred to as closed-loop automation. Network automation is supported by various processes, including the implementation of SDN, aimed at moving towards a zero-touch network and service management approach [GALL22]. While having valid and up-to-date information is important, choosing the appropriate intelligent model to detect and correct performance problems allows for making the best decisions to optimize network operation.

In this section, we focus on the second stage of zero-touch networking: optimal path decision based on both optical and packet-based telemetry information, using the well-known RL methodology, which enables optimal network configurations by allowing the control plane to learn from its interactions with the network and make decisions without human intervention. In the past, RL has been proposed to enable ZTN by providing the network with the ability to learn from its own experience and make decisions without human input [IAC22]. Several good surveys in this area are available [BAR20, HER23, MAM19].

In particular, we provide a methodology for generating rewards in an RL environment, where such rewards are based on both optical telemetry information (i.e., pre- Forward Error Correction (FEC) Bit Error Rate (BER)) and packet routing measurements (i.e., latency and queue occupation). Open-source code is also provided for the interested reader willing to replicate the experiments and incorporate new features into the algorithm [GAR20]. We use the *igraph* library for building network topologies and an implementation of the *Q-learning* algorithm for finding the optimal routing policy in a packet-optical network where a Path Computation Elements (PCE) decides the best route selection for every source-destination pair, using both optical metrics (measured pre-FEC bit error rate) and packet latency measurements (including propagation delay and link load).

The decision to employ Q-learning in addressing optical network routing challenges is grounded in its suitability for environments characterized by dynamic and stochastic nature. Optical networks, with their intricate configurations and the necessity for real-time adaptability, present a problem space where the state and action spaces can be vast and complex. Q-learning's model-free property allows for direct learning from the environment, making it ideal for optimizing routing decisions in the face of fluctuating network conditions without the need for a predefined model of the network's behavior. This adaptability is crucial for enhancing resource allocation, minimizing congestion, and improving fault tolerance within optical networks. The code is publicly available in Github for further developments by the research community [GAR20].

3.2.1 Detecting convergence episode in Reinforcement Learning

An essential aspect of RL that enhances the efficiency and reliability of algorithms, such as Q-learning, is the concept of convergence. Convergence in the context of RL signifies the point at which the algorithm's learning stabilizes, and its predictions (or decisions) no longer undergo significant changes. This stability is crucial for ensuring that the algorithm can reliably guide decision-making processes, such as optimal routing in network topologies.

The detection of convergence is pivotal for several reasons:

- Reliability: It signals that the algorithm has sufficiently explored and exploited the environment to make consistent decisions.
- Efficiency: It helps in determining the point beyond which further training might not yield significant improvements, thus saving computational resources.
- Benchmarking: Knowing when an algorithm converges is essential for comparing the effectiveness of different strategies or parameter settings.

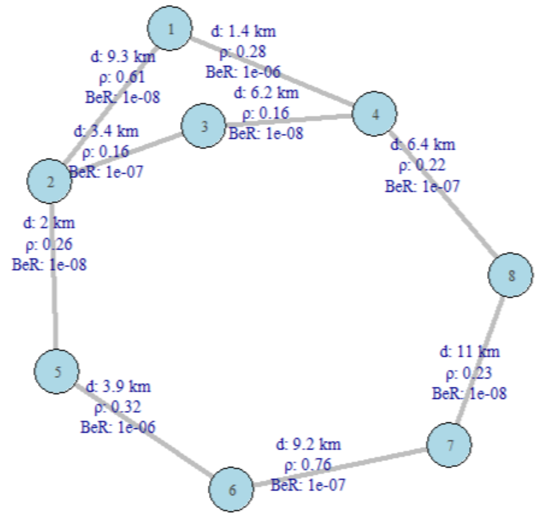
To detect convergence in our project, we employ a function that analyzes the progression of Q-values across episodes to identify the point at which these values stabilize. Here's a detailed explanation of the convergence detection process implemented in the code:

- Threshold Setting: A threshold value is predefined to determine the acceptable level of variation in Q-values for the algorithm to be considered as having converged. In our case, this threshold is set to 0.01, indicating that changes in the Q-values below this level signify convergence.
- Q-table Analysis: The function starts analyzing Q-tables from the 21st episode onwards. This delay allows the algorithm some initial episodes for exploration and learning before checking for convergence.
- Previous Q-tables Comparison: For each current Q-table, the function retrieves the 20 previous Q-tables and calculates their average. This moving average approach helps in smoothing out fluctuations and focusing on the overall trend in the Q-values.
- Convergence Criterion: The convergence is detected by comparing the current Q-table's sum of Q-values against the average sum of the previous 20 Q-tables. If the squared difference between these two sums is less than or equal to the threshold, the algorithm is considered to have converged.

3.2.2 Simulations and RL algorithm

Example on a small network topology

In the context of our study, we examine a network topology as depicted in Figure 7, which is structured around 8 interconnected nodes and 9 links. This topology is representative of a simplified model of an optical network, where nodes can be considered as switches or routers, and links represent the physical or logical connections between these nodes.



1
d: 1.4 km ρ: 0.28 BeR: 1e-06
d: 9.3 km ρ: 0.61 BeR: 1e-08
d: 6.2 km ρ: 0.16 BeR: 1e-08
3
4
d: 3.4 km ρ: 0.16 BeR: 1e-07
2
d: 6.4 km ρ: 0.22 BeR: 1e-07
d: 2 km ρ: 0.26 BeR: 1e-08
8
d: 11 km ρ: 0.23 BeR: 1e-08
5
d: 3.9 km ρ: 0.32 BeR: 1e-06
d: 9.2 km ρ: 0.76 BeR: 1e-07
7
6

Figure 7. Eight-node topology example with link metrics

Furthermore, we assume that all nodes report telemetry measurements regularly to the control plane, in which our RL algorithm is running to decide best routing strategies between nodes. These measurements refer to the data collected by the nodes regarding the operational status of each link.

In particular, each node reports: measured pre-FEC BER, which quantifies the error rate of bits on a link before FEC is applied; and link load, denoted as BER_i and ρ_i for the *i*-th link, which represents the utilization of the link's capacity. Monitoring link load is essential for avoiding congestion and ensuring efficient use of network resources. This information will be considered along the link distance d_i in kilometers, since this can affect signal quality, latency, and the need for signal amplification or regeneration.

Such monitored data (BER_i, ρ_i, and d_i) serve as input to the RL algorithm operation within the control plane. The RL algorithm utilizes this data to dynamically adjust routing decisions, aiming to optimize network performance by minimizing negative impacts such as errors and congestion. Specifically, the algorithm assigns negative rewards (penalties) based on the reported metrics, encouraging strategies that lead to lower BER, optimal load distribution, and efficient path selection considering distance. The formulation of penalties could be a function of these metrics, reflecting the cost associated with choosing a link under current conditions, as it follows:

- Propagation delay adds a penalty of d_i x 5 μs/Km, that is, the classical 5 μs/Km signal propagation latency per kilometer of silica fiber.
- Traversing a given link also adds a latency penalty of 1 μs x $(1-\rho_i)^{-1}$, which is the average transmission and queuing delay of a 1250-byte packet transmitted over a 10 Gb/s link with load ρ_i (for a classical M/M/1 queue).
- Monitored Pre-FEC BER adds a penalty of 1000 μs if the BER value of that link is 10^{-4} or above; 50 μs if the link's BER is in the range between $10^{-5} < BER_i < 10^{-4}$; or a 0 μs penalty otherwise.

As shown, traversing each link adds both packet-based penalty, propagation-delay penalty, and optical pre-FEC BER-related penalty to either encourage or discourage links in a path. This set of rules is crafted as Reward matrices for taking action *a* in state *s* (i.e. *R(s,a)* state-action pair), and inputs the Q-learning algorithm. Finally, node connectivity is also included as a bi-dimensional Matrix *P(s,s')* which contains the probability of jumping from state (or node) *s* to *s'*.

Table 3 Primary Optimal Path Selection under Normal Conditions

	Destination							
Source	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	-	1-2	1-4-3	1-4	1-2-5	1-2-5-6	1-4-8-7	1-4-8
2	2-1	-	2-3	2-1-4	2-5	2-5-6	2-5-6-7	2-1-4-8
3	3-4-1	3-2	-	3-4	3-2-5	3-2-5-6	3-4-8-7	3-4-8
4	4-1	4-1-2	4-3	-	4-1-2-5	4-1-2-5-6	4-8-7	4-8
5	5-2-1	5-2	5-2-3	5-2-1-4	-	5-6	5-6-7	5-2-1-4-8
6	6-5-2-1	6-5-2	6-5-2-3	6-5-2-1-4	6-5	-	6-7	6-7-8
7	7-8-4-1	7-6-5-2	7-8-4-3	7-8-4	7-6-5	7-6	-	7-8
8	8-4-1	8-4-1-2	8-4-3	8-4	8-4-1-2-5	8-7-6	8-7	-

Table 4 Secondary Optimal Path Selection under Degraded Conditions

	Destination							
Source	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	-	1-2	1-4-3	1-4	1-2-5	1-2-5-6	1-4-8-7	1-4-8
2	2-1	-	2-3	2-1-4	2-5	2-5-6	2-5-6-7	2-1-4-8
3	3-4-1	3-2	-	**3-2-1-4**	3-2-5	3-2-5-6	**3-2-5-6-7**	**3-2-1-4-8**
4	4-1	4-1-2	**4-1-2-3**	-	4-1-2-5	4-1-2-5-6	**4-1-2-5-6-7**	4-8
5	5-2-1	5-2	5-2-3	5-2-1-4	-	5-6	5-6-7	5-2-1-4-8
6	6-5-2-1	6-5-2	6-5-2-3	6-5-2-1-4	6-5	-	6-7	**6-5-2-1-4-8**
7	**7-6-5-2-1**	7-6-5-2	**7-6-5-2-3**	**7-6-5-2-1-4**	7-6-5	7-6	-	**7-6-5-2-1-4-8**
8	8-4-1	8-4-1-2	**8-4-1-2-3**	8-4	8-4-1-2-5	**8-4-1-2-5-6**	**8-4-1-2-5-6-7**	-

Table 3 shows the optimal routing policy decided by our RL algorithm for the 8-node topology of Figure 7. The policy finds the best next hop and primary path from source to destination taking into account the rewards for a given pre-FEC BER, propagation delay, and link load. In the example of Figure 7, all links operate with good quality optical links, i.e., pre-FER under 10^{-5}, hence only propagation delay and link load contribute to finding the best primary end-to-end path. However, if the optical quality of a link degrades (Table 4), then the RL algorithm finds an alternative or secondary route. This is the situation observed in the table when links 3-4 and 7-8 experience degraded pre-FEC BER. As shown, the RL algorithm finds new routes that avoid the use of such low-quality links (marked in bold font).

Extended example on a medium topology: Tokyo MAN

Figure 8 shows the 23-node MAN topology for Tokyo [TAC23] for testing our algorithm.

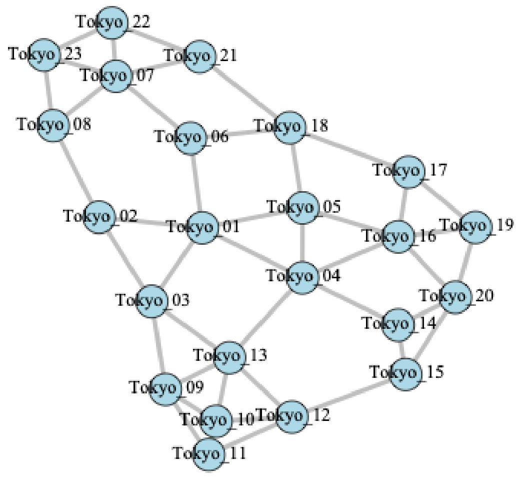


Figure 8. Tokyo Topology

In this detailed examination, given the extensive scale and intricate nature of the network topology, our analysis will be concentrated on a select number of routing paths rather than the entire network. Key routes, including but not limited to the journey from Router 1 to Router 22, will be scrutinized. The optimal paths for these specific routes under normal conditions (primary) are depicted in Table 5, that also includes secondary routes after degradation of links 1-6, 1-4, and 10-11.

Table 5 Optimal Policy: primary (in normal operation) and secondary (after degradation of links 1-6, 1-4 and 10-11 for the Tokyo Topology)

From	To	Primary	Reward	Secondary	Reward
1	22	1-6-7-22	-123.55	1-5-18-21-22	-151.00
4	7	4-1-6-7	-121.88	4-5-18-21-7	-157.61
4	11	4-13-10-11	-121.01	4-13-12-11	-122.83
1	19	1-4-16-19	-101.81	1-5-16-19	-108.21

To further validate the efficiency and adaptability of our Q-learning algorithm, we have incorporated a convergence analysis for the Tokyo MAN topology. The convergence graphs (before and after degradation of links), as shown in Figure 9 (left) and Figure 9 (right) illustrate the algorithm's progression towards a stable state of learning across episodes. This graph is critical for understanding how quickly and effectively our algorithm adapts to the complex network environment of the Tokyo topology.

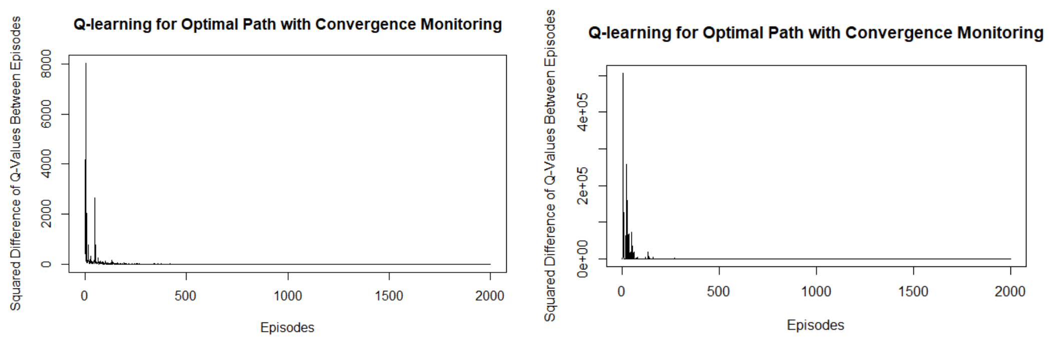


Figure 9. Convergence of RL algorithm before (left) and after (right) link degradation for the Tokyo topology

The Y-axis represents the squared difference of Q-values between episodes, which is an indicator of how much the Q-values are changing between successive episodes of learning. The X-axis shows the number of episodes, which are iterations of the learning process where the agent interacts with the environment and updates its Q-values. Let us provide more information on the first convergence graph:

- Initial Learning Phase (Episodes 0-100): The plot starts with a sharp peak, indicating significant changes in the Q-values between the initial episodes. This behavior is typical in the early stages of Q-learning as the algorithm is exploring the environment and updating its knowledge base with new experiences.
- Rapid Convergence Phase (Episodes 100-585): After the initial peak, there is a rapid decline in the squared difference of Q-values, suggesting that the algorithm is quickly learning from its interactions with the environment. The steep slope of the curve reflects the Q-learning algorithm's efficiency in adapting its policy based on the received rewards.
- Convergence Point (Episode 585): The plot levels off around episode 585, marking the convergence point of the algorithm. At this stage, the changes in Q-values have decreased to a level below the predefined threshold, indicating that the algorithm's policy has stabilized and is consistently making similar decisions across episodes.
- Post-Convergence Phase (Episodes 585-2000): Beyond episode 585, the graph shows minimal fluctuations around what appears to be a horizontal line close to zero. This horizontal trend indicates that the Q-learning algorithm's Q-values have stabilized and that the agent's policy is no longer changing significantly with further learning. The algorithm has effectively learned the optimal policy given the environment and the reward structure it has been interacting with.

The same analysis goes for the second graph, in which the convergence point occurred at Episode 497.

Extended example on a large topology: Milano MAN

Let's now take a look to an even larger topology. Figure 10 shows a 52-node MAN topology that closely represents the city of Milano.

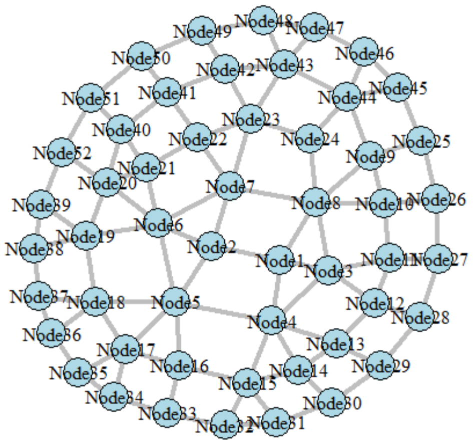


Figure 10. Milano topology

In this case, given the even more extensive scale of the network topology, our analysis will be again concentrated on a select number of routing paths rather than the entire network. The optimal paths for these specific routes under normal conditions (primary) are depicted in Table 6 that also includes secondary routes after degradation of links 1-6, 1-4, and 10-11.

Table 6 Optimal Policy: primary (in normal operation) and secondary (after degradation of links 5-6, 7-23, 22-41, 29-30 for the Milano Topology)

From	To	Primary	Reward	Secondary	Reward
5	22	5-6-7-22	-121.45	5-18-19-20-6-7-22	-248.52
7	50	7-22-41-50	-1944.66	7-6-21-40-51-50	-2121.94
13	32	13-29-30-31-22	-135.63	13-4-5-16-15-32	-176.65
44	52	44-43-23-7-6-20-52	-221.95	44-9-8-1-2-6-20-52	-296.21

In this case, we can also further validate the efficiency and adaptability of our Q-learning algorithm with a convergence analysis. The convergence graphs (before and after degradation of links) in this case are the ones delineated in Figure 11. Before degradation, convergence occurred at Episode 1270; and after degradation it happened at Episode 1331.

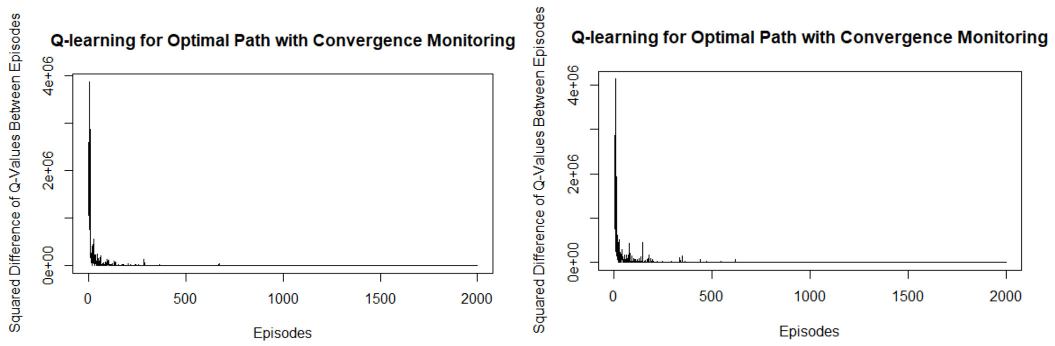


Figure 11. Convergence of RL algorithm before (left) and after (right) link degradation for the Milano topology

As shown, RL algorithms are very suitable for finding optimal routing policies taking into account different physical and optical metrics (link distance and pre-FEC BER) and congestion-based ones (link load). The RL algorithm is open-source and can be modified to enforce finding optimal paths based on other metrics, just by assigning rewards (or penalties) to different links based on speed or power consumption. Future work will include the application of the algorithm to other topologies that incorporate different segments of the network and also adding other QoE parameters as inputs for the algorithm.

3.3 LATENCY CONSTRAINED SERVICES

EONs have emerged as a sophisticated backbone infrastructure, pivotal for accommodating the burgeoning demands imposed by next-generation services, such as B5G, cloud computing, and the Internet of Things (IoT). These networks are tasked with fulfilling increasingly stringent requirements concerning bandwidth, latency, and reliability to support a wide range of applications. Therefore, EONs require flexible and efficient management of optical fiber spectrum resources.

A crucial challenge addressed in EONs is the RSA problem, where the objective is to optimize the path and spectrum allocation for lightpath requests while adhering to QoS constraints, such as maximum tolerated latency and guaranteed bandwidth. This problem's complexity is heightened by the need for a careful balance between efficient spectrum utilization and satisfaction of stringent end-to-end service requirements.

DRL has proven its effectiveness in the context of RSA in EONs [Ch19]. The introduction of DRL-based solutions represents a significant shift from traditional heuristic approaches to a more dynamic, intelligent, and autonomous system of network control and operation. In this context, DRL agents learn to make RSA decisions through interactions with the network environment. These agents aim to maximize a cumulative reward that encapsulates both the efficiency of spectrum utilization and adherence to service QoS requirements.

In this section, DRL-based strategies focusing on latency-aware RSA are presented and validated, revealing a promising approach to enhance the dynamism and responsiveness of EONs in facilitating diverse network services [Her23, Her22b].

3.3.1 SDN Control Plane

In the considered EON, a SDN controller oversees the entire service lifecycle, handling new service demands, resource allocation, and network element configuration. Connectivity services are requested through an API via the northbound interface. The SDN controller then coordinates with an external entity for path computation. Then the computed path is sent back to the controller, which executes the resource allocation, which includes programming transponders and ROADMs through southbound interfaces. The external entity performing RSA decisions is aided by machine learning models, and all network state information is stored in the Transport API (TAPI) "Context" for the path computation entity's use. Figure 12 illustrates the architecture and components of the SDN control plane.

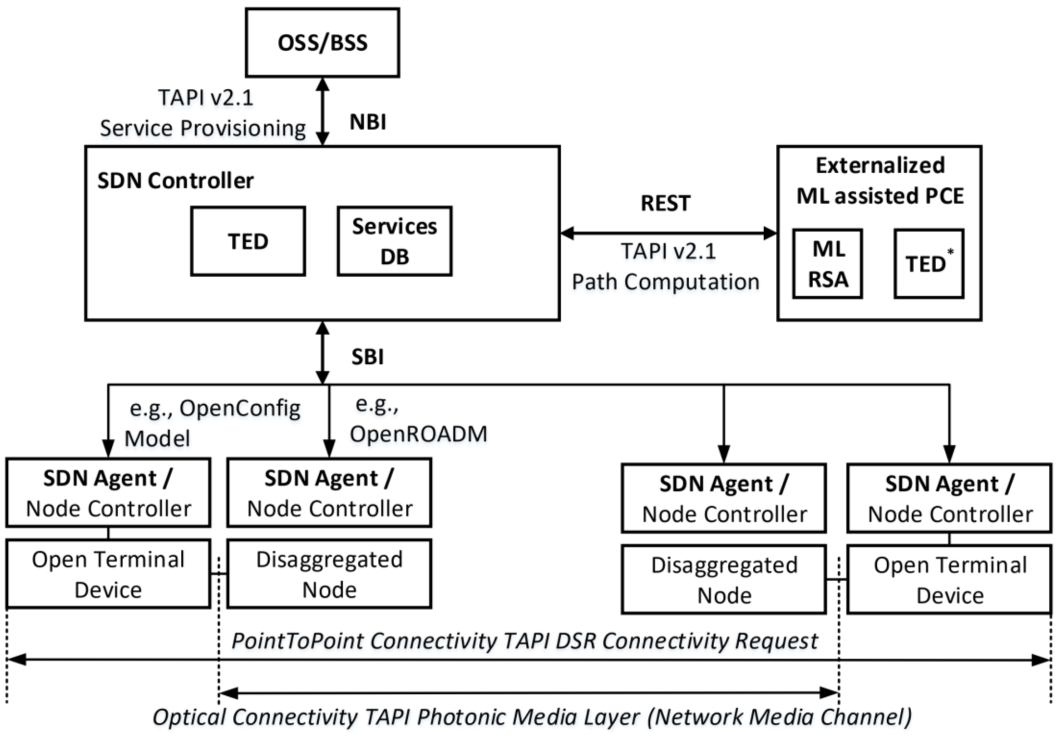


Figure 12. SDN control plane

3.3.2 DRL-based latency-aware RSA

Upon receiving a path computation request, the externalized ML-assisted PCE, in this case the DRL agent, retrieves the EON's current state to create a precise *state* representation. Then, this representation is used by DNNs to decide RSA actions. These decisions aim to fulfill connectivity service requests by identifying feasible paths (encompassing nodes, links, and frequency slots) while considering the network's current conditions and the bandwidth and latency requirements of the service.

The network state used by the DRL agent is formed by several critical components that define the current condition and operational parameters of the EONs. These components encompass:

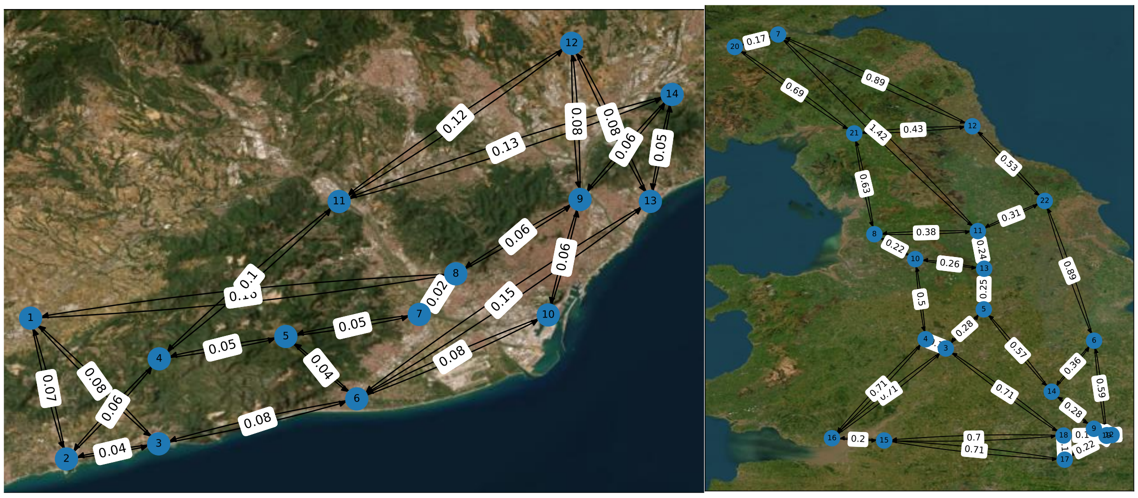
- *Source and Destination Nodes*: Initial and terminal points of the requested optical connection.
- *Bandwidth Requirement:* The amount of bandwidth that the service connection requires, represented by the number of FSs needed to fulfill the demand of the connection.
- *Latency Requirement:* The maximum tolerated end-to-end latency for the service connection, indicating the maximum delay that can be accepted for the connection to be considered successful.
- *Current Spectrum Utilization:* This includes detailed information on the optical spectrum available for every candidate path, particularly focusing on:
    - The position of the first available FS block, providing an indication of where in the spectrum the first unoccupied slot is located.
    - The average size of available FS blocks, which offers insights into the fragmentation and distribution of free spectrum slots.
    - The total number of available FSs, giving a clear picture of how much of the spectrum is currently unoccupied and potentially usable for new connections.
    - Specific characteristics for every candidate path, including end-to-end delay and spectrum utilization, to facilitate optimal path selection by the DRL agent.

The DRL agent operates within a pre-defined *action* space, choosing one of several candidate paths as the most suitable option for each request. It uses a *reward* function that considers both the bandwidth and latency requirements of the connectivity service. The function awards actions that allocate higher bandwidth and prioritize lower-latency connections, promoting an optimal balance that enhances overall network performance. Specifically, the reward is a sum of a component proportional to bandwidth demand and an inverse component of latency, encouraging efficient and latency-aware resource allocation. The reward function is defined as follows:

$$R = \begin{cases} b + \frac{1}{l}, if\ sucessful\ allocation \\ -10, \qquad\qquad\qquad otherwise \end{cases} \tag{4}$$

3.3.3 Simulations and Results

To evaluate the performance of the DRL approach for latency-aware RSA, two EON topologies were used: a metropolitan network in Barcelona City Network (BCNNet) and a national core network across Great Britain (BTNet). BCNNet includes 14 nodes and 42 bidirectional links, while BTNet encompasses 22 nodes and 70 links, as shown in Figure 13. Both networks use identical frequency grids supporting 100 Frequency Slots (FSs) on optical fiber links. Optical connection requests are simulated between random pairs of nodes following a Poisson process with an inter-arrival time (λ) fixed at 2 seconds, and service duration time follows an exponential distribution with an average (μ), which varied to simulate different traffic loads. Bandwidth demands are distributed among two, four, and eight FSs, with each FS requiring 12.5 GHz, assuming a spectral efficiency of 1 b/s/Hz, which translates to a transmission rate of 12.5 Gb/s per FS. The required latency for connection requests was uniformly generated, ranging from the average shortest path delay to twice that value across all node pairs. For BCNNet, latency varied between 0.30 ms to 0.60 ms, while for BTNet, it ranged from 1.25 ms to 2.5 ms. The DRL agent employed DNNs with five fully connected hidden layers for its decision-making process. Different DNN structures were used for the two topologies, conditioned by their number of nodes. The policy network's output layer featured four neurons, indicating the probability of choosing one of four candidate paths. The Adam optimizer was used for training with a 10^{-4} learning rate, and a discount factor was set to 0.95. Additionally, k-shortest paths first fit (kSPFF) computed the four candidate shortest end-to-end delay paths for path selection.



(a) (b)

Figure 13. Topologies used for the experimental evaluation

The performance of the proposed DRL-based RSA solution was compared to the traditional kSPFF heuristic algorithm in terms of Bandwidth Blocking Ratio (BBR). The BBR is defined as the ratio of the amount of bandwidth that could not be allocated due to blocking (unsuccessful provisioning of connectivity services) to the total bandwidth requested by all services. A lower BBR indicates better network performance, as it means a higher percentage of the requested services were successfully provided.

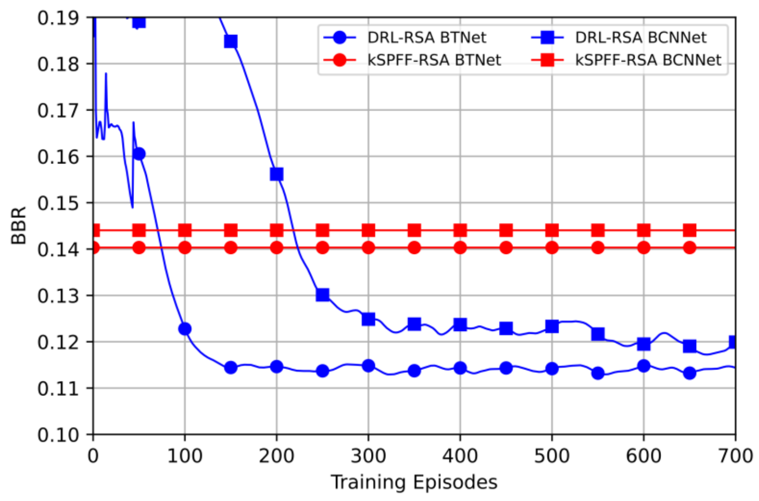


Figure 14. Training phase of DRL RSA

To train the DRL agent, the traffic load was set to 200 Erlangs. Figure 14 shows that, as the number of training episodes for the DRL model increased, a noticeable improvement in BBR was observed compared to the kSPFF algorithm. This indicates the DRL model's capability to learn and optimize its decisions over time, progressively enhancing its decision-making policy to better accommodate diverse network conditions and connectivity demands.

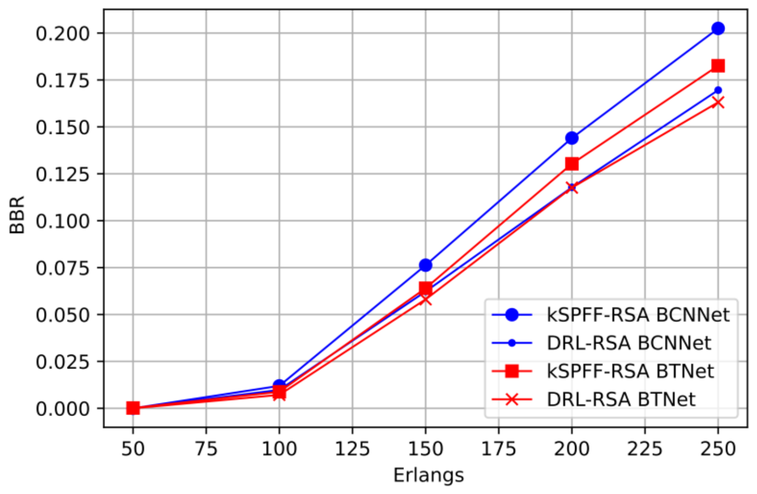


Figure 15. Performance benchmarking of kSPFF and DRL RSA

The performance of the trained DRL model was also assessed against the kSPFF algorithm over varying traffic loads, ranging from 50 to 250 Erlangs (Figure 15). For lower traffic loads (e.g., 50 Erlangs), no significant difference in BBR was noted between the two approaches. However, as traffic loads increased, the DRL model showed a marked improvement in BBR. At a traffic load of 150 Erlangs, the DRL approach achieved a BBR reduction of 16.20% relative to kSPFF.

Moreover, at a high traffic load of 250 Erlangs, the DRL model outperformed the kSPFF algorithm by reducing the BBR by around 18.35%.

These results underline the effectiveness and potential of using a DRL-based approach for RSA in EONs. Specifically, it showcases the DRL agent's key role in enhancing EON service provisioning through ML-driven decisions, aiming to improve resource utilization, satisfy stringent latency requirements, and adapt to the dynamic network scenarios.

3.4 Function Placement

The integration of computing and networking resources is pivotal for the efficient deployment of network services that meet the diversified requirements. This challenge becomes exceedingly complex when considering the orchestration across distributed cloud infrastructure interconnected through EONs, especially in the context of VNF placement and the establishment of lightpaths. Various approaches to VNF placement have been proposed, including optimization problem solving, heuristics, and ML techniques [Vil20, Che21]. DRL is a promising ML approach due to its ability to handle high-dimensional network states and actions, and to consider long-term performance through a discount factor.

This section showcases the employment of DRL to tackle the intricacies of VNF placement and service provisioning within such advanced network infrastructures. Herein, realistic network services are represented as VNF Forwarding Graphs (VNF-FGs) with meshed VNF interconnections and arrive and depart dynamically. DRL-based approach evidences a significant outperformance over traditional algorithms in terms of service blocking rates [Her23b, Her23c].

3.4.1 Proposed Framework

In this scenario, VNFs need to be placed across different points of presence (PoPs) in a substrate network. Each VNF has computational resource requirements, and the virtual links between VNFs have bandwidth and latency constraints. The process involves mapping VNFs to PoPs and establishing lightpaths (optical connections) between the selected PoPs to support the virtual link requirements.

The crux of the proposed approach lies in utilizing two cooperating DRL agents. The first agent (Path DRL Agent) is tasked with computing the optimal lightpaths between PoPs, thereby managing the networking requirements such as spectrum (i.e., continuity and contiguity) and latency constraints native to EONs. The interaction between these agents is critical, as the output from the lightpath computation directly influences the decision-making process of the second agent (Compute DRL Agent), which focuses on the strategic placement of VNFs across available PoPs, based on computing resource availability and networking criteria. The Compute DRL agent receives a reward of 0 for successful VNF allocation, a positive reward for complete VNF-FG deployment based on allocated computing capacity, and a negative reward for failed VNF deployment. Figure 16 depicts the design of the devised DRL-based solution. The solutions provide a holistic framework for service provisioning that addresses both, computation and optical spectrum allocation challenges in the substrate infrastructure. This dual-focused approach ensures that all aspects of service delivery are optimized, leading to more reliable and efficient network service deployment.

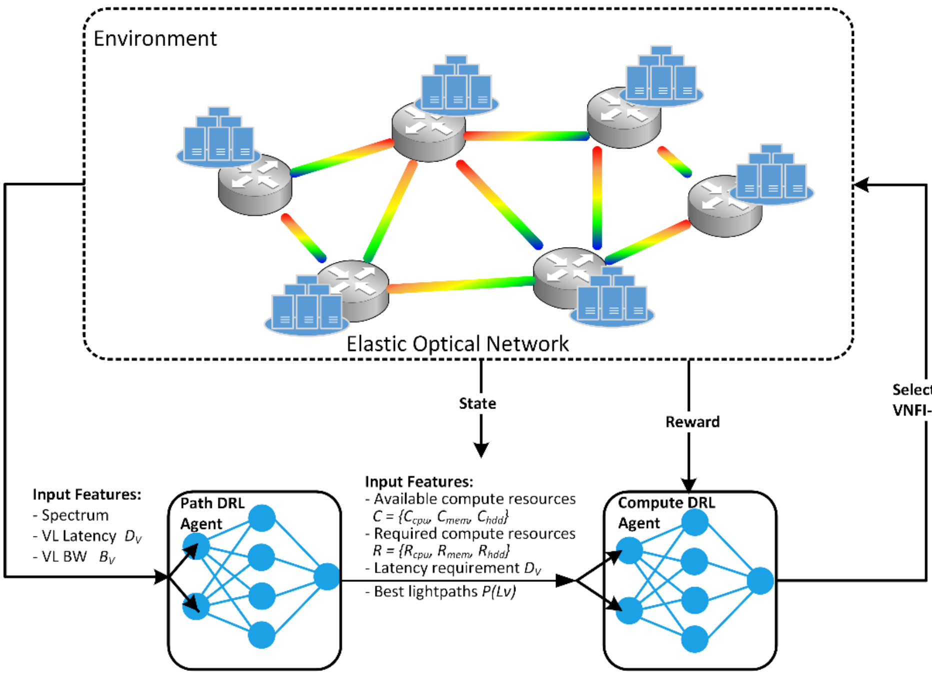

Environment
Elastic Optical Network
State
Reward
Selected VNFI-PoP
Input Features:
- Spectrum
- VL Latency D_V
- VL BW B_V
Path DRL Agent
Input Features:
- Available compute resources $C = \{C_{cpu}, C_{mem}, C_{hdd}\}$
- Required compute resources $R = \{R_{cpu}, R_{mem}, R_{hdd}\}$
- Latency requirement D_V
- Best lightpaths $P(Lv)$
Compute DRL Agent

Figure 16. DRL-based solution for VNF-FG Placement

For each service request, Compute-DRL iterates over each VNF, trying to place it on a PoP that meets its resource requirements and the virtual link constraints. Compute-DRL takes as input the VNF resource demands, virtual link constraints, available PoP resources, and latencies of the computed lightpaths between PoP pairs. While the action space specifies the selected PoPs for hosting the VNFs.

3.4.2 Simulations Scenario

The evaluation of the proposed DRL approach was conducted on the NSFNet network topology with 14 optical nodes and 42 fiber links, shown in Figure 17(a). Each node has a PoP with 100 CPU cores, and each optical link could accommodate 100 frequency slots. Three network service types with varying numbers of virtual network functions (VNFs), virtual links, bandwidth requirements, and maximum delay were defined as it is shown in Table 7.

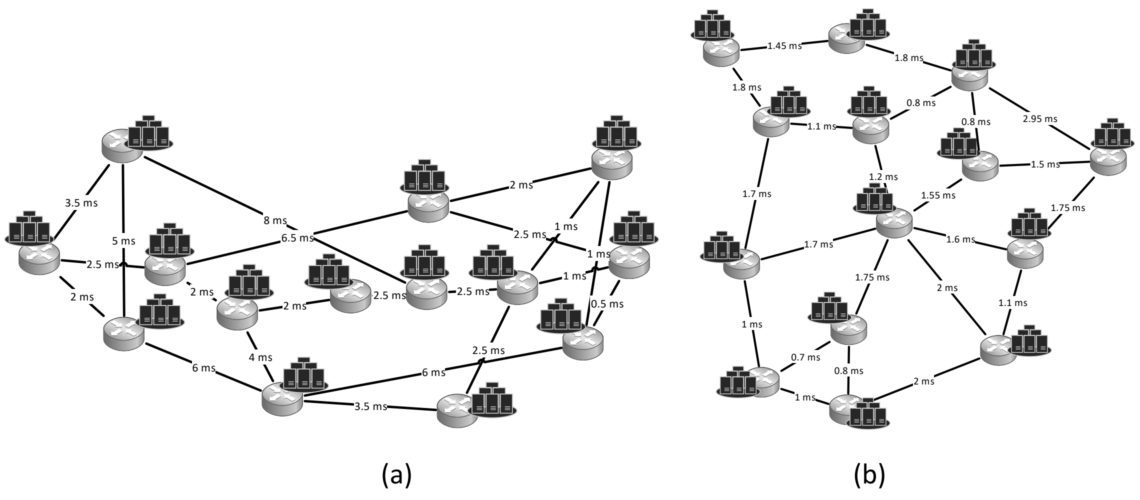


Figure 17. (a) NSFNet topology and (b) Unseen network topology

Table 7 Network Service Types

(NSD type)	# of VNFs	# of VLs	Bandwidth (FSs)	Latency (ms)
Industry 4.0	2	2	2	5
Automotive	4	10	4	10
Media	6	20	8	15

3.4.3 Performance Evaluation

For the evaluation, Compute-DRL agent was implemented based on the Proximal Policy Optimization (PPO) algorithm. Furthermore, some key innovations were incorporated in the solution. A notable advancement in the DRL algorithm's efficiency is the incorporation of invalid action masking. This technique significantly narrows down the action space for the agents by excluding infeasible actions, thereby reducing the training time and enhancing decision-making accuracy. It allows the DRL agent to converge faster during training. Results show that DRL agent with invalid action masking (Maskable PPO) converges 22% faster (i.e., from 3.6 to 2.8 million episodes) and achieves higher average rewards during training than a DRL agent without it, as it is shown in Figure 18(a).

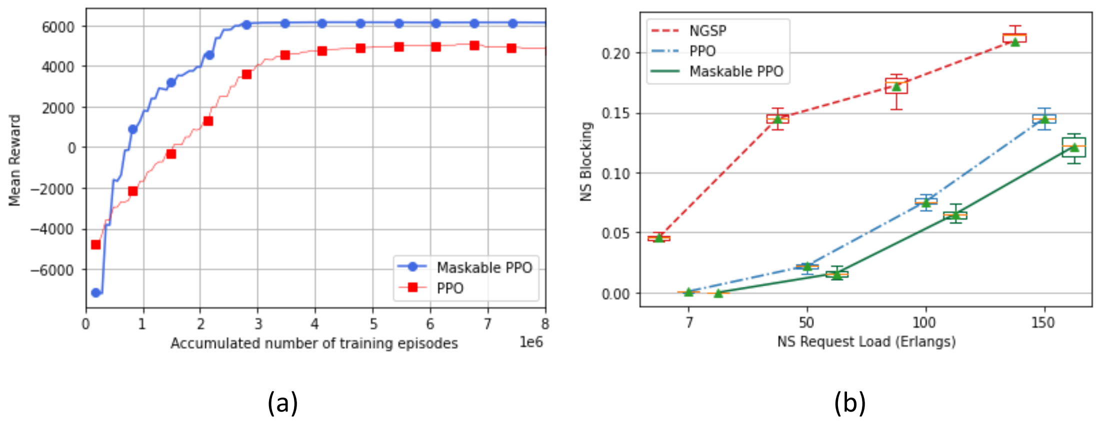
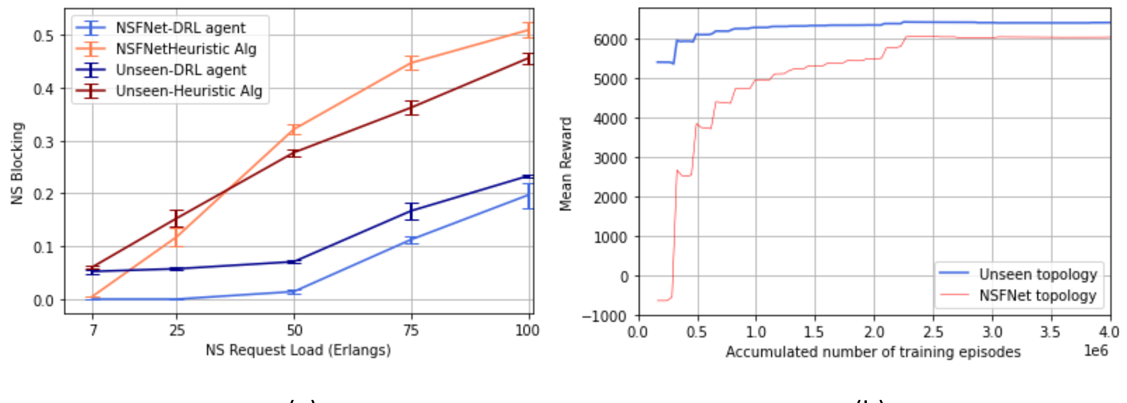


Figure 18. Convergence of DRL-based solution, and Network service blocking rate

The experimental results also show that the DRL approach outperforms the baseline Non-Recursive Greedy SFC Placement (NGSP) algorithm by reducing the number of blocked services. NGSP selects the PoP with higher residual computing resources and computes lightpaths using the k-shortest path algorithm. This is achieved through the cooperation between the two DRL agents, where each agent focuses on solving a specific problem (spectrum assignment or compute allocation), and the output of one agent is used as input for training the other. As can be seen in Figure 18(b), under 50 Erlangs of request load, the blocking rate obtained from the agent trained with Maskable PPO achieves the best performance, reducing blocking by 12.8% compared to that of the NGSP algorithm. Maskable PPO slightly reduces the blocking rate by 0.6% compared to the agent trained with PPO.

Once the DRL training is complete, the trained model was tested on an unseen network topology (Figure 17(b)) with 14 nodes and 22 bidirectional links to assess its generalization capability. Figure 19(a) exhibits that the DRL-based solution significantly outperformed a heuristic benchmark algorithm in both the original NSFNet topology and the unseen topology regarding the service blocking rate. This demonstrates not only the efficacy of the DRL solution in optimizing VNF placements but also its strong adaptability to different network topologies without prior exposure. To evaluate the capability of the DRL solution to adapt and improve over time by learning from its interactions with a dynamically changing environment, the trained model was used to learn in the unseen network topology. By using a pre-trained model, training convergence is accelerated, as shown in Figure 19(b).



(a) (b)

Figure 19. (a) Network service blocking rate and training convergence in unseen topology

In conclusion, the integration of sophisticated DRL techniques paves the way for more autonomous, efficient, and flexible network service provisioning in multi-layered network infrastructures like those involving EONs and distributed cloud systems. Leveraging the synergy between DRL agents helps to manage the complex demands of modern network services and scale solutions to encompass changing network topologies, and their applicability in real scenarios.

4 Failure Management

The performance of optical devices can degrade because of aging and external causes like, for example, temperature variations. Such degradation might start with a low impact on the QoT of the supported lightpaths (soft-failure). However, it can degenerate into a hard-failure if the device itself is not repaired or replaced, or if an external cause responsible for the degradation is not properly addressed.

The QoT is related to the linear and nonlinear (NL) optical noise, and it can be estimated based on a model describing the physics of propagation, e.g., the generalized Gaussian Noise (GN) model. Additionally, effects such as aging of optical devices might severely affect the QoT. Aging effects are usually considered by means of costly system margins. Examples include: *i*) the degradation of Optical Amplifiers (OA), which can be quantified as increased Noise Figure (NF); and *ii*) detuning of the lasers in the TRXs or frequency drift of the filters in Wavelength Selective Switches (WSS), which can lead to misalignments. If those degradations (*soft-failures*) are not properly handled (e.g., by retuning, repairing, or replacing the related optical device), they can degenerate into hard-failures when the SNR reduces, and zero post-FEC error cannot be achieved (*FEC limit*); this could affect a large portion of network services. Therefore, it is of paramount importance not only to detect any QoT degradation, but also to identify the cause and localize the device causing the degradation.

Further, a considerable effort is being paid towards disaggregating the optical layer to enrich the offer of available solutions and to enable the deployment of solutions that better fit optical network operators' needs. Such disaggregation, however, tends to make network surveillance and maintenance more complex in general, due to the absence of vendors providing support of vertically integrated network equipment.

To support failure management, the control plane needs to be enriched with Monitoring and Data Analytics (MDA) capabilities. Once monitoring data, notifications, and alarms have been collected from the data plane, data analytics algorithms (e.g., based on ML techniques) can analyze them to proactively detect the degradation, identify and localize the cause, and anticipate hard-failures before they occur. Once detected, identified, and localized, recommendations can be issued to the network controller so it can decide about rerouting and/or reconfiguring the network, as well as notifying the management plane for maintenance. Note that ML-based approaches require training and validation datasets, which makes their practical application difficult due to key drawbacks, namely: *i*) limited data availability; *ii*) long duration of the training and validation phases until obtaining robust and reliable ML models (this could be accelerated by using simulation tools in *sandbox domains*); *iii*) poor adaptability in the event of physical layer changes; and *iv*) reduced exportability to other scenarios/ conditions different than those used for training.

The topic of failure management in optical networks (including anticipated detection, identification, and localization) has been extensively explored. Previous works centered on analyzing the QoT represented by the measured BER, spectrum, etc. for detecting failures, or correlating alarms for localization. In contrast, the status of optical devices can be through measures related to device parameters like optical power, gain, temperature, etc. to proactively detect and localize potential faults and determining the likely root-cause. This approach to failure detection, derived from the analysis of devices' parameters, is key to really identify and localize the failure itself.

However, it is not always possible to obtain the right value of those devices' parameters that can be related to the QoT, in particular in disaggregated scenarios. Note that any QoT model uses a set of input parameters to describe the specific characteristics of the different optical devices that participate in the optical layer, like WSSs as building blocks of Reconfigurable Optical Add / Drop Multiplexers (ROADM), TRXs, and In-Line OAs, e.g., Erbium Doped Fiber

Amplifier (EDFA). In such cases, ML methods can be used for finding the right value of such QoT model's parameters aiming at improving the QoT estimation. Our approach applies reverse engineering from the real QoT values—collected periodically from the TRXs—to derive the evolution of the value of QoT model's parameters (referred to as *modeling parameters*) that explain such QoT observations. We believe that, by analyzing such evolution, it is possible to anticipate more precisely future degradations, and enable failure localization.

4.1 Failure detection, localization and identification

This section details our proposed MESARTHIM methodology that targets at: *i*) detecting and localizing the optical device responsible for the soft-failure; *ii*) identifying the modeling parameters that explain the observed effects in the QoT; and *iii*) estimating the evolution of the value of such parameters to find whether the soft-failure will degenerate into a hard-failure. This advanced network performance analysis procedure facilitates diagnosis and network maintenance. Furthermore, since the relation between monitored SNR and modeling parameters is not linear, the analysis carried out in the later space (i.e., modeling parameters) can accelerate soft-failure detection, identification, and localization. Specifically, the contributions are:

1. The MESARTHIM methodology for soft-failure detection, identification, and localization, modeling parameter estimation and severity estimation.
2. *Network surveillance* and *soft-failure localization* based on device modeling parameter estimation, which is a key part of the MESARTHIM methodology.
3. Procedures for *identification* and *severity estimation*, once a soft-failure has been detected and localized.

The discussion is supported by the experiments and numerical results.

4.1.1 The MESARTHIM Methodology

Among the effects degrading the QoT within optical systems, we consider degradations arising from ROADMs and In-Line OAs, where a ROADM consists of WSSs and OAs. Both building blocks face aging and non-ideal conditions. For example, although OAs are considered robust devices, they also suffer time-varying effects like Noise Figure (NF) which might increase over time due to the aging of the amplifier building blocks. The NF is also frequency-dependent and, as the allocation of the spectrum might become time-dependent. Therefore, the NF can be modeled as a time-frequency variation. The pump lasers of the EDFAs also subject to degradation, which can be adjusted thanks to internal control loops, but which still reduces the EDFA efficiency. For what concerns the WSSs, they might suffer temperature-dependent variations, which might lead to frequency shift over time; furthermore, individual channels can drift as well, and both effects can be highly detrimental in terms of QoT. We consider gradual time-varying device degradations on OA and add/drop (A/D) WSSs in the ROADMs. Specifically, we consider that soft-failures can be explained by one of the following events in the modeling parameters: *a*) NF increase; *b*) maximum optical output power (P-max) decrease; and *c*) Optical SNR (OSNR) degradation caused by frequency drifts of the WSSs due to temperature fluctuation.

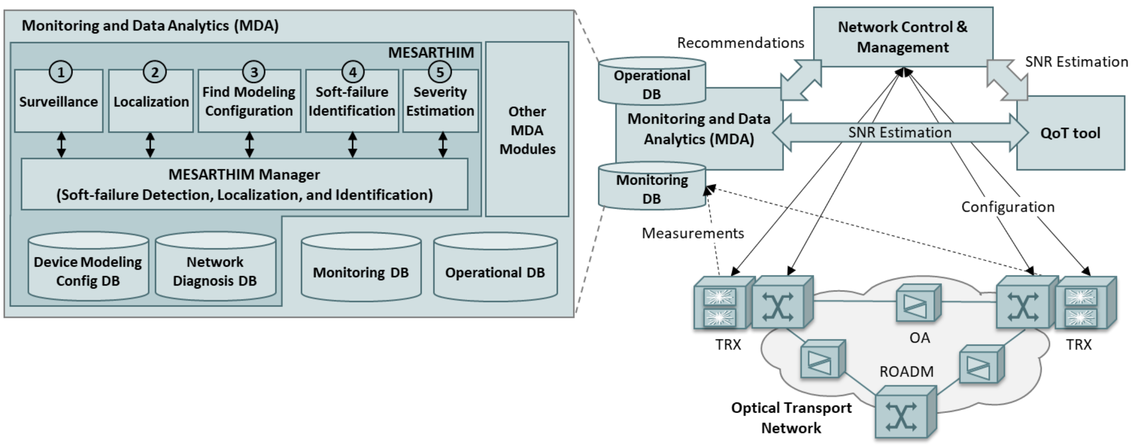

Monitoring and Data Analytics (MDA)
MESARTHIM
1 Surveillance
2 Localization
3 Find Modeling Configuration
4 Soft-failure Identification
5 Severity Estimation
Other MDA Modules
MESARTHIM Manager (Soft-failure Detection, Localization, and Identification)
Device Modeling Config DB
Network Diagnosis DB
Monitoring DB
Operational DB
Network Control & Management
Recommendations
SNR Estimation
Operational DB
Monitoring and Data Analytics (MDA)
QoT tool
Monitoring DB
Configuration
Measurements
TRX
OA
ROADM
Optical Transport Network

Figure 20. Overview of the proposed failure analytics architecture and the MESARTHIM methodology

Our proposed architecture for soft-failure analysis is illustrated in Figure 20. The optical layer consists of a disaggregated set of ROADMs and TRXs, and a set of optical links with a number of In-Line OAs interconnecting ROADMs. The control plane includes: *i*) a Network Controller to program the network devices, which is coordinated by the Network Management plane, and includes network maintenance; *ii*) an MDA system that collates measurements from the data plane, analyses the data and issues recommendations to the network controller, as well as notifications regarding failures; and *iii*) a QoT tool based on GNPy that estimates the SNR of the lightpaths and it is used for connection provisioning and for failure analytics.

The MDA system stores a replica of the operational databases (DB) that are synchronized from the network controller. In addition, it collects measurements from the optical devices with a given periodicity and stores them in a *Monitoring* DB; we assume that the MDA collects SNR samples from the TRXs every 15 minutes. These measurements are used by MESARTHIM to: *i*) estimate those modeling parameters related to optical devices (resources); *ii*) analyze the evolution of the measured SNR and that of the modeling parameters to detect any degradation as soon as it appears; and *iii*) determine the severity of the degradation based on the foreseen impact on the performance of the lightpaths.

Figure 20 also sketches the MESARTHIM methodology implemented in the MDA system. Specifically, the following building blocks can be identified: (i) the *Surveillance* block that analyzes the SNR measurements and the value of modeling parameters to detect any meaningful degradation (e.g., by threshold crossing); (ii) the *Localization* block that localizes the soft-failure; (iii) the *Find Modeling Configuration* block that finds the most likely value of the modeling parameters of a given resource, so it results into SNR values of the lightpaths being supported by such resources similar to those that have been actually measured; (iv) the *soft-failure Identification* block that, assuming a resource has been localized as the source of the soft-failure, finds what is the modeling parameter responsible for such failure; and (v) the *Severity Estimation* block that estimates whether and when the soft-failure will degenerate into a hard-failure. In addition, two internal repositories are used: *a*) the *Device Modeling Config* DB with the evolution of the value of modeling parameters along time for every resource; and *b*) the *Network Diagnosis* DB that stores historical data for analysis purposes. The MESARTHIM manager coordinates those blocks to achieve intelligent QoT analysis, as well as manages the interface with the QoT tool. The main procedures for the different blocks of MESARTHIM are detailed next.

4.1.2 Surveillance and Localization

In this section, we describe two different approaches for the surveillance block, named *SNR-wise* that analyzes the evolution of the SNR, and *Modeling-wise* that analyzes the evolution of the value of modeling parameters. Resources affected by a soft-failure procedure are localized. Additionally, the main procedure for the *Find Modeling Configuration* block is presented. Table 8 introduces the used notation. We assume that surveillance is carried out periodically, e.g., after at least one new SNR measurement has been collected for every lightpath in the network. Both algorithms return the resources with the found likely modeling configuration, each with a subset of lightpaths, indicating that some soft-failure has been detected.

Table 8. Notation for Failure management

G	Graph representing the network topology.
P	Set of all lightpaths
P'	Subset of lightpaths (*P'*⊆*P*)
R	Set of optical devices, index *r*.
C	Set of clusters of lightpaths with found same behavior ({<*behavior*, *P'*>})
SR	Set of suspicious resources. Each element identifies the resource *r* and the lightpaths that it supports (*SR* = {<*r*, *P'*>}).
SF	Set of Soft-Failures. (*SF* = {<*r*, *P'*>})

SNR-wise Surveillance

This approach focuses on the analysis of SNR measurements and compares them to the SNR estimated by the QoT tool for every lightpath, to detect any meaningful deviation (exceeding a differential threshold). The lightpaths that exceed the differential threshold are considered degraded and are further analyzed in terms of the *behavior* of the measured SNR evolution to find a correlation among them; behavior is the result of stationarity analysis of the SNR evolution. Non-stationary patterns (e.g., trend, periodicity), if found, are quantified (e.g., a period interval in case of seasonality) to compose the behavior. For illustrative purposes, Figure 21 shows three examples of behavior: a) *stationary* (typical for lightpaths that do not exceed the differential SNR threshold); b) *gradual* decay; and c) *cyclic*. In the case of finding groups of lightpaths with similar behavior, common underlying resources are analyzed to localize the responsible one for such degradation. It is worth highlighting that grouping lightpaths with similar behavior enables localizing multiple soft-failures. In such a case, the likely configuration parameters are estimated using the *Find Modeling Configuration* block with all lightpaths supported by that resource.

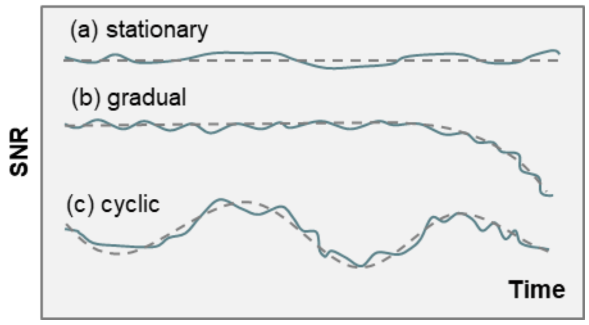


Figure 21. Three examples of behavior.

Algorithm 1 describes the procedure used to find the behavior of the evolution in time of a time series data. The algorithm receives an object *x*, which includes: *i*) the time series to be analyzed (e.g., the SNR of a path *p*); *ii*) a window size *w* used for smoothing purposes; and *iii*) a threshold *thr* used to detect a significant non-stationary behavior. The data series (*Y*) is smoothed ($\tilde{Y}$) by computing a moving average in non-overlapped windows of size *w* (lines 1-2 in Algorithm 1). Then, the difference between maximum and minimum in $\tilde{Y}$ is computed and compared with the threshold; if it is lower than the threshold, no behavior is returned (line 3). Otherwise, the algorithm carries out an analysis to characterize the type of behavior by clearly distinguishing between gradual degradation (*gradual*) and cyclic fluctuation (*cyclic*), as well as any other undefined evolution (e.g., random peaks). Specifically, if $\tilde{Y}$ presents an incremental or decremental monotonic evolution, the fittest trendline (in terms of Pearson correlation coefficient) among linear, polynomial, exponential, and logarithmic trends is computed and returned as a parameter of the identified gradual degradation (lines 4-6). Otherwise, the periodicity of $\tilde{Y}$ is computed based on the results of automated periodogram power spectral density analysis. In case that a significant period is found, cyclic fluctuation with that period is returned (lines 8-9), whereas other behavior is returned if neither gradual nor cyclic behavior is found (line 10).

Algorithm 1. Find Behavior Procedure

```
INPUT: x                    OUTPUT: hasBehavior
 1: Y ← x.<time series>      # snr OR evol
 2: Ỹ ← nonOverlappingMovingAverage(Y, x.w)
 3: if max(Ỹ) - min(Ỹ) < x.thr then return False
 4: if isMonotonic(Ỹ) then
 5:     trendline ← computeFittestTrendline(Ỹ)
 6:     x.behaviour ← <"gradual", trendline>
 7: else
 8:     period ← findPeriodicity(Ỹ)
 9:     if period then x.behaviour ← <"cyclic", period>
10:     else x.behaviour ← <"other", ∅>
11: return True
```

Algorithm 2. SNR-wise Surveillance Algorithm

```
INPUT: G, P, T, current_t          OUTPUT: SR
 1: P' ← ∅
 2: for p in P do
 3:     if getMonitoringData(p, current_t) <
                              SNR_threshold(p) then
 4:         P' ← P' ∪ {p}
 5: if P' = ∅ then return ∅
 6: C = {<behavior, P'>} ← ∅
 7: for each p in P' do
 8:     p.snr ← getMonitoringData(p, T)
 9:     hasBehavior ← findBehavior(p)
10:     if hasBehavior then addByBehaviorSimilarity(C, p)
11: if C = ∅ then return ∅ else SR ← ∅
12: for each c in C do
13:     R = {<resource, P'>} ←
                  FindCommonResources(G, c.P')
14:     SR ← SR ∪ R
15:     for each r in R do
16:       r.evol ← ∅
17:       for each t in T do findLikelyModelingConfig (r, t)
18: return SR
```

Algorithm 2 details the pseudocode of the *SNR-wise Surveillance* algorithm; it receives as input the network graph *G*, the list *P* of lightpaths currently established in the network, the number *T* of historical monitoring samples to be considered, and the current time (*current_t*). The algorithm first retrieves and examines the last monitoring data available for every lightpath looking for those with degraded SNR (lines 1-5 in Algorithm 2). In case some SNR degradation is found, *SNR-wise Surveillance* proceeds with an in-depth SNR analysis carried out in two steps. During the first step, the set of clusters *C*, capturing the behavior observed in the lightpaths, is found (lines 6-10); for this analysis, the last *T* monitoring samples are considered. Note that by considering the evolution of lightpaths' SNR, spurious measurements in one lightpath can be detected and ignored. In the case that, at least, one set of lightpaths presents a similar behavior, e.g., decay or periodicity (as in Figure 21), the algorithm continues with the second step. The common resources supporting the lightpaths in each cluster *c* are computed and added to the set *SR* of resources that are suspicious of being affected by a soft-failure (lines 12-14). Moreover, for each common resource, a likely evolution of the input parameters is found (lines 15-17); the estimated configuration is stored in the *Device Modeling Config* DB for further analysis. The resources with the found likely configuration, each with a subset of lightpaths, are eventually returned (line 18).

Modeling-wise Surveillance

This surveillance approach analyses the evolution of the value of modeling parameters of the resources. In this case, the SNR measurements of *all* the lightpaths in the network supported by a resource are used to estimate the most likely modeling configuration of such resource using the *Find Modeling Config* block. The found modeling configuration is stored and its evolution is analyzed to detect any meaningful degradation, e.g., a significant trend and/or variation.

Algorithm 3. Modeling-wise Surveillance Algorithm

```
INPUT: G, P, T, current_t        OUTPUT: SR
 1:  SR ← ∅
 2:  R = {<resource, P'>} ←
                   GroupPathsByResources(G, P)
 3:  for each r in R do
 4:     for each p in r.P do
 5:        p.snr ← getMonitoringData(p, current_t)
 6:     findLikelyModelingConfig(r, current_t)
 7:     r.evol ← configDB.SELECT(r, T)
 8:     hasBehavior ← findBehavior(r)
 9:     if hasBehavior then SR ← SR ∪ {r}
10:  return SR
```

Algorithm 3 details the pseudocode of the *Modeling-wise Surveillance* algorithm; it first initializes the *SR* data structure (line 1 in Algorithm 3) and creates the set of resources with the lightpaths that each one supports (line 2). Next, for every single resource, the algorithm uses the last SNR measurements to find the current value of the parameters modeling the resource, which are stored in the *Device Modeling Config* DB by the *Find Modeling Configuration* block (lines 3-6). The behavior of the modeling parameters evolution is analyzed (by calling Algorithm 1) and, if a non-stationary pattern is found, the resource is added to the *SR* set (lines 7-9). It is worth noting that this analysis could detect soft-failures that have not yet had a relevant impact on the lightpaths (i.e., the SNR degradation threshold has not been exceeded yet), as parameters and SNR are not linearly related.

Soft-Failure Localization

In case that the surveillance phase has identified a set of suspicious resources, Algorithm 4 localizes the soft-failures. The algorithm first removes the suspicious resources that explain the very same set of lightpaths (lines 1-6 in Algorithm 4), as localization is not yet possible in those cases. Next, the resources that explain the SNR of complete subsets of lightpaths are localized and classified as *independent* soft-failures, *iSF* (lines 6-10). The rest of the suspicious resources are related to soft-failures that affect common subsets of lightpaths, so a given lightpath can be affected by more than one soft-failure.

Algorithm 4. Soft-Failure Localization Algorithm

```
INPUT: SR                         OUTPUT: iSF, mSF
 1: for each r in SR do
 2:     Rr ← ∅
 3:     for each r′ in SR | r ≠ r′ do
 4:         if r.P = r′.P then Rr ← Rr ∪ {r, r′}
 5:     SR ← SR - Rr
 6: iSF ← ∅
 7: for each r in SR do
 8:     if r.P ∩ r′.P = ∅ ∀r′ SR | r ≠ r′ then
 9:         iSF ← iSF ∪ {r}
10:         SR ← SR - {r}
11: return iSF, SR
```

Finding the Most Likely Modeling Configuration

The above surveillance approaches use the *FindModeling Configuration* block, which estimates the most likely modeling configuration of a given resource *r*. Given the ranges of feasible configuration values $\mathbb{V}$ of *r*, the configuration estimation problem consists in finding the most likely values *v*, by minimizing the error (computed as mean squared error -*MSE*) between the measured (S) and the estimated ($\hat{S}$) SNR for the set of lightpaths being supported by *r*, *P*(*r*), i.e.:

$$\min_{v \in \mathbb{V}} MSE\left(S(P(r)), \hat{S}(P(r)|v)\right) \qquad (5)$$

Algorithm 5. Modeling Config Search

```
INPUT: P, r, $v_{min}$, $v_{max}$          OUTPUT: <v, m>
 1: <v, m> ← configDB.getMin(); numIters ← 0
 2: while m > epsilon AND numIters++ < MaxIters do
 3:     <$v_l$, $m_l$>, <$v_r$, $m_r$> ← configDB.getNeigbours(v)
 4:     $v_a$ ← intersect(v, m, $v_l$, $m_l$, $v_{min}$, $v_{max}$)
 5:     $v_b$ ← intersect(v, m, $v_r$, $m_r$, $v_{min}$, $v_{max}$)
 6:     if $v_a = v_l$ then $v_a \leftarrow (v + v_l) / 2$
 7:     if $v_b = v_r$ then $v_b \leftarrow (v + v_r) / 2$
 8:     $m_a$ ← configDB.getConfig($v_a$)
 9:     if $m_a$ = None then
10:         $m_a$ ← MSE(P.SNR, QTool(P, r, $v_a$))
11:         configDB.addConfig(<$v_a$, $m_a$>)
12:     $m_b$ ← configDB.getConfig($v_b$)
13:     if $m_b$ = None then
14:         $m_b$ ← MSE(P.SNR, QTool(P, r, $v_b$))
15:         configDB.addConfig(<$v_b$, $m_b$>)
16:     <v, m> ← configDB.getMin()
17: return <v, m>
```

The SNR estimation $\hat{S}(\cdot)$ is obtained by calling the QoT tool and thus, the optimization problem in eq. (5) cannot be solved by traditional methods like *Steepest Descent*, which are based on computing the gradient of the function to be minimized. In view of that, the *findLikelyModelingConfig*() procedure uses the *modelingConfigSearch*() one (Algorithm 5) to solve the optimization problem in eq. (5). The algorithm interrogates the QoT tool with different values of the parameters and the configuration entailing the lowest error with respect to the SNR values measured is returned. The procedure assumes that: *i*) the function is convex in $\mathbb{V}$, i.e., there is just one minimum that is the global minimum (tests supporting this assumption will be carried out); and *ii*) there is just one of the modeling parameters of *r* with a value different than the initially found. The procedure fixes the value(s) of the parameter(s) and requests the QoT tool to compute, with such configuration, the SNR values of the lightpaths being supported by *r*. By computing the *mse*(·) function, the procedure determines if such configuration is likely enough or if more queries to the QoT tool are needed.

As calls to the QoT tool are time consuming, *modelingConfigSearch*() targets at minimizing them; instead of using the brute force and request the estimation of the SNR for the whole range of possible values, the procedure uses the projection of two points to determine the next value of the parameters to be used for the estimation. As parameters typically evolve smoothly in time when devices are affected by a soft-failure, it also stores the last configuration(s) in the *Device Modeling Config DB*; every time it is called, it uses such configurations as starting points for the search aiming at finding the optimal solution fast.

4.1.3 Soft-Failure Identification and Severity Estimation

Let us now focus on the *Soft-Failure Identification* and the *Severity Estimation* blocks, which are assumed to be executed as soon as degradation is detected and localized (via SNR and/or device configuration analysis). These blocks make MESARTHIM able to generate notifications to the network controller containing not only the list of degraded lightpaths, but also their expected evolution and the estimated time for the soft-failure to degrade into a hard-failure.

Let us imagine that the surveillance and localization analysis presented above detected some degradation at time *t* and therefore, non-empty *ISR* and/or *MSF* sets were found. In addition, let us define *state* as the combination of the estimated modeling configuration of the resources that are involved, and the SNR experienced by the supported lightpaths. With the aim of an accurate diagnosis, it is interesting not only to analyze the current state but also to predict its future evolution. It is for this very reason that historical data (within a pre-defined time window) are gathered from monitoring and device config DBs and stored in the *Network Diagnosis DB*. The evolution of each parameter (lightpaths' SNR and device modeling config parameters) is analyzed individually using time series forecasting techniques. By using different techniques with different parametrization, several expected evolutions can be obtained as a practical way to generate different likely projections for parameter degradation.

For illustrative purposes, let us analyze an example of the evolution of three parameters selected from a hypothetical state: the SNR of a given lightpath affected by an SNR degradation, as well as the NF and P-max parameters modeling an OA traversed by that lightpath. Figure 22 presents three time-evolutions of the OA modeling parameters and SNR, and their analysis at current time *t*. At this time, the failure has been correctly localized in the selected OA; however, the cause of the degradation (either NF or P-max) is still unclear. This is the reason behind performing the estimation and evolution analysis for all modeling parameters; such analysis is carried out following forecasting techniques. Three curves representing the *upper* (f_u), *centered* (f_c), and *lower* (f_l) projected possible evolutions are depicted for each parameter. Based on such projections, monitored data are evaluated and an *indicator* (φ) is computed and used to discard modeling parameters while identifying the one that is the most likely to be the real cause of the observed performance degradation. Hence, the indicator provides large values when the modeling parameter is far from the expected behavior of an affected parameter. The obtained indicator is accumulated with the values obtained in the previous computations, and identification is positive when the parameter with the lowest accumulated indicator is far from all the rest of the parameters with a significantly higher accumulated one. This evidence can be easily observed by setting up a threshold allowing the discrimination of low and high accumulated indicator modeling parameters.

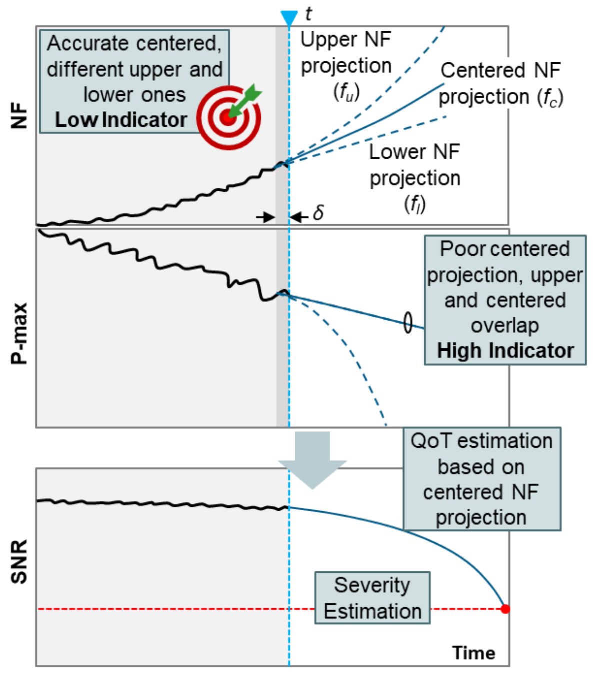

t
Accurate centered, different upper and lower ones
Low Indicator
Upper NF projection (f_u)
Centered NF projection (f_c)
Lower NF projection (f_l)
δ
NF
P-max
Poor centered projection, upper and centered overlap
High Indicator
QoT estimation based on centered NF projection
SNR
Severity Estimation
Time

Figure 22. Example of identification and severity estimation at time t.

Once the *Soft-Failure Identification* block has found enough evidence of the modeling parameter explaining the soft-failure, the *Severity Estimation* block uses the centered projection of the evolution of such modeling parameter to estimate the likely evolution of the SNR for the affected lightpaths (i.e., those being supported by the localized resource). Analyzing such evolution, it is easy to check whether any lightpath would exceed a given threshold. For instance, the minimum SNR defined by its modulation format and bit rate. The next subsections detail these two MESARTHIM blocks.

Soft-failure Identification

Algorithm 6 is used to identify the most likely modeling parameter(s) explaining the observed performance degradation; it receives as input the resource responsible for the soft-failure and the number *T* of historical monitoring samples, and, if sufficient evidence is found, it returns the modeling parameter identified as a potential failure.

Algorithm 6. Soft Failure Identification

```
INPUT: $r, T$                OUTPUT: $param$
1:  $v \leftarrow$ getModelingParameters($r$)
2:  $r$.evol $\leftarrow$ configDB.SELECT($r, T$)
3:  for each $i$ in $v$ do
4:      $Y \leftarrow$ getTimeSeries($r$.evol, $i$)
5:      $Y_{fit} \leftarrow$ Y[1..$T$-$\delta$]
6:      $Y_{eval} \leftarrow$ Y[$T$-$\delta$+1..$T$]
7:      $F \leftarrow$ fitTimeDependentFunctions($Y_{fit}, \delta$)
8:      $F^*=<f_l, f_c, f_u>\leftarrow$selectLikelyEvolution($F,T$)
9:      $r.\phi[i]\leftarrow \phi(F^*,Y_{eval})$ (Eq. (6))
10: $v^* \leftarrow$selectHighIndicatorParameters($r$)
11: for each $i$ in $v^*$ do
12:     $r$.accum$\phi[i]$+=$r.\phi[i]$
13:     if $r$.accum$\phi[i]>thr$ then $v\leftarrow v\backslash i$
14: if $|v| = 1$ then return $v$.getFirst()
15: return $\emptyset$
```

The algorithm starts by retrieving the modeling parameters *v* that characterize the state of the resource, as well as the evolution of each modeling parameter in the last *T* measurements (lines 1-2 in Algorithm 6). Then, each parameter *i* in *v* is evaluated independently to compute a score indicating how likely it is, that such parameter is responsible for the resource failure. Specifically, the time series of the parameter *i* is selected and split into two portions: one with the first T-δ data points (Y_{fit}) used for fitting and a second one (Y_{eval}), with the last δ measurements, used for evaluation and indicator computation (lines 3-6) (dark grey area in Figure 22). Using the Y_{fit} segment, a set of time-dependent functions *F*, where the parameter value is modeled as a function of time, is obtained by applying Holt-Winters exponentially-based modeling and polynomial fitting in a wide range of degrees (e.g., from 1 to 7) (line 7). The set *F* contains all functions with similar goodness-of-fit, i.e., +/- 5% of variation in terms of Pearson correlation coefficient. Moreover, functions are evaluated in the time interval of *Y* to verify that trend is always monotonic; otherwise, the function is discarded. Then, three of those functions are selected to be the likely projections illustrated in Figure 22 (line 8). This selection contains: *i*) f_l (lower) and f_u (upper), with the functions with the lowest and highest value at the current time, respectively; and *ii*) f_c (centered), with the function with the smallest number of coefficients providing maximum Pearson correlation coefficient (+/- 5%), which is assumed to be the most likely parameter evolution.

After obtaining the likely projections, they are compared with the trendline (computed using the same methodology of f_c) in Y_{eval}. This comparison is carried out using an additive multi-factorial indicator (φ) that combines several Boolean and continuous variables according to Eq. (6), where x_i represents a variable and b_i the weighting coefficient for that variable.

$$\varphi = \sum_{i=1..n} b_i \cdot x_i \tag{6}$$

Table 9 briefly describes the considered variables, sorted in the descendent degree of importance. Two Boolean functions are used: x_1 returns *True* if there is no evidence of degradation in the parameter, whereas x_4 checks if any of the upper and lower projections is the same function as the centered one. In addition, two continuous variables are defined: x_2 accounts for the relative Mean Square Error (rMSE) of the data in the evaluation portion with respect to the centered projection, whereas x_3 returns the minimum rMSE of the data with one of the projections. Note that if measurements in the evaluation portion follow a trend showing a parameter degradation similar to the projections (preferably, the centered one), the indicator remains low; otherwise, the indicator increases, indicating that the observed behavior differs from the expected one and therefore, the likelihood of the parameter not to be the cause of the failure increases.

Table 9. Indicator function components

i	**Description (x_i)**
1	[boolean] Projections f_l, f_c, f_u do not show parameter degradation
2	[continuous] rMSE(Y_{eval}, f_c)
3	[continuous] min(rMSE(Y_{eval}, f_i)) | i = {l,c,u}
4	[boolean] $f_l = f_c$ OR $f_u = f_c$

Once the indicator is computed for all modeling parameters, characterizing the state of the resource, the selection of parameters, whose indicator is significantly higher than the rest, is conducted (line 10). Thus, considering φ_{min} as the minimum indicator value of a parameter, the limit φ_{min}+$\Delta\varphi$ is defined as the reasonable limit for parameters with low indicator, being all parameters above φ_{min}+$\Delta\varphi$ selected as those with high indicator. For the resultant set of parameters (if not empty), the cumulative indicator is updated by adding the current one (lines 11-12). Finally, the cumulative indicator is compared with an identification threshold *thr* and, if exceeded, the parameter is removed from the candidate parameter set (line 13). The identification is considered positive when just one parameter remains as a candidate and it is eventually returned (line 14). Otherwise, the algorithm returns no identification (line 15). It is worth noting that the value of *thr* needs to be high enough to clearly identify the parameter without false positives.

Severity Estimation

Once a modeling parameter has been identified as the cause of the observed degradation, the severity estimation algorithm is executed. This method uses the projection of the modeling parameters to estimate when some affected lightpath will cross the FEC limit. Although the severity estimation could be based on the projected evolution of the modeling parameters, defining a threshold for each one is not easy as different lightpaths are impacted differently by its degradation.

Algorithm 7. Severity Estimation

```
INPUT: r, P, T, param      OUTPUT: estimatedTime
1:  r.evol ← modelingConfigDB.SELECT(r, T)
2:  $f_c$←getCenteredProjection(r.evol[param], timeWindow)
3:  for t in [1, timeWindow] do
4:     SNR ← $\hat{S}(P \mid f_c(t))$
5:     for each p in P do
6:         if SNR[p] < p.minSNR then return t
7:  return INF
```

Algorithm 7 details the pseudocode; it receives as input the resource responsible for the soft-failure, the list of lightpaths supported by such device, the number *T* of historical monitoring samples, and the identified modeling parameter, and it returns the estimated time that the soft-failure will cause a major impact on, at least, one of the lightpaths. The algorithm first retrieves the last *T* configuration values from the *Device Modeling Config DB*, which are used to compute the centered projection f_c for the considered input parameter on a given time window (lines 1-2 in Algorithm 7). The projection f_c is used as the input parameter to estimate the SNR for the list of lightpaths; in case that the estimated QoT for any of the lightpaths falls behind the minimum SNR, the algorithm returns the estimated time of this event to happen (lines 3-6); otherwise, it returns a large value that exceeds the considered time window (line 7).

4.1.4 Results

Experimental assessment

Being the basis of the MESARTHIM methodology, the *Find Modeling Configuration* has been evaluated experimentally in the testbed depicted in Figure 23. Two commercial coherent transponders (labeled TRX-1 and TRX-2), with optical line interface at 100G (32-GBd Quadrature Phase-Shift Keying -QPSK), have been connected using an optical multi-span link. The pair of transponders has been equipped with a specifically designed driver, enabling both the configuration and the real-time monitoring of the SNR; the generated signal is filtered by a WSS Wave Shaper device. The optical link consists of 4 spans, each realized by an 80 km single-mode-fiber spool, for a total distance of 320 km. Five EDFAs have been used to compensate for the power attenuation; all being the single stage with gain in the range 15-25 dB. OA1 was configured with a constant 17.5 dB gain to compensate for the filter insertion losses and the other OAs with a constant gain of 16 dB to compensate entirely for the fiber losses within the span.

Two different experiments were carried out. In the first, we reproduced the effect of a filter detuning. We configured the TRXs at 193.9 THz and the WSS with a bandwidth of 50 GHz, collecting the estimated SNR at TRX-2. Then, we reconfigured the WSS filter, reducing the bandwidth with steps of 2 GHz, until the operational status of TRX-2 card was down. GNPy was used as a tool to estimate the expected QoT for the lightpath. In order to estimate the QoT, together with the scenario that includes fiber types and span length, modeling parameters of the WSS, OAs and the TRXs are provided as input to GNPy; it estimates the QoT using the generalized GN model, which considers both the ASE noise and NL Interference (NLI) accumulation.

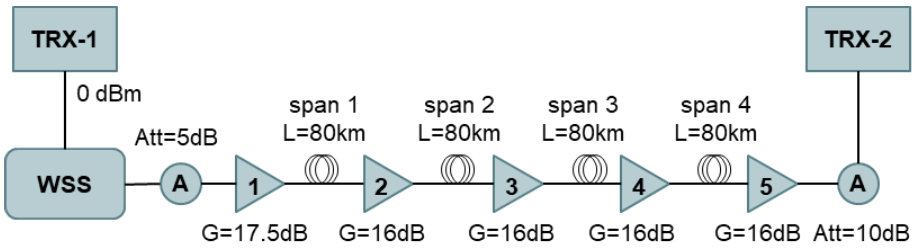


Figure 23. Experimental testbed.

Assuming that the device responsible for the SNR measured in TRX-2 is unknown, we run the *Find Modeling Configuration* block of MESARTHIM to estimate the most likely modeling configuration of the WSS and one of the OAs in the optical link. Figure 24 presents the results obtained when the observed SNR is explained by a reduction in the OSNR of the A/D WSS (Figure 24a) and by an increased value of the NF in one of the OAs (Figure 24b). We observe that changes in both modeling parameters could explain the evolution of the observed SNR in the lightpath with minimal error, being the resulting values of the modeling parameters within a feasible range. Note that with just one lightpath in this experiment, it is not possible to make any localization, as any of the devices could be responsible for the observed reduction in the SNR.

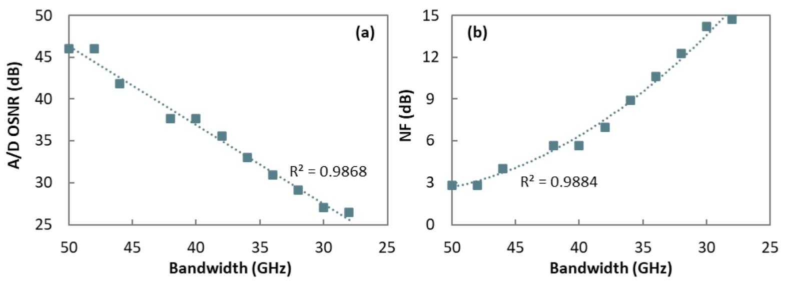


Figure 24. Modeling parameter value vs. bandwidth for A/D WSS OSNR (a) and OA NF (b)

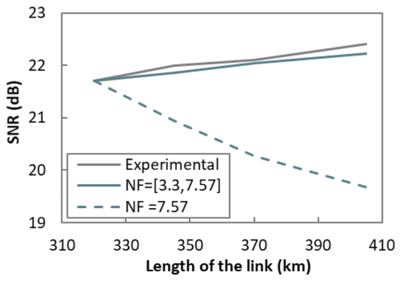


Figure 25. SNR vs. link length

In the second experiment, we slightly changed the multi-span link scenario with respect to that in Figure 23, by adding 5 dB attenuators before spans 2-4 emulating an additional 25 km in each span. Therefore, the gain of OAs 3-5 had to increase to 21 dB to compensate for the increased losses that also affected the NF of our amplifiers (which is inversely proportional to the configured gain). The experiment was carried out in three steps, where one span was modified at a time. For each step, we continuously collected the estimated SNR at TRX-2, detecting the variation of the transmission metrics during the testbed evolution. Figure 25 presents the results from the *Find Modeling Configuration* block when the length of the spans 2-4 was increased to an equivalent of 105 km and consequently, the gain of OAs 3-5. At each step, the module was able to explain the increment in the SNR of the lightpath by a reduction in the NF of the related OA. For illustrative purposes, the estimated SNR of the lightpath that would be obtained by keeping the NF constant is also represented in Figure 25. These two experiments assess the high accuracy of the *Modeling Config Search* for estimating the modeling parameters of the devices.

The next subsections evaluate the MESARTHIM methodology through simulation.

Simulation environment

For the simulation, we selected a German-like network topology with 17 nodes and 26 bidirectional links (see Figure 26). 136 bidirectional lightpaths, representing all the origin-destination pairs, were established through the shortest route in terms of hops. Figure 27 plots the number of lightpaths that every link in the network is supporting (which is particularly important for failure localization). For the sake of simplicity in the analysis of the results, we assume that all signals use the QPSK modulation format.

The optical data plane was simulated by a GNPy instance. We generated SNR measurements for every lightpath by varying every modeling parameter of every intermediate OAs and A/D WSSs in the ROADMs in the network, independently. With these measurements, a set of realistic types of failures (use cases) affecting optical devices were reproduced by forcing the modeling parameters of the selected devices (*NF* and *P-max* in the OAs and *OSNR* in the WSSs) to vary over time. From the different variations that might happen, we focus on *gradual* variations, i.e., those soft-failures that can eventually degenerate into hard-failures. Among all possible gradual degradations, we focus on the exponential increase (NF) and logarithmic decay (P- max and OSNR), since both types of degradations accelerate in time and hence, it is crucial to anticipate their detection and localization as much as possible. Finally, variability to the SNR samples was added in the form of random noise.

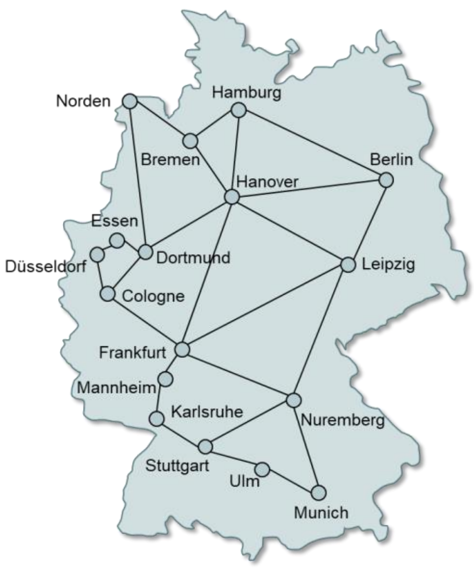


Figure 26. Considered optical network topology.

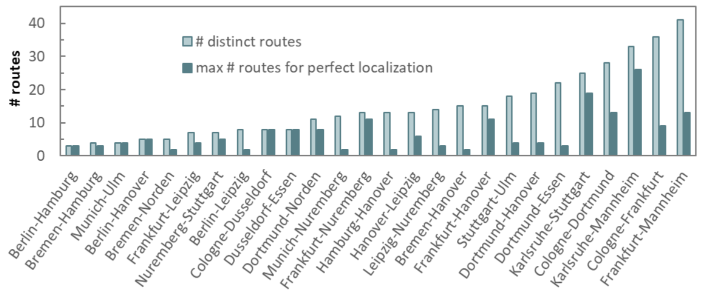
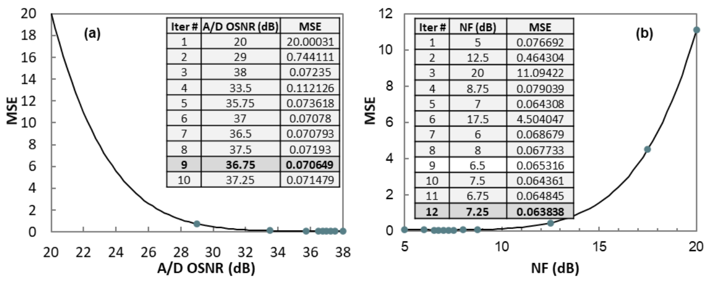


Figure 27. Number of distinct routes per link and max number of routes for localization.



Iter #	A/D OSNR (dB)	MSE
1	20	20.00031
2	29	0.744111
3	38	0.07235
4	33.5	0.112126
5	35.75	0.073618
6	37	0.07078
7	36.5	0.070793
8	37.5	0.07193
9	**36.75**	**0.070649**
10	37.25	0.071479

Iter #	NF (dB)	MSE
1	5	0.076692
2	12.5	0.464304
3	20	11.09422
4	8.75	0.079039
5	7	0.064308
6	17.5	4.504047
7	6	0.068679
8	8	0.067733
9	6.5	0.065316
10	7.5	0.064361
11	6.75	0.064845
12	**7.25**	**0.063838**

Figure 28. Modeling Config Search.

The resulting samples were stored in the simulated control plane and fed the module implementing the MESARTHIM methodology. In the case of the *SNR- wise Surveillance* algorithm, the *SNR_threshold* (line 3 in Algorithm 2) was set to the expected SNR for each given lightpath minus a fixed value that exceeds the random variations introduced by the monitoring generator.

The next subsections present the obtained results for the different procedures of the MESARTHIM methodology based on this simulation setup.

Surveillance and Device Configuration Estimation

Let us first illustrate the convergence of the *Modeling Config Search* algorithm with an example entailing two sets of lightpaths. We are interested in finding the most likely modeling config for the OSNR of an A/D WSS and for the NF of an OA (each supporting one of the sets of lightpaths), given its monitored SNR. Figure 28 plots the MSE as a function of the configuration value, as well as those values explored by the algorithm for the two optical devices. The inset tables specify the MSE values, whereby the configuration that gives the minimum MSE is finally selected. From these results, as well as from those in the experimental assessment, we conclude that the algorithm converges in the whole range of the true soft-failure origin, regardless of the selected QoT parameter.

We now focus on the evolution of the SNR over time for the defined use cases. The graphs in the upper row in Figure 29 present such evolution, where, for the sake of clarity, we plot only one sample of the affected lightpath. Note that only the lightpaths affected by the failure will experience an evolution in their SNR, whereas the rest of the lightpaths will show no variation over time other than a random one plus some uncorrelated spurious measurements introduced by the monitoring generator. The time in the graphs is normalized, as the time-scales for the considered soft-failures are different, ranging from days to months or even years. The evolution of the modeling parameters is shown in the bottom-row graphs, where the actually programmed value and the interval of values [max, min] estimated by the *Find Configuration* block is plotted.

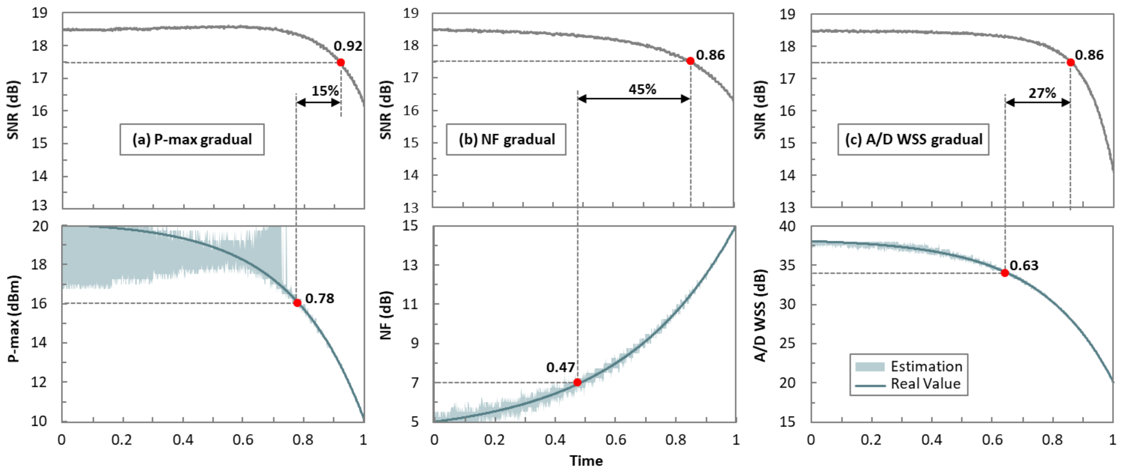


Figure 29. Evolution of monitored lightpath SNR against time and estimation of modeling parameters.

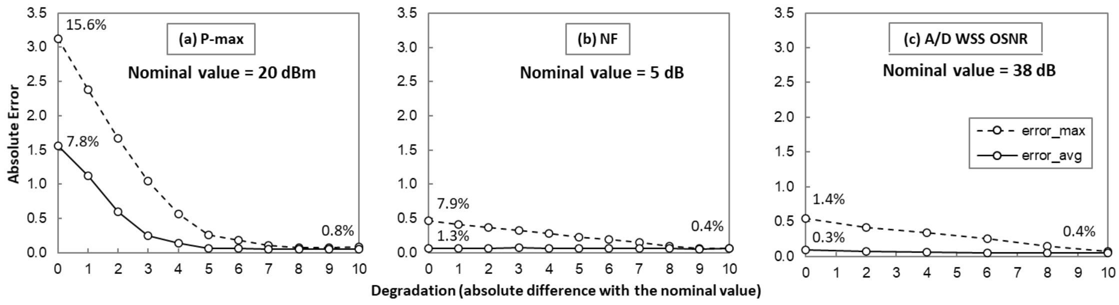


Figure 30. Absolute and relative modeling parameter estimation error.

We observe that the range of possible values of the modeling parameters is tighter when the value of the parameter deviates from its nominal one. In addition, the range of possible values for the modeling parameters is different for the different parameters, being the P-max of the OAs the one with the largest range. This might have a clear impact if the detection of the soft-failure is performed by tracking the evolution of that parameter. Figure 30 complements Figure 29 by plotting the maximum and average error in the estimation of the modeling parameters as a function of the magnitude of the degradation. We observe in Figure 30a that both maximum and average P-max estimation errors are high for low degradation magnitudes (15.6% and 7.8%, respectively). In contrast, the average error for NF and A/D WSS (Figure 30b-c) are remarkably low and almost constant for the degradation magnitudes studied. Figure 30 also confirms the observation regarding the error in the estimation of the value of the modeling parameters greatly reduces with the magnitude of the degradation, which is a very promising result and it can be exploited for soft-failure localization and identification.

Some conclusions can be drawn from the results obtained so far: i) the proposed method for estimating the value of modeling parameters of the devices has shown remarkable accuracy in the experimental tests, which has been confirmed by simulation for all failure use cases; ii) in general, the estimation interval is tighter when the impact of the value of the parameter on the observed SNR is higher; iii) in the specific case of the maximum power of the OAs, the range of values that result in the SNR values observed is large when the observed SNR remains around the nominal value. However, when SNR degrades with evident trend, the correlation between P-max and SNR becomes larger.

To help developing intuition about the differences that can be expected by analyzing the SNR of the lightpaths and the value of the modeling parameters of optical devices, Figure 29 compares the time to detect a degradation by analyzing the measured SNR and the value of the modeling parameters. For the sake of simplicity, let us assume that degradation detection is performed by threshold crossing; the threshold was set to 1 dB below the nominal SNR value for the lightpaths, not below a minimum SNR resulting in a pre-FEC BER over $4\cdot10^{-3}$ for QPSK signals. The thresholds related to modeling parameters were defined as a percentage of the variation range given by the nominal value and the extreme value. Values are selected to reduce detecting false degradations: *i*) for P-max it was set to 40% in the interval between the nominal value (20 dBm) and the extreme value (10 dBm) due to the large range of variation observed; *ii*) for NF of the OAs, the percentage was set to 20% in the interval between the nominal (5 dB) and the extreme value (15 dB); and *iii*) for the OSNR of the A/D WSSs, the percentage was set also to 20% in the interval between the nominal (38 dB) and the extreme value (20 dB).

With these values, the detection of the degradation in the case of the SNR of the lightpath happened at normalized times 0.92, 0.86, and 0.86 for the P-max, NF, and A/D WSS OSNR gradual soft-failure use cases, respectively. This contrasts with the detection at times 0.78, 0.47, and 0.63 when the analysis was in the value of the P-max, NF, and OSNR of the A/D WSSs, respectively, which results in anticipation between 15% and 45%. Note that such anticipation is enabled by the different evolution of modeling parameters and their non-linear impact on the SNR of the lightpaths.

Soft-Failure Localization

The above discussion considered the time for the detection only. Note that soft-failure location (Algorithm 4) requires several lightpaths to find the common resources in the network topology. When the evolution of the monitored SNR changes suddenly, *SNR-wise* surveillance (Algorithm 2) collects enough lightpaths to easily localize the failure; however, under a gradual degradation, the lightpaths exceeding the threshold might be not enough for the localization.

Figure 27 includes a study of the maximum number of distinct routes that need to be considered to unambiguously identify every link as responsible for a soft-failure, considering that longer lightpaths will be more affected by device degradations. For the study, we selected all the links in the network, together with an incremental number of lightpaths selected by their total length, and fed Algorithm 4 for the (multiple) soft-failure localization (i.e., 26 soft-failures were localized with a single execution of Algorithm 4). For the localization, not only the number of lightpaths is important, but also their routes. We repeated the experiments with the lightpaths sorted in inverse order, i.e., assuming that shorter lightpaths would exceed the (relative) threshold first. The results showed that all soft-failures could be perfectly localized when the resources of the two shortest lightpaths were analyzed.

Table 10. Examples of Soft-Failure Localization

Time	Degraded paths	Common Resources
Failure in OA in link Frankfurt-Mannheim		
0.86	1	2 TRXs, 2 A/Ds, 1 Link
0.90	2	2 Links
0.93	4	1 Link (Failure in Frankfurt-Mannheim)
Failure in A/D WSS Düsseldorf		
0.86	1	2 TRXs, 2 A/Ds, 1 Link
0.91	2	1 A/D WSS (Failure in Dusseldorf)

With the above in mind, let us now show how the *SNR-wise* surveillance and localization evolves over time. Table 10 presents the results from two different failures. The first failure was in an OA in the link Frankfurt-Mannheim (supporting 41 distinct lightpaths), and the second in the A/D WSS in the Düsseldorf ROADM (supporting 16 distinct lightpaths). For each failure, each row shows the detection time when the measured SNR of some new lightpaths is below the threshold (bear in mind that the threshold is relative to the expected SNR for that specific lightpath). However, the localization of the soft-failure is not successful until just one resource (assuming that it is responsible for the failure) can be identified. Such identification happens at normalized times 0.93 and 0.91 for the failures in the OA and the A/D WSS, respectively. Note that the number of lightpaths needed to localize the soft-failure, although small, adds some extra time that could be of paramount importance for the impact on the network.

In contrast, soft-failure localization under the *Modeling-wise* approach considers all the lightpaths supported by such resource, as it analyzes the estimated evolution of the modeling parameters by resource. In consequence, Algorithm 4 under the *Modeling-wise* approach can localize the cause of degradations in their very early stages (between 31% and 49% with respect to the *SNR-wise* approach).

As a final remark, it should be noted that even though the *Modeling-wise* approach considers all the lightpaths supporting a given resource, unambiguous localization might still not be possible if few lightpaths with distinct routes are established. Figure 27 shows, that for some links in the considered scenario, unambiguous localization of a soft-failure is only possible when all supported lightpaths are analyzed. Therefore, in case that not all these paths are established, the localization procedure (Algorithm 4) would be unable to localize the failure. E.g., let us consider the soft-failure in the link Frankfurt-Mannheim in Table 10 and imagine that only one lightpath is supported by such link. Then, a degradation in that lightpath can be explained by degradation in more than one resource. This highlights the need for additional procedures to be applied for those scenarios which result in ambiguous localization.

Identification and Severity Estimation

Once the degradation has been localized, let us focus on the identification of the modeling parameter responsible for such degradation. We assume that the device explaining the observed degradation is an OA. According to the defined notation, identification performance is clearly dependent on the configuration of all coefficients related to indicator φ. Several configurations were tested to find the one giving the desired importance to every component, ensuring that parameters with a high indicator are correctly selected, while guaranteeing no false positives. Such configuration is $<b_1, b_2, b_3, b_4, \Delta\varphi, thr> = <20, 10, 2, 1, 5, 30>$, which will be used hereafter.

Figure 31 shows the obtained results for a gradual degradation caused by NF, whereas Figure 32 shows the results for a gradual degradation caused by P-max. In both cases, Figure 31a and Figure 32a plot the evolution of the accumulative indicator with time, and the decision threshold *thr*. Both indicators remain clearly under the threshold until time around 0.32, where enough evidence of the cause of the degradation is found. Note that the time of such identification represents 32% of anticipation compared to the earliest time obtained for degradation detection (at time 0.47) using a threshold on the evolution of the input parameter and 64% compared to the earliest detection time using a threshold on the evolution of the SNR. Figure 31b-c and Figure 32b-c show the computed projections for the two modeling parameters under study and for both failures; note that these figures are similar to Figure 22. The projections are plotted for a long window (around 0.25 time units) for clarity purposes, although the value used for fitting and evaluation was δ =0.03 normalized time units.

In the case that the failure is a consequence of the NF degradation (Figure 31b-c), we observe that the centered projection f_c for the NF parameter is highly accurate and clearly between the upper and lower ones, which results in the minimum indicator parameter. On the contrary, P-max indicates an evident but less accurate projection and moreover, f_c overlaps with f_u, which increases its indicator above 5 units from the one of NF. As a consequence of this, P-max is selected as a parameter with a high indicator, so its accumulative indicator increased. Conversely, in Figure 32b-c the minimum indicator is that of P-max, where there is an evident degradation for f_l. Note that this indicator is much lower than that computed for NF, where no degradation is observed, thus exceeding by far the indicator limit. In conclusion, we see how the proposed methodology allows discriminating the actual failing parameter and perform a fine failure identification.

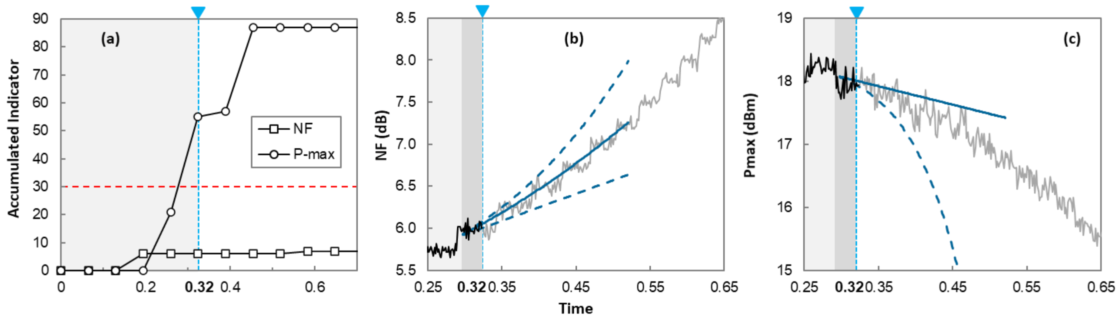


Figure 31. NF Gradual Soft-Failure Identification.

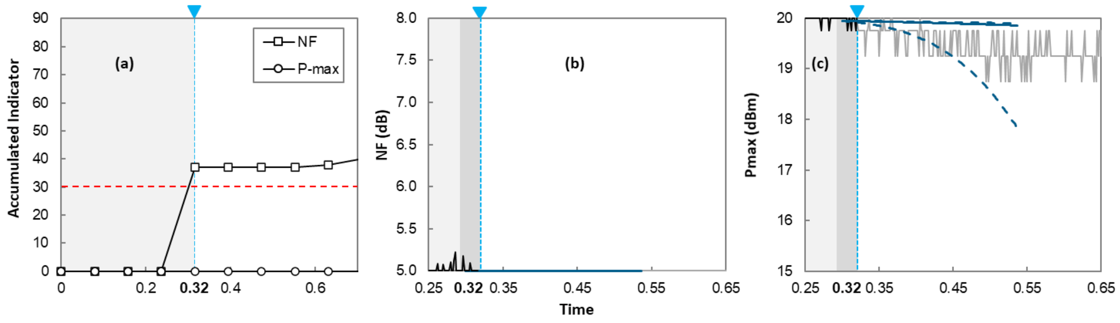


Figure 32. P-max Gradual Soft-Failure Identification

We can take advantage of the identification method to implement an alternative localization method that can be applied when not enough lightpaths with distinct routes are established in the network. This allows the unambiguous localization of a soft-failure, as previously motivated. In this case, we assume that only lightpaths between the two end ROADMs of the link Frankfurt-Mannheim are established and have considered soft-failures in one of the supporting devices (OA, TRX or A/D WSS), in line with Table 10. In this case, we execute Algorithm 6 considering a single *virtual* resource that abstracts the supporting optical device types. The algorithm returns the most probable modeling parameter among those of OA, TRX and A/D WSS, which helps to reduce the number of devices to be analyzed manually.

Figure 33 shows the performance of the identification procedure for soft-failure localization. Three scenarios are considered for the real cause of the soft-failure: *i*) gradual NF degradation in an OA (Figure 33a), *ii*) gradual OSNR degradation in an A/D WSS (Figure 33b), and *iii*) gradual OSNR degradation in a TRX (Figure 33c). We observe that the accumulated score clearly increases for the two types of devices that are not the real cause of the failure for all the analyzed scenarios. This happens at time 0.3 in all the cases.

Finally, Figure 34 presents the obtained results for severity estimation for gradual degradation of P-max, NF and A/D WSS OSNR as a function of the time. The plots show the evolution of the estimated time with the real-time, where the area in grey color highlights the real-time when the soft-failure degenerates into a hard-failure ±5%. We observe that the estimated time takes a large value when the estimation does not observe a major impact in the selected *timeWindow* for any of the affected lightpaths, and rapidly converges to the real degeneration time. In fact, assuming that the severity is accurately estimated after two consecutive executions returning estimated times close enough one to the other, anticipation over 42% to the degeneration time are obtained, which leave enough time to plan the adequate maintenance operations.

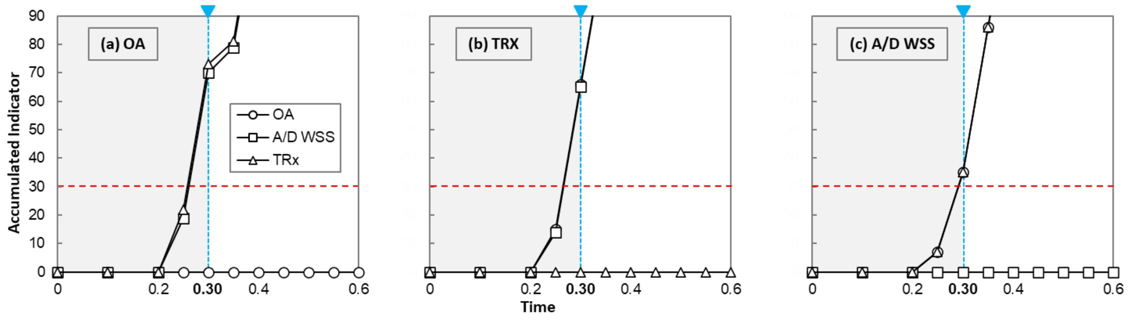


Figure 33. Localization by identifying the Soft-Failure.

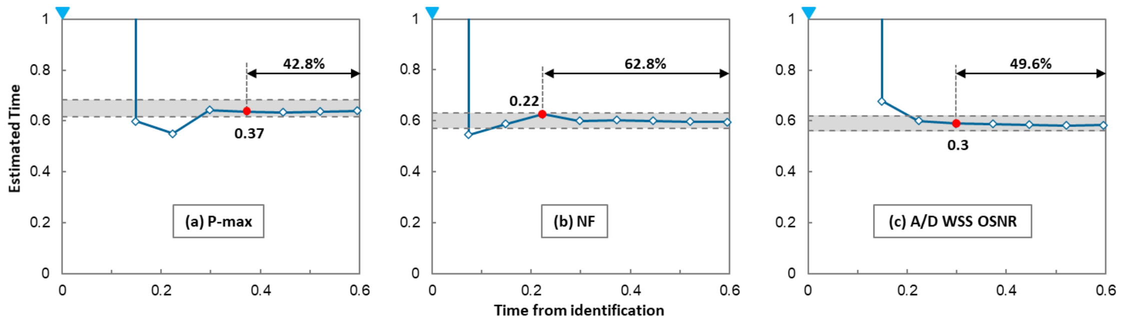


Figure 34. Severity Estimation.

4.1.5 Concluding Remarks

QoT estimation is typically carried out during the provisioning phase and in-operation planning to ensure that computed lightpaths will provide zero post-FEC errors, assuming some values for the QoT model input parameters related to the optical devices in the network. In this section, QoT estimation was used for the reverse process, i.e., given the real measured QoT of a set of lightpaths, we were interested in estimating the value of the modeling parameters of the optical devices.

Because of the non-linear relation between lightpaths QoT and the value of the modeling parameters, the ability to estimate the value of such parameters opens the opportunity to analyze its evolution, which can be as a result of, e.g., aging, temperature variations, etc. The proposed MESARTHIM methodology combines analysis of the evolution of the monitoring QoT and their transformation into the estimated modeling parameters space, not only for the degradation detection, but also for its localization, identification, and severity estimation.

After the experimental assessment of the method for estimating the modeling parameters of the devices, the MESARTHIM methodology was evaluated through simulation. The methodology demonstrated remarkable anticipation in failure detection and localization by analyzing the estimation of the value of the modeling parameters of the devices. A simple example showed the reason behind such potentials in the different evolution of the modeling parameters and the lightpaths SNR. In addition, accurate cause identification based on the analysis of the projected evolution of the modeling parameters was demonstrated, which enabled the estimation of the severity in terms of the time when the soft-failure degrades into a hard-failure. Such severity estimation allows planning maintenance, as it largely anticipates degradation.

Table 11 summarizes the main characteristics with pros and cons of the MESARTHIM methodology.

Table 11: Summary of the MESARTHIM Methodology

Method	Degradation Detection and Failure Localization	Cause Identification and Severity Estimation
Analysis in the lightpaths' SNR space	• SNR-wise surveillance finds common resources in sets of affected lightpaths by analyzing its SNR. • Analyzes all lightpaths. For failure localization, the algorithm needs that several lightpaths to be affected.	• No cause can be identified. • QoT can be estimated based on the projected evolution of the SNR.
Analysis in the devices' modeling parameters space	• Modeling-wise surveillance analyzes the value of the modeling parameters for all network devices. • The algorithm detects degradations and localizes their sources very ahead in time.	• Cause identification based on the projected evolution of the modeling parameter of the device where the failure is localized. • Severity estimation difficult to estimate by analyzing the evolution of the input parameters.

4.2 RESTORATION

To meet the stringent demands of the evolving B5G communication traffic, which requires networks to support high capacity, low latency, and ultra-high reliability, it is imperative that optical networks must ensure effective and reliable data transport mechanisms. One critical area highlighted for development is failure management, which is essential because outages in optical connections can severely impact numerous users, applications and services.

Noteworthy is the growing interest in applying ML techniques to automate various control operations in optical networks, especially in the area of failure management. Previous research has delved into ML strategies that are both proactive (e.g., performance monitoring and failure prediction) and reactive (e.g., failure detection, localization, and identification) [Mus19]. However, the use of ML, and more specifically Reinforcement Learning (RL), in the subsequent restoration process of lightpaths disrupted by network failures remains an underexplored area. This gap presents a significant opportunity for advancing the reliability and efficiency of optical networks through intelligent automation.

This section presents the evaluation of a DRL-based agent specifically designed for the autonomous restoration of disrupted lightpaths following an optical link failure [Her22].

4.2.1 RL agent for restoration in Optical Networks

The optical network is conceptualized as a set of flexi-grid ROADMs connected by C-band optical links. When lightpath requests are made, specifying source, destination, and required bandwidth or optical spectrum, a control system is tasked with handling resource selection and allocation. This process is facilitated by a RSA algorithm, which seeks feasible paths that meet the lightpath requirements while adhering to the spectrum continuity and contiguity constraints.

The RL agent has as its primary function the selection of the optimal restoration sequence from a set of candidate solutions provided by the Global Concurrent Restoration (GCR) algorithm during a link failure incident [Mar21]. This agent is integrated into a SDN controller, thus having access to current network state information such as topology and resource utilization.

Upon the detection of a link failure, the RL agent retrieves the GCR candidate restoration solutions. It then inputs these solutions into its neural network (NN), which evaluates each solution based on a combination of features including total restored lightpaths, total restored bandwidth, and the number of restored and unrestored lightpaths categorized by their required bandwidth in terms of frequency slots (FS). The NN produces an output that represents the chosen restoration sequence. The reward mechanism for the agent is directly proportional to the amount of bandwidth restored, incentivizing solutions that maximize network restorability. The proposed DRL agent for restoration is shown in Figure 35.

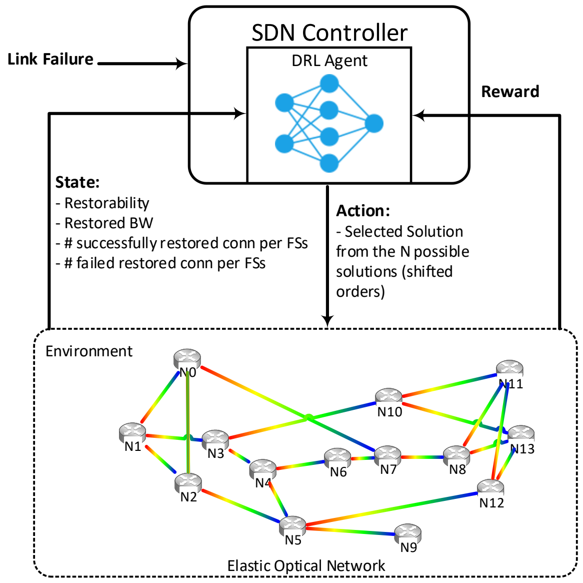


Figure 35: DRL Agent for Restoration.

4.2.2 Evaluation Setup

Dynamic requests were generated randomly, selecting end nodes for provisioning. The bandwidth demands were uniformly distributed among 1, 2, and 4 FSs. These requests arrived following a Poisson process, characterized by a mean inter-arrival time (IAT), with the connection durations governed by an exponential distribution with a mean holding time (HT).

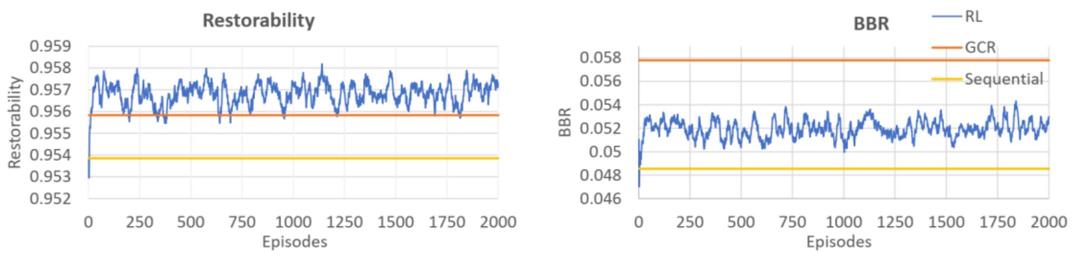
The occurrence of link failures was modeled as a Poisson process, with failure inter-arrival time (FIAT) and failure durations following an exponential distribution with a failure hold time (FHT). A failed link was randomly chosen within the 14-node NSFNET topology shown in Figure 35, capable of accommodating 100 FSs. The RSA algorithm then iterated over the k shortest candidate paths to find feasible routes for restoration, sorting these paths based on the end-to-end delay.

The DRL agent was implemented based on the Asynchronous Advantage Actor Critic (A3C) algorithm and underwent training across 2000 episodes, with each episode involving the provisioning of 10,000 new connection requests. The agent used its neural network to decide the optimal restoration sequence whenever optical links failed.

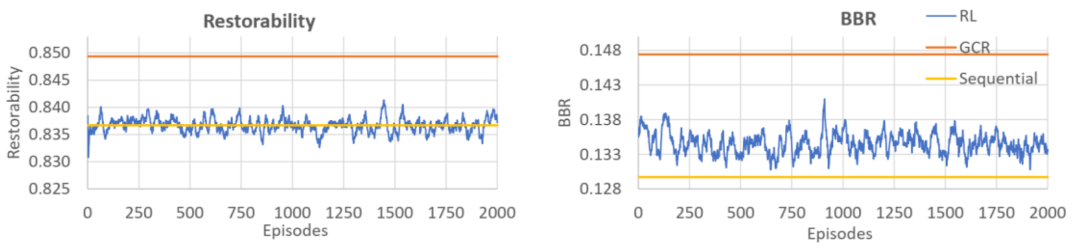
4.2.3 Experimental Results

The evaluation focused on the restoration capabilities, particularly across different traffic loads and failure duration scenarios. An essential aspect of the evaluation was comparing the restorability and the BBR of the proposed RL agent against both the GCR heuristic algorithm and a sequential approach.

Figure 36 depicts the DRL agent's relative success in improving restorability and BBR under different traffic loads compared to the GCR and sequential approaches. In scenarios with low FHT=2 and low traffic loads = 200 Erlangs shown in Figure 36 (a), the DRL agent showcased an improvement in restorability over the sequential approach and slightly outperformed the GCR method, achieving an average restorability of 0.957 compared to GCR's 0.956 and the sequential approach's 0.954. However, when it came to the BBR, the DRL agent only surpassed the GCR's performance without outperforming the sequential method. The BBRs achieved by the DRL agent, GCR, and the sequential approach were approximately 0.052, 0.058, and 0.048, indicating a nuanced outcome, wherein the DRL agent's advancements in restorability did not necessarily translate to superior BBR performance across all compared approaches. This may be due to the fact that the restorability values are high and similar among the samples used in the agent training. On the other hand, for the scenario with high FHT of 32, and a high total traffic load of 500 Erlangs, the performance of the DRL agent in terms of restorability and the BBR is presented in Figure 36(b). In this setup, the DRL agent's restorability is comparable to that achieved by the sequential algorithm, with a restorability value of approximately 0.837. The GCR algorithm, however, achieves a slightly higher restorability, with a value of around 0.849, marking it as the most effective approach in terms of restoring disrupted lightpaths under these specific traffic conditions. The BBR values reported for the DRL agent, GCR, and the sequential algorithm are approximately 0.135, 0.147, and 0.130, respectively. This performance suggests that while the DRL agent manages to exceed GCR's efficiency in terms of BBR, it is not able to surpass the efficiency of the sequential algorithm, which remains the most effective method in minimizing bandwidth blockages under these conditions.



(a) IAT= 2 FIAT=16 FHT=2 200 Erlangs



(b) IAT= 2 FIAT=16 FHT=32 500 Erlangs

Figure 36: Restorability and BBR using RL agent, GCR and Sequential Approach.

By leveraging RL, more efficient pathways for restoring optical network functionality after failures were explored, contributing to the overall resilience and reliability of optical communication systems in the face of stringent future demands. The proposed RL-based solution demonstrates a capability to slightly outperform traditional heuristic algorithms in terms of restorability, indicating its effectiveness in restoring disrupted lightpaths to maintain network service continuity. However, when it comes to minimizing the blocked bandwidth ratio (BBR), the RL-based approach does not consistently outperform other methods. In some instances, its performance is worse compared to these heuristic solutions. This difficulty is partly attributed to the simplified system model and the limited number of parameters used to train the RL agent. There's an expectation that the RL agent would perform better in scenarios involving higher complexity with a larger number of variables. These complex scenarios would require understanding and capturing the correlations between various factors to make effective restoration decisions.

4.3 Failure Recovery in MANTRA Architectures

The -Metaverse ready Architectures for Open Transport – MANTRA- architecture has been defined by TIP, the main Internet Service Providers (ISP).

This architecture, following a hierarchy for controlling packet optical networks that utilize pluggable transceivers, aims to establish a comprehensive reference control architecture for multi-layer networks considering the utilization of IP over Wavelength Division Multiplexing (IPoWDM) nodes. There have been different approaches how to control MANTRA architectures, namely single (packet controller is the only one accessing the IPoWDM node, while optical control is solely responsible for managing optical resources (ROADMs, amplifiers etc) and dual where the optical parameters are exposed by the node. Figure 37, shows the hierarchical control architecture that we have implemented to reproduce the approach defined by MANTRA.

At the top hierarchical level, B5G-OPEN network planner (NetP) resides that orchestrates the network layers. Specifically, it receives on its Northbound Interface (NBI) the user-initiated orders (e.g., activation of a new connectivity service) and, through its Southbound Interface (SBI), it interacts with the packet SDN Controller (PckC) and with the optical SDN Controller (OptC). The PckC and the OptC are based on ONOS. Both of them have been extended with a set of REST-base endpoints to enable the orchestration performed by the NetP. Moreover, the PckC has been extended with a NETCONF driver for forwarding the packet configuration, including IP and BGP parameters received. The IP and BGP configurations received from the NetP toward the OpenConfig agent running on the IPoWDM node. For the configuration of ROADMs, in the optical domain, the OptC uses NETCONF drivers, build upon the OpenROADM models, available in the master ONOS distribution.

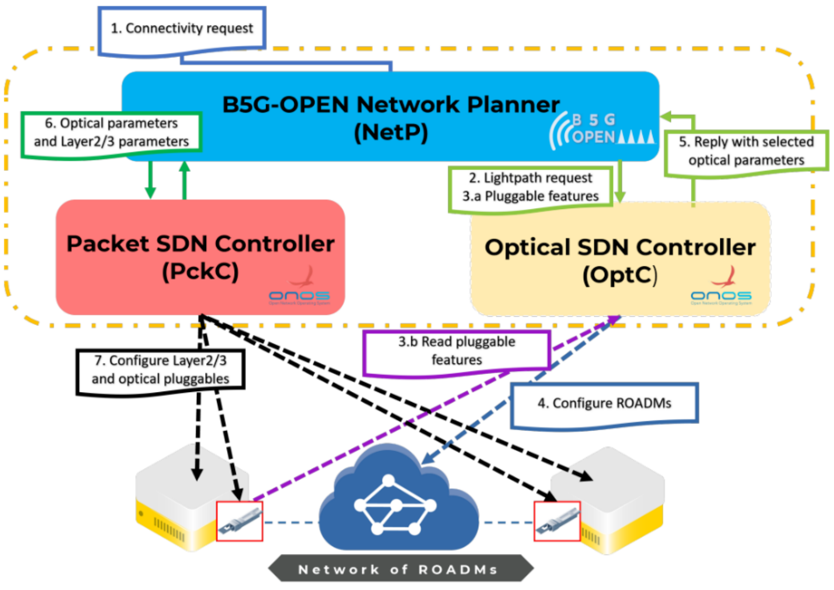


Figure 37: MANTRA Dual Architecture Implementation, with workflow steps for activating a connectivity service.

The IPoWDM nodes expose the standard OpenConfig model through a NETCONF agent deployed within the SONiC system; the emulated ROADMs exposes OpenROADM model using the same NETCONF agent implementation. The Lumentum ROADM-20 exposes a proprietary YANG model through a built-in NETCONF server. Finally, all the interactions among controllers have been implemented using custom-built REST APIs building on top of the APIs natively exposed by ONOS. Within the B5G-OPEN project there is other ongoing work for the development of a dedicated tool to be deployed on top of the OptC translating the custom-built ONOS REST APIs to standard T-API, however this component will be integrated in further work. The developed software already produced several patches to the official ONOS distribution and further contribution is planned, e.g., the REST APIs extensions for the pluggable configuration.

Figure 38 display the relevant IPoWDM node architecture. An Ethernet switch runs the SONiC network operating system on top of the Open Network Install Environment (ONIE) environment. In addition to the default SONiC applications (such as soniccfggen, syncd, swss, pmon, FRR, and the Redis database) the system includes additional containerized applications: the NETCONF agent and the coherent manager that have been specifically developed for this work. The NETCONF agent container is employed for configuring and monitoring optical pluggables. This agent utilizes the OpenConfig model for representing hardware components of the box including ports and pluggables. To prevent misconfiguration problems when multiple controllers access the node (and consequently the NETCONF agent) ownership segregation has been implemented, through the Network Configuration Access Control Model (NCACM) solution. NCACM has been designed to limit NETCONF access to specific operations and content in a user-based manner. More specifically, for each configured user (i.e., packet controller and optical controller), a set of rules is established within the NETCONF agent. These rules either permit or deny operations (such as write or read-only) on specific prefix-based configuration sections. For example, considering the Dual approach as defined in MANTRA, the OptC is provided with read-only rights on the optical parameters of coherent pluggables. Similarly, PckC is provided with writing rights on both packet ports and optical pluggables. Thus, the adoption of NCACM is a valid and effective solution to control responsibility and preserve confidentiality.

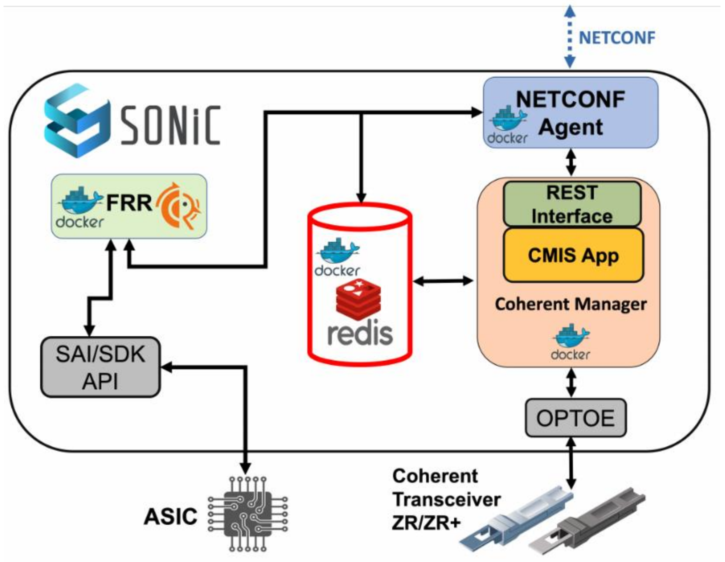


Figure 38: MANTRA Dual Architecture Implementation, with workflow steps for activating a connectivity service.

The Coherent Manager container comprises the Common Management Interface Specification (CMIS) App and the REST Interface. The CMIS App is an application that utilizes the coherent optics Python library provided by SONiC to forward received commands received by the REST interface toward the Module State Machine (MSM) and Data Path State Machine (DPSM) state machines. Moreover, it is used to read the operational parameters exposed by the pluggables (i.e., ESNR, OSNR and received power). The CMIS App directly interface to the OPTOE, which is the driver used by SONiC to manage and monitor the coherent pluggable modules. To allow other containers/applications to interact with the Coherent Manager, a REST interface is exposed. Thus, the NETCONF container, exposing the OpenConfig representation of the device, can access the pluggable modules via the REST interface allowing it to configure and monitor the optical parameters available in the pluggable modules.

4.3.1 Common Management Interface Specification (CMIS) for Pluggable Transceiver Management

This section presents the implementation of the CMIS version 5.0 and C-CMIS version 1.1 interfaces, including their respective regulating finite state machine. The CMIS interface, defined by the Optical Internetworking Forum (OIF), is becoming the de-facto standard management interface for pluggable modules. It offers a clearly defined mechanism for initializing and managing optical (and copper) modules, while also ensuring the ability to accommodate custom functionalities when needed. The electrically erasable programmable read-only memory (EEPROM) within the pluggable module is organized in pages, each with read and write capability. Each page contains a different set of information, such as: module information, Versatile Diagnostics Monitoring (VDM), current configuration, and operational state parameters. The initialization of the module is represented by the MSM, while the configuration is represented by the DPSM.

The MSM, is depicted in Figure 39 (left). It defines the initialization process between the device that hosts the pluggable module and the module itself. Once the pluggable module is inserted into the IPoWDM box, the MSN enters the INSERTED state. Consequently, the module is powered, and the MgmtInit transition is initiated. During this phase, the module initializes the Memory Map to default values and sets up the management communication interface, allowing the host to eventually manage the module. After that, the module enters the ModuleLowPwr state, during which the host can configure the module using the management interface to read from and write to the management Memory Map. If the operation is successful, the PowerPwrUp transition is triggered, and the host is informed that the module is in the process of powering up to High Power Mode. However, if it fails, the Resetting transition is triggered, clearing the Memory Map, and transitioning back to the INSERTED state. When the MSN reaches the READY state, the module is in High Power mode, and the host can initialize or deinitialize the Data Path State Machine (DPSM). If the READY state is not reached, the ModulePwrDown transition is triggered, and the module returns to the ModuleLowPwr state. DataPatg State Machine (DPSM) is illustrated in Figure 39 (right). It defines the host-module interactions required to configure the parameters within the module including transmission power, frequency, threshold alarms, and more. The module starts in the READY state waiting for a configuration request by the host, such as a change in frequency. When a configuration request is received, the DPDeinitS transition is initiated. During this phase, the module performs all the necessary de-initialization activities on all resources associated with the current configuration within the module, and the transmission power is turned off. Once the DP_Init state is reached, the DPInit Complete transition is triggered, and the module performs all the necessary initialization activities on its internal resources to apply the new configuration. Afterward, the module reaches the DP_Initialized state where it is fully operational, initialized, and ready to transmit traffic, but it is not powered. Subsequently, the DPTxTurnON transition is triggered, and the module is powered. If this phase is completed correctly, the module reaches the DP_ACTIVATED state. This indicates that the new configuration has been successfully applied to the module, and traffic can now be transmitted and received. Finally, the module is moved to the READY state through the Prepared transition, which informs the host that the module is ready to handle a new configuration.

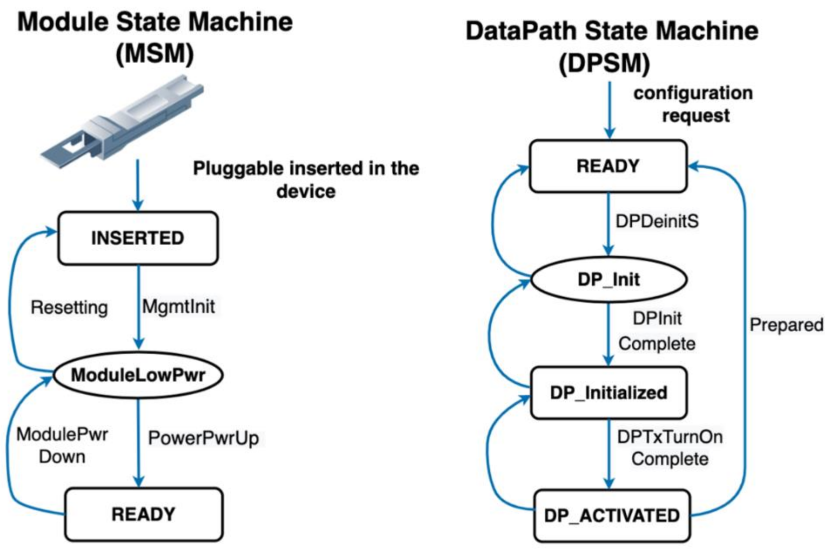


Figure 39: Module State Machines (MSM) for the initialization of the pluggable modules

Coherent CMIS (C-CMIS)

The Coherent Common Management Interface Specification (C-CMIS) extends CMIS to enable the management of digital coherent optical modules. In more detail, C-CMIS provides additional parameters relevant to the coherent pluggable in the VDM table. Besides that, C-CMIS defines a new set of tables for monitoring and configuring the media lanes, such as Media Lane Provisioning and FEC Performance Monitoring.

In 5GB-OPEN, the C-CMIS has been used to retrieve the following parameters: received power, BER, OSNR, and eSNR.

4.3.2 Connectivity setup and Failure Recovery in MANTRAS Architectures

Workflow steps for Connectivity service

The MANTRAs architectural design, illustrated in Figure 37, shows the workflow steps for activating a connectivity service. In particular, the request to establish a layer-2 connectivity between the two IPoWDM node interfaces attached to ROADMs has been considered that may imply the activation of a new digital signal rate (DSR) connectivity between a pair of coherent optical pluggables. When such connection request arrives, NetP performs end-to-end path computation across both the IP and optical layers, thus understanding if a new connection in the optical layer (i.e., a new lightpath) is required or not. If a new lightpath is required, the following steps are executed to orchestrate the connectivity setup:

a. NetP receives and processes the request;
b. NetP sends a request to Optc, for establishing a lightpath between two specific ROADMs' ports;
c. a) The features supported by the pluggables attached to such ports (e.g., modulation formats, tuneability range) are communicated by the NetP within the request. Or alternatively,
   b) the OptC can dynamically read such information as defined in the Dual MANTRA's design. In our current implementation the NetP only includes the endpoints in the request, and the OptC dynamically retrieves the tuneability range of the pluggables;
d. OptC performs routing and spectrum assignment and configures the lightpath in the traversed ROADMs. In BG5-OPEN implementation, OptC uses shortest path routing and first-fit spectrum assignment. However, it could apply impairment-aware optical path computation, e.g., making use of external tools dedicated to physical impairment evaluation and QoT estimation;
e. Once the lightpath is configured Optc sends notification to NetP including selected optical parameters (i.e.,the frequency slot in the current implementation);
f. At this point NetP sends layer-2/3 configuration request to PckC, including optical parameters to perform pluggable configuration, in BG5-OPEN implementation NetP sends a request for an IP link specifying the pluggabble end-points and the frequency slot to be used;
g. PckC performs the actual configuration of the coherent pluggables and layer-2/3 protocols (i.e., Ethernet, IP and BGP).

Failure Recovery service

Within the described control architecture, a failure along one fiber link is considered and three different procedures are experimentally compared to better understand the reachable performance, in terms of recovery time. The failure happens in the optical layer, but the recovery can be performed involving either only the optical layer, only the packet layer, or both.

1st solution. The first solution (i.e., optical restoration) is purely applied at the optical switching layer without the involvement of the PckC. Thus, it only involves the reconfiguration of ROADMs while the configuration of coherent pluggables cannot be modified. This implies that the recovery path at the optical layer must be suitable using the same frequency slot, modulation format and transmission power. The recovery workflow is quite simple, when OptC detects the failure. The workflow steps are as follows:

a. (OptC computes a new path between the same endpoints, avoiding the failed link and utilizing the same frequency slot that was previously used;
b. OptC re-configures the ROADMs to activate the new path.

2nd solution. The second solution (i.e., hybrid restoration) involves both the OptC and the PckC. This solution typically provides much more flexibility since it allows to change the configuration of pluggables allowing the utilization of different frequency slots and modulation format thus even enabling the utilization of much longer paths. However, it requires coordination among the two controllers after failure occurrence and re-configuration of pluggables, that may require several tens of seconds. The recovery workflow is as follows, (after failure detection by the OptC):

a. OptC computes a new path between the same endpoints avoiding the failed link;
b. OptC re-configures the ROADMs to activate the new path;
c. OptC communicates to the PckC (through NetP) the new optical parameters to re-configure the coherent pluggable modules;
d. PckC modifies the coherent pluggables configuration.

3rd solution. The third solution (i.e., hybrid protection) is implemented minimizing the involvement of the optical layer. It does not require heavy coordination among controllers (i.e., only failure notification is propagated) but it does require the pre-provisioning of an optical backup path (including pluggables in IPoWDM nodes), thus implying waste of resources when the network operates without failures. After failure a occurrence and detection by the OptC, the recovery workflow is as follows:

a. OptC communicates the failure to the PckC (through NetP);
b. PckC performs a layer-2/3 configuration on working and recovery pluggables so that the traffic is automatically redirected toward the pre-planned backup optical path using recovery pluggables. This latter step can be implemented in different ways, considering two alternatives:
    i. operating at layer-3, the pre-planned recovery path is implemented as an alternate BGP adjacency with higher cost, thus upon failure notification it is enough to teardown the working pluggables so that the traffic is automatically routed toward the backup adjacency;
    ii. operating at layer-2, the backup pluggables are not attached to the VLAN switching the traffic, upon failure occurrence the working pluggables are teardown and backup pluggables are added to the VLAN switching the traffic.

4.3.3 Experimental Validation and Results

Testbed design

For validation and testing a dedicated packet-optical testbed has been developed, as illustrated in Figure 40. It includes two IPoWDM nodes equipped with pluggable transceivers, two emulated ROADMs (e.g., OpenROADM NETCONF agents running in dedicated docker containers) and one physical ROADM (e.g., implemented using a Lumentum ROADM-20 device). The physical ROADM (i.e., the Lumentum) is connected at the first IPoWDM node, representing the ingress point of the lightpath. The two outgoing paths, traversing the two emulated ROADMs, are combined in a coupler going towards the second IPoWDM node. The traffic is generated via Spirent N4U connected to the SFP+ interfaces of the IPoWDM nodes. The two IPoWDM nodes are implemented using a single Edgecore switch (i.e., model AS9716-32D with 32 x 400G QSFP-DD switch ports with a Tomahawk 3 chipset) that has been sliced using two different VLANs. The switch is equipped with a pair of Cisco coherent optical pluggable modules (i.e., model 400G QSFP-DD High-Power) that support a transmission rate up to 400 Gbps exploiting 16-QAM modulation. The switch runs the SONiC operating system, in addition to the basic SONiC components two custom-built docker containers have been deployed on the switch (see Figure 17): (i) a container implementing a NETCONF server and exposing the OpenConfig model of the node to the optical controller; (ii) a container exposing REST APIs endpoints for performing BGP, CMIS and CCMIS configurations, as well as, reading operation on CCMIS parameters.

The control plane is realized by three controllers (as illustrated in Figure 37): the PckC, the OptC and the NetP. The OptC controller is based on ONOS software and runs on a workstation equipped with Intel i7-8700 12-core 3.2 GHz clock, 32 GB RAM. NetP is java based and runs on a VMWare VM with 8 CPUs and 8 GB of RAM. Finally, PckC controller runs on a VMWare VM with 8 CPUs and 8 GB of RAM. PckC is in charge to configure the packet domain and the optical pluggables interfacing to t e OpenConfig model deployed on top of the IPoWDM nodes. OptC controls the ROADMs via NETCONF, exploiting the OpenROADM modeling of the ROADMs devices. NetP orchestrates the operations enabling the communication of the optical parameters necessary to configure the optical pluggables from the OptC to the PckC.

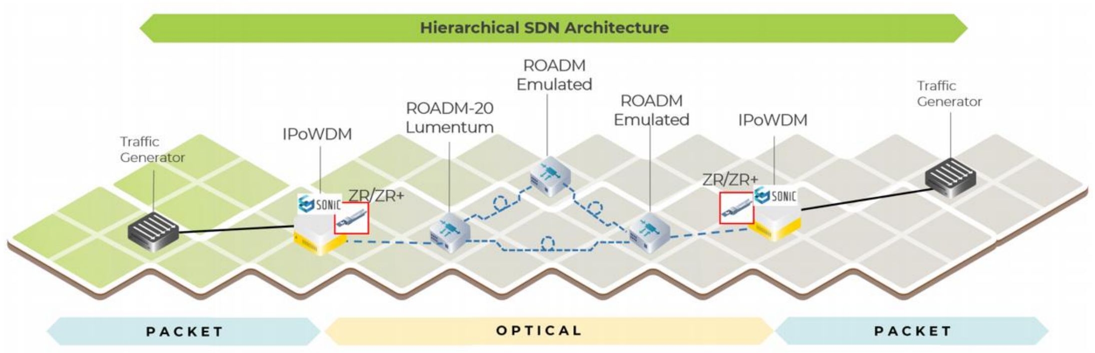


Figure 40: Top level of Testbed topology for validating connectivity and failure services in MANTRA architectures.

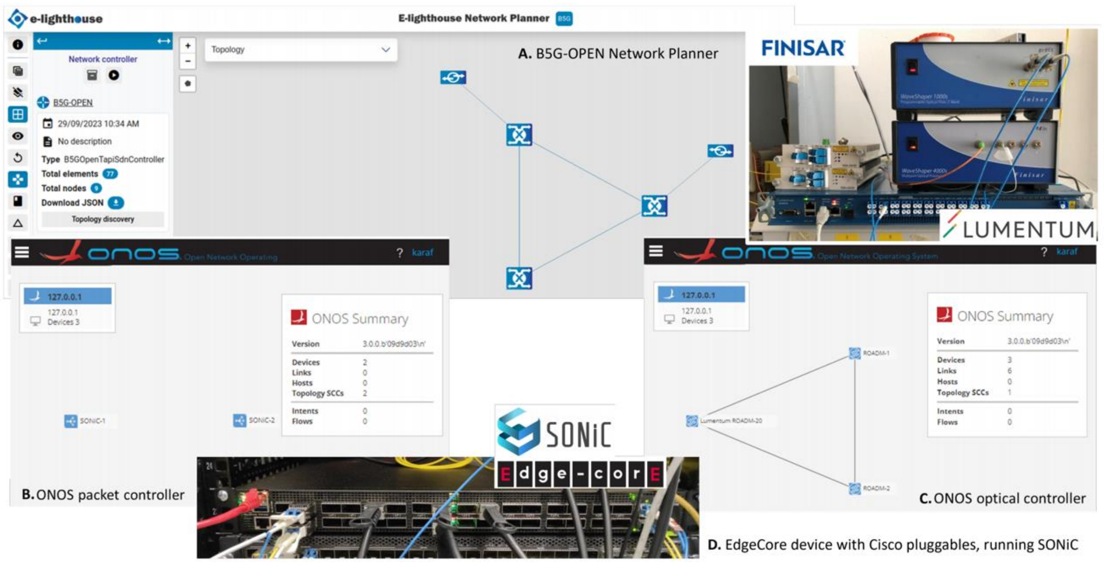


Figure 41: Screenshots of GUIs of deployed controllers and hardware devices.

Figure 41 shows a collection of screenshots and pictures of the control plane tools GUIs and deployed hardware. In particular, Figure 41(a) shows the NetP GUI illustrating the overall network including IPoWDM nodes and ROADMs; Figure 41(b) and Figure 41(c) respectively report the ONOS GUI of the packet and optical controllers. Figure 41(d) illustrates the deployed hardware, i.e., the Lumentum ROADM-20 node degree, the Finisar WSS filter and the EdgeCore switch running SONiC operating system.

Experimental Results

Reconfiguration time of central frequency. The performance of 400ZR/ZR+ transceivers has been first measured in terms of time required to reconfigure the central frequency. Results are summarized in Table 12. Specifically, the optical carrier has been moved interacting directly with the REST APIs deployed on SONiC. The frequency shifts have been performed considering both nearby frequencies and distant frequencies in order to check whether the reconfiguration time depends from the frequency gap. The optical configurations have been performed considering two types of pluggable transceivers (i.e., two ZR and two ZR+ modules). For each run, two values of time have been collected, both time intervals start when the frequency change command is issued to the SONiC system; the first time (*prompt* in Table 12) is the interval from the command issue and return the prompt; the second time (*laser* in Table 12) is the interval form the command issue and the signal detection at the RX.

Considering the two ZR modules, performing a frequency reconfiguration with 2 THz gap (passing from 193.0 THz to 195.0 THz) has taken around 10 seconds (10.39 and 9.6 seconds, respectively) at the prompt level, while a longer time interval (around 67.6 seconds) at the laser level. The reverse operation, passing from 195.0 THz to 193.0 THz, has presented similar time for the reconfiguration. Even considering smaller frequency gap (from 193.0 Thz to 193.1 THz with 0.1 THz of gap) or bigger frequency gap (from 196.0 THz to 192.0 THz, with 4 Thz of gap) has not differently impacted the reconfiguration time.

Considering instead the two ZR+ modules, the frequency reconfiguration with 2 THz gap (passing from 193.0 THz to 195.0 THz) has achieved similar performance in terms of prompt results (respectively 9.78 and 9.45 seconds). However, significantly faster performance have been experienced considering the optical transmitted signal achieving configuration time around 15 seconds (i.e., 16.15 and 15.69 seconds, respectively). The reverse operation, passing from 195.0 THz to 193.0 THz, has presented similar time for the reconfiguration, with the prompt issuing time in the order of 10 seconds and a shorter laser activation time (respectively, 15.31 and 13.20 seconds). Even considering small frequency gap (from 193.0 THz to 193.1 THz with 0.1 THz of gap or bigger gap (from 196.0 THz to 192.0 THz, with 4 THz of gap) has not differently impacted the reconfiguration time, with similar values collected.

Further analyzing the results collected using the two classes of pluggable transponders (e.g., the ZR and ZR+ modules), it is clear that ZR+ modules guarantee a faster laser activation time. Moreover, the table shows that the (re)-configuration time interval does not depend on the gap between the starting and stop frequencies. In addition, the collected results show that modules of the same type present similar behavior, showing a stable and uniform configuration time.

Table 12: Tuneability performance of coherent pluggable modules within SONiC IPoWDM box.

Frequency		**ZR Sample 1**		**ZR Sample 2**		**ZR+ Sample 1**		**ZR+ Sample 2**	
Start	Stop	prompt	laser	prompt	laser	prompt	laser	prompt	laser
193.0	195.0	10.39	67.65	9.60	67.63	9.78	16.15	9.45	15.69
195.0	193.0	9.67	67.8	9.46	70.58	9.38	15.31	9.46	13.20
193.0	193.1	9.50	69.43	9.39	69.54	9.45	15.26	9.52	15.70
193.1	196.0	9.78	68.04	9.47	69.54	9.64	15.66	9.51	19.10
196.0	192.0	9.47	71.06	9.79	67.34	9.51	15.90	9.52	15.21

Estimation **of channel bandwidth.** The actual channel width generated by the ZR+ modules is not explicitly mentioned within the pluggables data-sheet. Thus, to estimate the effective bandwidth of the 400 Gbps channel generated by the ZR+ modules using 16-QAM modulation format, two IPoWDM through an optical line composed of three amplified spans of 80 Km. A Finisar WSS has been used to filter the generated signal and apply symmetric narrow-filtering of the channel. More specifically, initially the channel is configured with 100 GHz width, then gradually narrowed (i.e., cutting left and right an incremental portion of the spectrum) while reading the parameters at the receiver side. The parameters are read using the REST APIs deployed on SONiC that interacts with the C-CMIS driver.

Collected results are illustrated in Table 13, operating an optical channel with central frequency 195.5 THz. Each row of the table reports the received power, BER, OSNR, eSNR and the operational status of the interface in SONiC. From the collected results, it is evident that the channel continues to operate also if it is narrow filtered at 60 GHz (e.g., 20 GHz left and 20 GHz right). Therefore, considering a flex-grid scenario, significant bandwidth can be saved by accurately configuring the optical filters along the path with respect to the nominal value of 100 GHz.

Table 13: Transmission performance from CCMIS interface with narrow-filtering.

Central Freq. (THz)	Bandwidth (GHZ)	Received Power (dBm)	BER	OSNR (dB)	ESNR (dB)	Status
195.5	100	-16.31	0.00253	29.9	16.6	UP
195.5	90	-16.31	0.00256	29.8	16.6	UP
195.5	80	-16.36	0.00257	29.7	16.6	UP
195.5	70	-16.42	0.00276	29.5	16.5	UP
195.5	64	-16.54	0.00319	28.9	16.3	UP
195.5	62	-16.66	0.00359	28.5	16.2	UP
195.5	60	-16.72	0.00417	28.0	16.0	UP
195.5	58	-16.88	0.00728	26.6	15.3	DOWN
195.5	56	-17.03	0.01297	25.0	14.4	DOWN

Recovery time with real traffic. The experiment, which was performed on the testbed shown in Figure 37, starts with the connectivity setup and its step reference. In particular, a layer-2 Ethernet link between the IPoWDM nodes interfaces is requested that triggers the activation of a DSR connection between the pluggables using a central frequency of 191.900 THz and a bandwidth of 100 GHz. Pluggables are configured for 400 Gbps using 16-QAM modulation format. This connectivity is used to transport a 10 Gbps traffic flow generated by the Spirent N4U traffic generator/analyzer.

Starting from this scenario, the recovery procedures are started following the three schemes detailed in the failure recovery service Section above. Thus, the failure detection time needed by the OptC is always neglected. However, from the experience matured in other experimental work, one can estimate such time in the order of 100 ms. Thus, with all the recovery procedures, the traffic is disrupted while the re-configuration procedure is on-going, that is measured counting the number of lost packets using the Spirent tester.

Figure 42 presents the obtained results, showing the obtained time distribution over 30 experiment repetitions. Figure 42(a) refers to the optical restoration procedure. In this case, the OptC reconfigures the connection along the longest path keeping the same channel central frequency of 191.900 THz. In this case the traffic disruption has an average value of 2.6 seconds.

Figure 42(b) refers to the hybrid restoration procedure that implies the re-tuning of the central frequency, i.e., traffic is re-routed on the longest path moving the central frequency to 193.100 THz. In this case the traffic disruption has an average value of 102.6 seconds, this result may appear not in accordance with the fact that the experiments related to reconfiguration time of central frequency above, show that, using ZR+ modules, the time for performing frequency reconfiguration is about 15 seconds. However, this time is measured only at the physical layer considering the moment in which optical power is detected on the receiver side. While restoring a real traffic flow may imply longer time due to re-alignment needed after frequency change, thus leading to the measured values of about 100 seconds.

Finally, Figure 42(c) refers to the hybrid protection procedure, in which the OptC forwards the failure notification to the PckC, which dynamically includes in the traffic VLAN a second pair of coherent pluggables that were in hot backup at the IPoWDM nodes. In this case the traffic disruption has an average value of 1.8 seconds. The measurements made show that traffic recovery in multi-layer networks using IPoWDM nodes is still a critical feature.

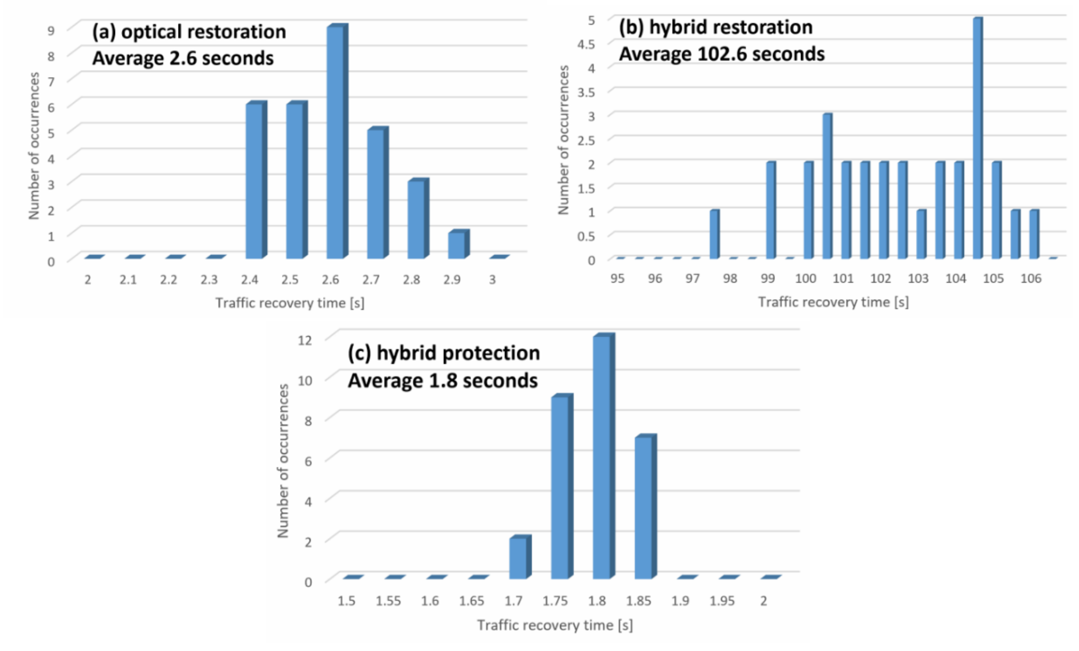


Figure 42: Distribution of traffic recovery time over 30 experiments: (a) optical restoration; (b) hybrid restoration; (c) hybrid protection.

All three solutions do indeed present critical issues. Optical restoration provides an acceptable recovery time, with the strong requirement to use the same frequency on the backup path. This condition is difficult to implement in a real network where, under normal traffic conditions, it may be very unlikely to find the required channel end-to-end available. Hybrid restoration is the most flexible solution that is certainly the best candidate solution in case of soft-failure, but requires unacceptable recovery times in case of hard-failure. Finally, hybrid protection provides acceptable recovery time but requires significant network overprovisioning.

5 Digital Twin for the Optical Time Domain

Monitoring and real-time data analytics are key enablers for the realization of network automation. In particular, AI/ML-based algorithms have been extensively applied to optical communications to enhance their overall performance. Applications include identifying and predicting optical transmission parameters to mitigate different physical layer impairments, including both linear interference, e.g., Amplified spontaneous emission (ASE) noise, and noise caused by the Kerr effect. In fact, one of the most active fields of application in optical networks is for optical performance monitoring and particularly interesting are those ML-based models that combine the characteristics of the physical system and real-time monitoring data to produce accurate estimation of NLI noise. Other approaches receiving large attention are those exploring deep learning (DL) techniques to extract information from complex, dense monitoring data inputs, without knowledge of the physical characteristics. E.g., in optical coherent systems with advanced Digital Signal Processing (DSP) techniques, the analysis of In-Phase and Quadrature (IQ) optical constellation diagrams as images can be performed by means of training convolutional neural networks to estimate the QoT of optical signals.

AI/ML/DL techniques usually require large data sets for training purposes to produce accurate ML models. In addition, precise knowledge of physical input parameters is needed. Examples include the length of the optical connection (*lightpath*) from Tx to Rx and the Tx launch power. Although some physical parameters might vary with time, considering them as inputs of ML models strongly increases the applicability of those models to real scenarios. In this regard, a possible approach is to model individual network elements and concatenate them to create one model for the complete lightpath. Note that model concatenation is a common approach, which is part of other ML techniques, e.g., AE.

In this section, we propose a comprehensive solution for in-operation lightpath analysis of IQ constellations. Specifically, the contribution is two-fold:

- A novel network functional architecture is presented. A sandbox domain is used to obtain DL models suited for the lightpath under analysis. Models include AEs, statistical distributions of IQ constellation points, and DNN-based lightpath metric predictors. The models are designed to run at the Rx site, and continuously analyze lightpath's metrics to compress monitored constellation samples and detect potential anomalies.
- A methodology for constellation analysis based on Gaussian Mixture Models (GMM) (supervised) and AE-based (unsupervised) feature extraction is proposed. Moreover, a lightpath modelling approach consisting in concatenating DNN models emulating the performance of optical components, e.g., ROADMs and optical links including intermediate OAs, is presented and used to generate expected constellations and synthetic constellation samples.

Two different use cases of lightpath analysis using the proposed constellation analysis and lightpath modelling methodology are detailed. Specifically, path length and power analysis using GMM-based and AE-based constellation analysis are proposed.

5.1 AI-Based Constellation Analysis

Figure 43 overviews the considered network architecture and will be used for describing the main workflow; for the sake of simplicity, only the directly involved elements, like a lightpath, a node controller and a sandbox are detailed, whereas other components have been sketched - e.g., the SDN- or omitted to better highlight the key concepts involved in this work. Lightpath *i* is considered as the entity under analysis, which is represented as a sequence of optical components (Tx, ROADMs, links, and Rx) supporting that lightpath. Figure 44 details the internal architecture of the sandbox domain and node agents.

At set-up time, the SDN controller solves the Routing, Spectrum, and Transponder Assignment before configuring the involved devices to establish the lightpath. Then, after the lightpath is provisioned, the sandbox domain receives the lightpath's configuration from the SDN controller (labeled 1 in Figure 43), including its route on the optical network and some metrics. This configuration is used in the sandbox domain to set up an accurate representation of that lightpath to be set in the Rx agent (2 in Figure 43). Such representation is defined as a sequence of pre-trained DL-based models that emulate the behavior of each individual optical component that the optical signal traverses.

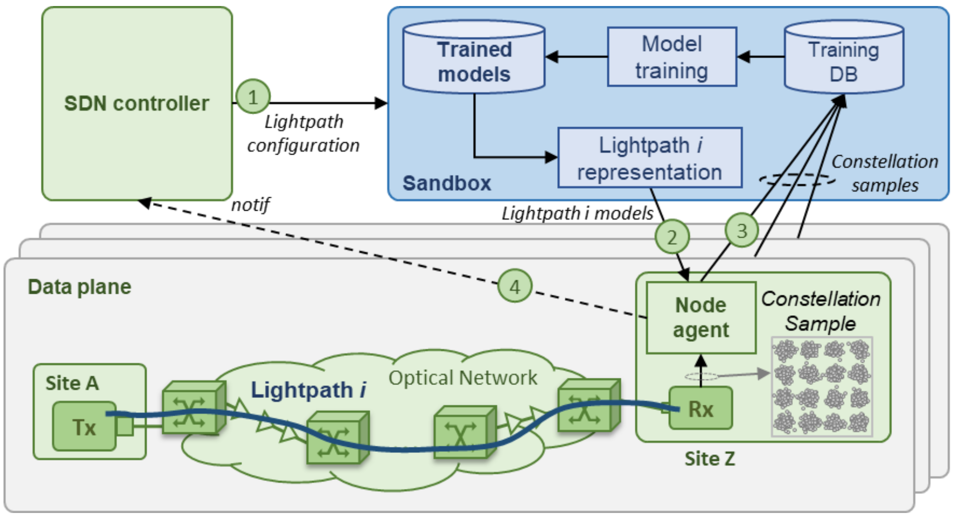


Figure 43. Reference network architecture.

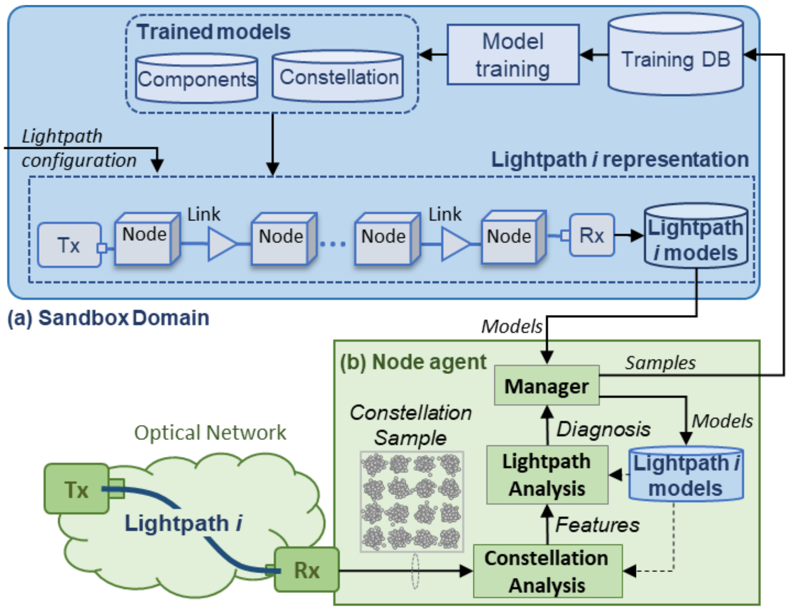


Figure 44. Details of sandbox (a) and node agent (b).

The role of the models is different depending on the physical element they characterize (Figure 44a). For instance, the model for the Tx characterizes the output signal according to its specifications, whereas the models for intermediate elements (ROADMs and fiber links) propagate forward a set of features related to the signal's constellation. Specifically, intermediate components introduce distortion on the constellation as a result of LI and NLI noise. Finally, the Rx model receives the constellation features and performs additional actions before returning the output of the model. Additionally, any relevant change affecting the lightpath during its life-time, e.g., path rerouting, needs to be notified to the sandbox, so as to adapt the lightpath's representation and avoid misleading diagnosis due to mismatch between the physical lightpath and its models.

Both constellation and lightpath analysis require from models that characterize the monitored lightpath. Thus, anomaly detection based on comparing observed features and expected ones coming from lightpath's models can be carried out at the Rx. Note that this scheme highly reduces the amount of data to be sent to the centralized elements (3 in Figure 43), as well as its computational demand for real-time data analysis purposes. Once the lightpath is set-up and the lightpath models set in the Rx agent, they are used for analysis. With a predefined frequency, e.g., every 1s, the Rx samples the received constellation and gathers *n* IQ symbols. The sample is then processed by the *constellation analysis* block in the Rx agent (Figure 44b). The aim of this block is to extract a set of relevant constellation *features* that facilitates posterior analysis, as well as compressing constellation data to be used for multiple purposes, such as model training. These features are obtained by means of both supervised and unsupervised statistics and ML-based techniques (details are given in following subsections). Next, the *lightpath analysis* block processes the features extracted from the received constellation and analyzes key lightpath's configuration metrics, such as length and/or power configuration. The result of this analysis produces a diagnostic report highlighting, e.g., whether some of the metrics does not follow the expected behavior. The diagnostic report is processed by the *manager* block that implements a set of rules and generates notifications to the SDN controller depending on the diagnosis (4 in Figure 43).

Following the above generic architecture, Figure 45 illustrates two different use cases for lightpath analysis. The first use case is devoted to checking whether the real length of a given lightpath matches with the expected one (Figure 45a). This analysis is based on the fact that both LI and NLI noise increase with path length and additionally, both affect the magnitude and shape of the dispersion of the symbols around the expected constellation points. Therefore, differences can be found by comparing the features of the observed constellation points, extracted with supervised techniques, and the expected ones. A model is used to detect any significant difference, as well as to estimate the real length.

As an example, the received constellation of a 16QAM signal is represented in Figure 45a, where a constellation point (3+3i) is zoomed in. External constellation points get more affected by the NLI noise since not only their shape becomes more dispersed around the central point as it happens in presence of LI noise, but they also become more elliptical, with eccentricity and direction of the axes that depend on the traveled distance. Let us assume that the expected features characterize a distribution that is larger and more elliptical than the one observed. This will be detected by the length analysis module as an anomaly in path length, specifically as a length shorter than expected. The reasons behind that anomaly can be multiple, e.g., inaccurate lightpath configuration, wrong sandbox domain model configuration, lack of synchronization between SDN controller and sandbox after a path rerouting, just to mention a few. Upon the notification of the length analysis module, the SDN controller can trigger the needed procedure to detect and isolate the actual reason for the detected anomaly.

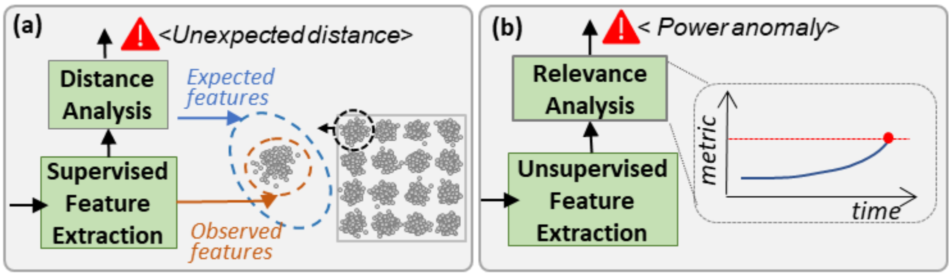


Figure 45. Lightpath analysis use cases.

The second example explores a different approach to detect anomalies affecting the launch power at the Tx side (Figure 45b). Instead of characterizing the gathered constellation through features with the distribution of the constellation points, this use case leverages AEs to compress constellation samples into a reduced set of latent space features in an unsupervised way. The analysis is performed in both forward and backward directions through the AEs to quantify relevance metrics at both the latent feature space and the input. This relevance can be tracked over time so as to detect any drift or shift directly related to a power anomaly (e.g., power drop) in the Tx. Note that an early detection of power anomaly can prevent degradations.

5.2 CONSTELLATION ANALYSIS AND MODELLING

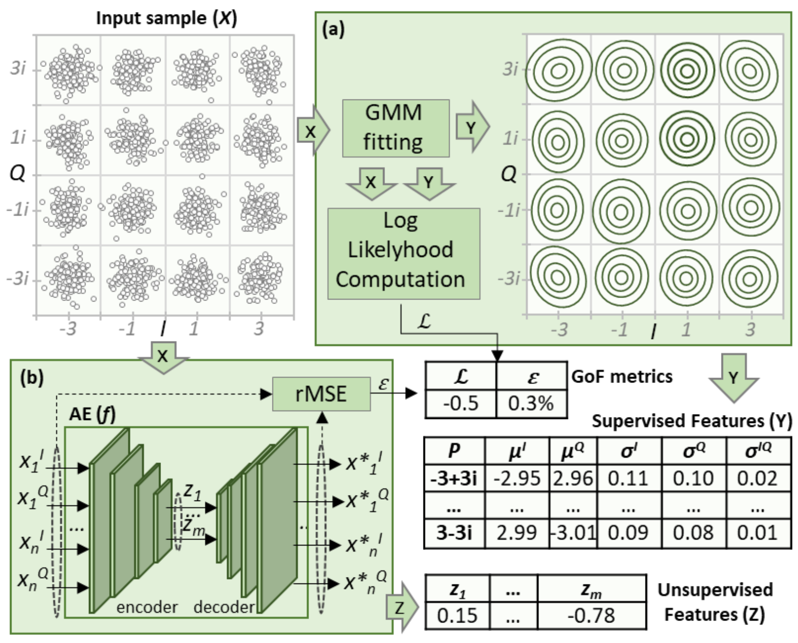
In this section, we present the main procedures for constellation analysis and lightpath modelling. In the following, we are going to consistently denote $X=\{x_1,\ldots, x_n\}$ as an optical constellation sample consisting of n IQ symbols. Although each symbol $x_i \in X$ is typically represented as a complex number, for the sake of simplicity, it will be alternatively denoted as a tuple of real values $<x_i^I, x_i^Q>$ with the in-phase x_i^I and quadrature x_i^Q components, respectively. Further, a constellation is defined by a set of constellation points P, each identified by its expected centroid $<p_j^I, p_j^Q>$.

Supervised and Unsupervised Feature Extraction

Figure 46 illustrates the feature extraction procedure for a given 16QAM optical constellation sample X. The procedure is used to summarize the optical constellation into a number of supervised and unsupervised features; it also produces goodness-of-fit (*GoF)* metrics that allow further evaluation of the quality and usefulness of the generated features.

Let us first focus on the supervised feature extraction approach (Figure 46a). The objective of this approach is to generate the set of features (also referred as processed sample) Y that summarizes X with a number of clear, unequivocal, and predefined characteristics. To this aim, we model the constellation points as bivariate Gaussian distributions. This approach characterizes every constellation point $p\in P$ with a two-component vector $<\mu^I_p, \mu^Q_p>$ representing the mean position in the constellation and with a three-component vector $<\sigma^I_p, \sigma^Q_p, \sigma^{IQ}_p>$, which captures the variance and symmetric covariance terms that the symbols belonging to the constellation point p experienced around the expected mean.

Aiming at allowing an accurate fitting of each of the constellation points, especially when LI and NLI noise are large and symbols are dispersed far from the expected centroid, we apply GMM fitting for multiple and joint bi-variate Gaussian distribution estimation. GMM is initialized to find $|P|$ different bi-variate Gaussian distributions whose expected centroids are the ones in P. An illustrative example is depicted in Figure 46a, where the inset values in the table are the features Y computed from the sample and the level curves depict the bi-variate Gaussian distributions.



$\mathcal{L}$	ε
-0.5	0.3%

P	μ^I	μ^Q	σ^I	σ^Q	σ^{IQ}
-3+3i	-2.95	2.96	0.11	0.10	0.02
…	…	…	…	…	…
3-3i	2.99	-3.01	0.09	0.08	0.01

z_1	…	z_m
0.15	…	-0.78

Figure 46. Feature extraction. Supervised (a) and unsupervised (b).

It is worth noting that by forcing the constellation points to be modelled as Gaussian distributions and by selecting the expected centroids in *P* as initial points for GMM fitting, the obtained features are strongly conditioned. In order to estimate how accurate the GMM fitting is, i.e., how well they characterize the symbols dispersion around the expected mean, the lowest (worst) likelihood value $\mathcal{L}$ (in logarithmic scale) for one constellation point is returned as GoF metric. The $\mathcal{L}$ metric might have different potential applications depending on the specific use case.

In contrast, unsupervised feature extraction (Figure 46b) aims at transforming input sample *X* into a latent sample *Z* that accurately represents the main characteristics of *X* without actually defining how to achieve such characterization (e.g., without imposing any statistical distribution). In this case, we use an AE with 2·*n* inputs for the *I* and *Q* components of every symbol in *X*, followed by a number of hidden layers, each with a number of hidden neurons. The last layer of the encoder component contains *m* outputs and is commonly known as latent feature space *Z*. The latent space is the input of the decoder component, which also contains a structure of hidden layers (not necessarily equal to those of the encoder) that lead to a final output layer with 2·*n* values, each corresponding to one of the initial encoder inputs. The AE is trained using the mean absolute error as loss function (typical for regression applications) so that the error between a given encoder input neuron and its related decoder output is minimized. In this way, the encoder component codes input sample *X* into latent sample *Z*, whereas the decoder reconstructs sample *Z* into the original feature space. Note that the reconstructed sample *X** differs from the original one *X*; such a difference, that can be quantified in terms of relative mean square error (rMSE), is used as GoF metric (denoted ε) for unsupervised feature extraction.

DNN-based Concatenation Modelling

Figure 47 illustrates the proposed approach to build DL-based lightpath models as a concatenation of DNN-based component models. As illustrated in the example in Figure 47a, the signal crosses three ROADMs (A, B, C) and optical links with different length and number of spans. ROADMs are modelled with two WSS, and every intermediate ROADM, except the last one before the Rx (drop), includes a booster OA that compensates for WSSs insertion losses.

Typically, the insertion losses in the last ROADM are compensated by DSP techniques at the digital coherent Rx. The optical links consist of fiber spans and inline OAs that compensate for the losses of the fiber spans. We assume that the pre-OA at ROADM's input is a part of the link model (see the insets in Figure 47a).

The concatenation model abstracting the lightpath in Figure 47a is presented in Figure 47b. In this case, the ROADMs and optical links in Figure 47a are modeled using DNNs. The lightpath is modeled as an ordered sequence of: *i*) a Tx model, *ii*) an add ROADM model representing ROADM A, *iii*) a 240-km link model representing fiber link A-B, *iv*) a transit ROADM model representing ROADM B, *v*) a 450-km link model representing fiber link B-C, and finally *vi*) a drop ROADM model representing ROADM C.

The Tx model includes a pseudo-random bit sequence (PRBS) generator used to generate the initial optical constellation following a Tx configuration. Such initial constellation can be generated using analytical equations, simulation, ML models, etc. Once the initial optical constellation is generated, a feature extraction block computes the supervised features Y as described in the previous section.

Models for both ROADM and link components follow a similar architecture. Those models propagate feature set Y, modifying the mean and variance of each constellation point according to the LI and NLI noise that the physical element introduces. Aiming at reducing the complexity of the DNN models, a subset of relevant constellation points is selected as representative of the impact of noise during propagation, whereas the rest are generated as a function of the propagated points. Hence, we need to first select the reduced set of constellation points $P^* \subset P$. In particular, all the features in Y that belong to constellation point subset P^*, denoted as $Y(P^*)$, are selected. Then, this reduced set of features is propagated through a DNN model specifically trained for the component that is represented. The structure of the DNN consists in $5 \cdot |P^*|$ input and output features (i.e., μ and σ vectors of the selected constellation points), and a number of hidden layers with variable number of hidden neurons each. Since the final outcome of the model must include the whole set of features, a linear regression model mapping the characteristics of the non-selected constellation points, denoted as $Y(P \backslash P^*)$, as a function of selected ones is used. This model is generic and can be shared between components of different types; it is defined by a matrix of linear coefficients β of size $5 \cdot |P \backslash P^*| \times 5 \cdot |P^*|$ and an intercept vector β_0 of length $5 \cdot |P \backslash P^*|$.

The proposed modelling approach can be also used as a lightweight optical system simulator. By generating random samples following the bi-variate Gaussian distributions defined by Y, synthetic constellation samples can be obtained. In Figure 47b, such random sampling is performed to generate a synthetic optical constellation at the Rx side; however, random sampling can also be applied after any component model, thus generating intermediate constellations. It is worth noting that the time to generate the resulting optical constellation samples (at the Rx and intermediate points) is noticeably short, since it entails propagating values through a set of DNNs, i.e., only a very limited number of simple calculations is required.

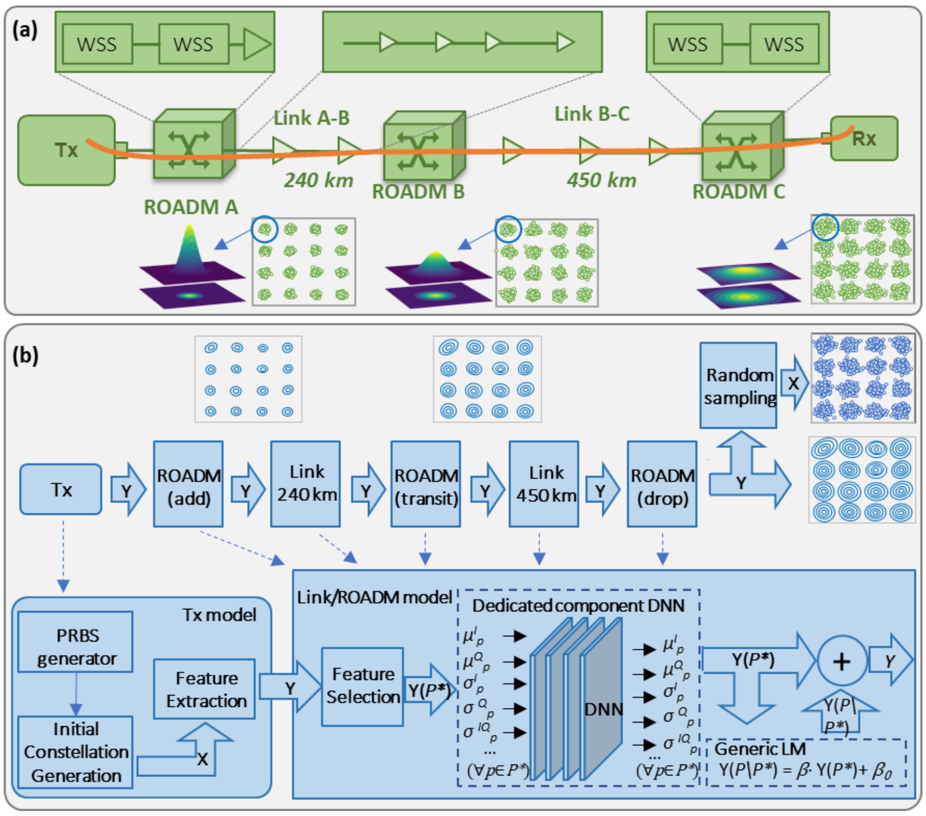


Figure 47. Lightpath example (a) and its proposed model (b).

5.3 Lightpath Analysis Use cases

In this section, we detail the main algorithms to perform the two use cases of lightpath analysis sketched in Section 5.1. The proposed algorithms use the feature extraction and lightpath modelling procedures detailed in Section 5.2. Firstly, the algorithm to set up the lightpath models is presented.

Lightpath Models Setup/Update

Upon lightpath provisioning or the modification of the attributes of an already in-operation one, e.g., because of rerouting, the models for lightpath analysis need to be loaded in the Rx agent (labeled 2 in Figure 43). Algorithm 8 details such procedure, which runs in the sandbox domain and is triggered every time a notification is received from the SDN controller with the details of the established/modified lightpath (1 in Figure 43). The algorithm receives as inputs: *i*) the description of lightpath *R*, including its id and the sequence of nodes and links that composes the lightpath from Tx to Rx, each with its own configuration attributes; *ii*) the connection to the database of trained models (*DB*); and *iii*) a set of configuration parameters.

The output of Algorithm 8 is the set of models and parameters needed for lightpath analysis at the Rx site. These output set includes: *i*) parameter $\mathcal{L}_{thr}$ with the minimum threshold to consider constellations points as accurate Gaussian distributions, *ii*) the statistically-based constellation analysis (*sca*) model used for comparing monitored constellations with the expected one in terms of supervised features *Y*; *iii*) an AE-based (*ae*) model used for compressing and analyzing constellations using unsupervised features *Z*; and *iv*) the path metric estimation (*pme*) model used to predict the path length as a function of supervised features.

Algorithm 8. Lightpath models setup/update

```
INPUT: R, DB, params      OUTPUT: < sca, ae, pme, $\mathcal{L}_{thr}$>
1:  L ← ∅; sca ← ∅; Yhist← ∅
2:  for r∈R do
3:     l ← get(DB['component'], r.attributes)
4:     L ← append(L, l)
5:  add(DB['lightpaths'][R.id], L)
6:  for i = 1..params.nrep do
7:     for l∈L do
8:        if l.type == "tx" then
9:           b ← l.PBRS(params.n)
10:          $X \leftarrow l$.generateConstellation($b$)
11:          $Y \leftarrow$ GMMfitting($X$)
12:       else Y ←l.propagate(Y)
13:    $Y_{hist} \leftarrow Y_{hist} \cup Y$
14: for <$Y_i$, $Y_j$>∈ $Y_{hist}$ do
15:    val←computeChi2($Y_i$,$Y_j$) (eq. (7))
16:    if sca == ∅ or sca.thr<val then
17:       sca.ref ← <$Y_i$, $Y_j$>
18:       sca.thr ← val
19: ae ← get(DB['AE'], R.tx.power)
20: pme ← get(DB['predictor], R.tx.power)
21: $\mathcal{L}_{thr}$ ← get(DB['GoF'], R.tx.power)
22: return < sca, ae, pme, $\mathcal{L}_{thr}$>
```

After some initializations, the lightpath model *L* is built by selecting, for each of the components in *R*, the available component model in *DB* that better fits component's attributes (lines 1-4 of Algorithm 8). After saving model *L* for further purposes (line 5), a number of random constellation samples are generated, propagated through *L* (see Figure 47b) and saved in a temporary data set Y_{hist} (lines 6-13). Y_{hist} is processed in order to build the *sca* model. To this aim, the difference between the features of every pair <Y_i, Y_j> ∈ Y_{hist} is computed by means of a statistical test based on the chi square test (lines 14-15). The proposed chi2-based statistic (*chi2*) is formally defined as:

$$chi2(Y_i, Y_j) = \sum_{k=1..|Y_i|} \frac{(Y_i(k) - Y_j(k))^2}{min(Y_i(k), Y_j(k))} \tag{7}$$

In consequence, the *sca* model includes the pair of reference samples <Y_i, Y_j> that maximizes the value of *chi2*, as well as such maximum *chi2* value that will be later used as threshold for acceptable difference between samples (lines 16-18). Finally, *ae* and *pme* models and parameter $\mathcal{L}_{thr}$, are retrieved from the set of trained models and eventually returned jointly with *sca* (lines 19-22). Although *pme* can be designed in multiple ways, we consider it as a DNN model that predicts the lightpath length as a function of features *Y*.

Lightpath Length Analysis

Once the sandbox manager feeds the Rx agent with updated models, in-operation constellation analysis can be carried out.

Algorithm 9 details the procedure used to detect mismatch between the received and the expected constellation in terms of the supervised features *Y*. Therefore, it requires to process the monitored constellation *X* jointly with models *sca* and *pme*, and parameter $\mathcal{L}_{thr}$. After some initializations (line 1 in Algorithm 9), the supervised features are computed and the logarithm of the likelihood is compared against $\mathcal{L}_{thr}$ to detect whether observed features are unlike to follow a Gaussian distribution (lines 2-4). If so, a new message with the obtained likelihood is added to the diagnosis report (line 5).

Algorithm 9. [Rx] - Length Analysis

```
INPUT: $X$, $sca$, $pme$, $\mathcal{L}_{thr}$; OUTPUT: $Y$, $diagnosis$
1:  $diagnosis \leftarrow \emptyset$
2:  $Y \leftarrow$ GMMfitting($X$)
3:  $\mathcal{L} \leftarrow$ logLikelihood($X$, $Y$)
4:  if $\mathcal{L} > \mathcal{L}_{thr}$ then
5:      $diagnosis$.add(<'Inaccurate Features', $\mathcal{L}$>)
6:  $test_i \leftarrow$computeChi2($sca$.Y$_i$, $Y$) (eq. (7))
7:  $test_j \leftarrow$computeChi2($sca$.Yj, $Y$) (eq. (7))
8:  if min($test_i$, $test_j$) < $sca$.thr then
9:      $diagnosis$.add(<'Unexpected Length, $test$>)
10:     $diagnosis$.add(<'Estimated Length, $pme(Y)$>)
11: return $Y$, $diagnosis$
```

The analysis continues by computing the *chi2* test between the observed features and the two reference samples stored in *sca*. The minimum of both values is compared against the threshold (lines 6-8), and, in case of threshold violation, an unexpected lightpath length is detected and *pme* is used to provide an estimation of the real length of the lightpath, which adds relevant information to the diagnosis report (lines 9-10). The procedure finishes by returning the diagnosis report, as well as the supervised feature sample *Y*, which can be eventually stored for further analysis (line 11).

Power Anomaly Detection

Finally, Algorithm 10 performs the analysis of the monitored constellation using the *ae* model. This analysis computes the relevance (importance) *h* of the *ae* model inputs and keeps track of them in time to detect any strong variation, like drift or shift. To that end, besides the monitored sample *X* and *ae* model, the algorithm receives the set of historical relevance measurements *H* of such a lightpath. First of all, forward analysis is performed (lines 1-6 in Algorithm 10). Specifically, the original sample *X* is transformed into the latent sample *Z* using the encoder and reconstructed into *X** using the decoder. Then, the rMSE between original and reconstructed sample is compared against the maximum error computed during *ae* training to detect whether latent features *Z* are inaccurate. Similarly as for the previous use case, inaccurate features are diagnosed if reconstruction error is high.

Algorithm 10. [Rx] - Importance analysis

```
INPUT: $X$, $H$, $ae$; OUTPUT: $Z$, $H$, $diagnosis$
1:  $diagnosis \leftarrow \emptyset$
2:  $Z \leftarrow ae$.encoder.propagate($X$)
3:  $X^* \leftarrow ae$.decoder.propagate(Z)
4:  $\varepsilon \leftarrow$rMSE($X$,$X^*$)
5:  if $\varepsilon > ae.\varepsilon_{max}$ then
6:      $diagnosis$.add(< 'Inaccurate Features', $\varepsilon$>)
7:  $h \leftarrow$relevanceBackpropagation($ae$, $X^*$)
8:  $H \leftarrow$update($H$,$h$)
9:  if detectVariation($H$) then
10:     $diagnosis$.add(< 'Power anomaly', $h$>)
11: return $Z$, $H$, $diagnosis$
```

Regardless of the result of forward diagnosis, analysis continues by applying relevance backpropagation techniques, using the *ae* model backwards (from *X** to *X*) in order to compute the relevance of every input. By averaging the relevance of the inputs of the same constellation points, the relevance analysis vector *h* is computed, with one value for each constellation point.

Once the relevance vector *h* is computed, historical set *H* is updated (line 8). Note that *H* consists in |*P*| time series with the temporal evolution of the relevance of each constellation point. Then, a procedure to detect variations in time series can be applied to detect variations such as gradual drift and instantaneous shift. This variation analysis can either be applied to each of the time series independently or the aggregation (sum) of time series belonging to a group of constellation points, e.g., outer or inner constellation points. If, regardless of the type, some variation is detected, a power anomaly message is generated and added to the report jointly with the current measured relevance (lines 9-10). Finally, the diagnosis report and the historical relevance set *H* are returned, jointly with the latent sample *Z* (line 11).

5.4 ILLUSTRATIVE NUMERICAL RESULTS

In this section, we first introduce the simulation scenario and constellation data generated for numerical evaluation purposes. Then, the feature extraction procedure is evaluated and next, the DNN-based concatenated model for lightpath modelling, as well as the setup algorithm are validated. Finally, the lightpath analysis use cases are evaluated.

Simulation Scenario and Data Sets

To evaluate the proposed methods for constellation and lightpath analysis, a MATLAB-based simulator of a coherent WDM system was developed to generate IQ constellations for a 16QAM@64GBd signal under different physical path characteristics. Assuming 100 GHz channel spacing and full spectrum occupancy, signal samples containing 2,048 symbols and shaped by a root-raised cosine filter with a 0.06 roll-off-factor are generated at the Tx side. Then, the signal is propagated through standard single mode fiber spans, characterized by optimal power of -1 dBm, attenuation factor of 0.21 dB/km, dispersion parameter of 16.8 ps/nm/km, and nonlinear parameter of 1.14 1/W/km. Spans are modeled by solving the nonlinear Schrödinger equation using the well-known split-step Fourier method, whereas ideal inline optical amplification is modeled as EDFAs with a noise figure of 4.5 dB, introducing linear noise. Finally, a DSP block is considered at the Rx able to perform ideal chromatic dispersion compensation and phase recovery.

Figure 48 shows the two different scenarios configured for constellation data generation. Under the *single link* scenario (Figure 48a), a sequence of spans between Tx and Rx without intermediate ROADMs is configured. The first span has a variable length ranging from 40km to 80km and places an optical attenuator after the Tx to adjust the signal power according to the length of the first span, whereas the remaining spans have a fixed length of 80km. We considered up to 25 spans, so 3,000 constellation samples with a total length ranging from 80km to 2,000km were generated. Moreover, each sample belongs to one of the following configurations for the first span length and initial attenuation: *i*) *optimal* (80 km, 0 dB); *ii*) *sub-optimal* (60 km, -4 dB), *iii*) *degradation* (40 km, -8 dB). These different configurations have been devised to introduce power variations that result in small changes in the optical constellations without impacting lightpaths' QoT. In contrast, intermediate ROADMs between Tx and Rx are considered under the *multiple link* scenario (Figure 48b). The WSSs inside the ROADMs are based on commercially available ones and modeled. In this case, four different optical link configurations in terms of total length and number of spans are considered: 100-km (2x50-km spans), 240-km (4x60-km spans), 400-km (5x80-km spans), and 560-km (7x80-km spans). By generating lightpaths with hop length (number of links in the lightpath's route) in [1, 4] and combining links with different configurations, 3,000 signals with a total length between 100km and 2,240km were generated. In this work, we use a typical data split of 60%-20%-20% for training, testing, and validation purposes, respectively.

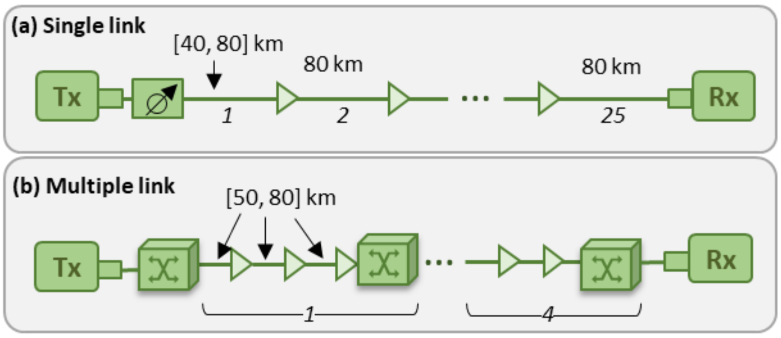


Figure 48. Single link (a) and multiple link (b) lightpath scenarios.

The network architecture in Figure 43 has been reproduced in a Python-based simulator implementing its main elements and functional blocks. For model training in the sandbox domain, we used *sklearn* and *keras* as main libraries for training and testing DNN models. The configuration of each DNN model and the selected data set for training, testing, and validation is specified in the following subsections.

Feature Extraction Evaluation

Let us first focus on evaluating the supervised feature extraction methodology based on GMM fitting. For this study, we used the data generated under the single link scenario with the optimal power configuration. Figure 49 shows the evolution of the supervised features as a function of the total lightpath length. For the sake of simplicity, only one outer (-3+3i) and one inner (-1-1i) constellation points are selected. We observe that the average position of constellation points (Figure 49a), which has been normalized to the expected centroid, slightly varies with length. Nonetheless, the selected outer constellation point shows some remarkable drift in the Q axis. Regarding variance terms (Figure 49b), it is clear that they are strongly correlated with length in the whole range, whereas the covariance term (Figure 49c) has a significant shift for long path lengths. We observe in the figures that clear and strong patterns between the supervised features and the length of the lightpath exist, which anticipates good accuracy of the length analysis procedures based on these supervised features.

In addition to the previous results, Figure 49d shows the likelihood GoF metric $\mathcal{L}$, which stays above -3.5 for all the considered distances. Then, such value can be selected as $\mathcal{L}_{thr}$ parameter for validation of feature extraction procedures. Moreover, the Henze-Zirkler multivariate normality test was conducted for all the samples belonging to the selected dataset. We concluded that all 16 constellation points can be accurately modeled as Gaussian distributions for all the considered distances, since the obtained p-value of the test always exceeded the commonly accepted significance level of 0.05. To better illustrate the valid fitting of constellation points as bi-variate Gaussian distributions, Figure 50 zooms in the selected inner and outer constellation points of two samples belonging to different lightpath lengths (400km and 1600km). The computed Gaussian distribution is plotted together with the samples, showing different level curves for different variance values. In view of the results, we can validate the proposed supervised feature extraction procedure for the characterization of constellation points.

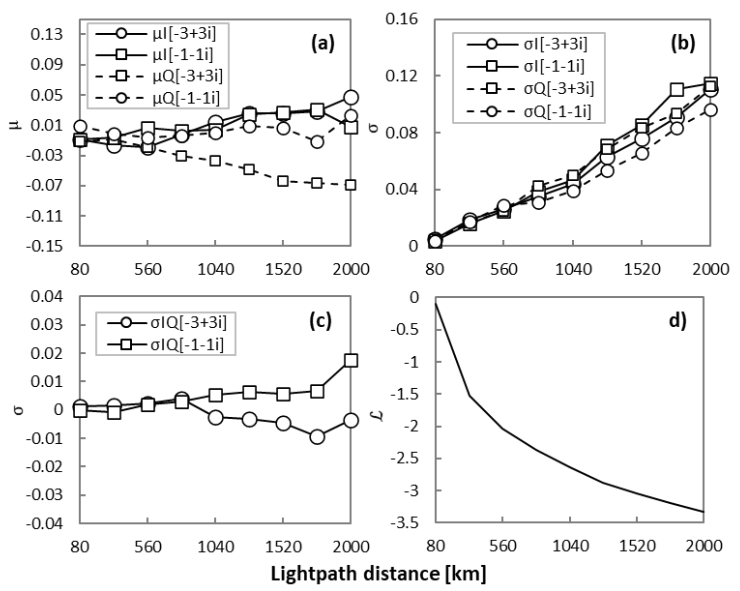


Figure 49. Supervised feature extraction performance.

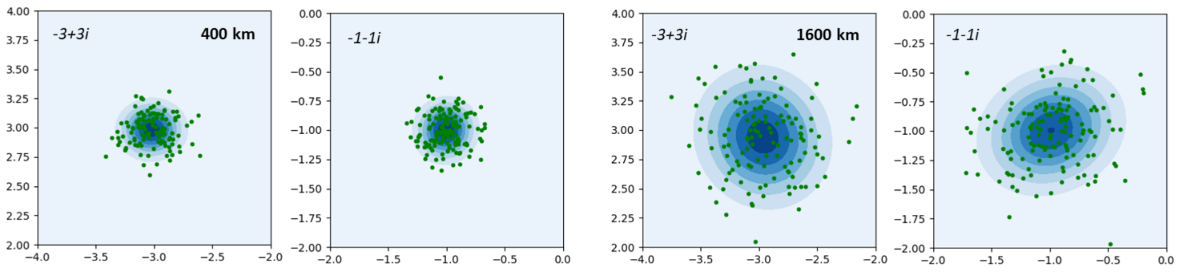


Figure 50. Example of supervised feature extraction for two constellation points after 400km and 1,600 Km.

Let us now numerically evaluate the performance of the unsupervised feature extraction based on the AE model, which is part of the forward analysis. To this aim, we initially trained an AE model with data from the single link scenario and optimal power configurations. Fixing a symmetric encoder and decoder configuration, each with 4 hidden layers (1024, 256, 128, and 64 ReLU neurons), we trained different AEs for a size of latent space Z (*m*) ranging from 4 to 64 features. The results in terms of reconstruction error *ε* for the testing samples are presented in Figure 51a. For benchmarking purposes, Principal Component Analysis (PCA) was conducted, where the training data set was used to compute the first *m* PCs that collect the maximum information from the original data. Testing data samples were next compressed and reconstructed with the selected PCs, thus emulating the encoding/decoding AE network components. Supported by the results, we can conclude that 32 latent features are enough to reach a negligible (< 2%) reconstruction error. Note that the same number of PCs doubles the error of that of the AEs.

Figure 51b plots the compression rate achieved as a function of the target reconstruction fidelity (defined as 1- *ε*). We observe that the AE clearly outperforms PCA. Note that 99% of reconstruction fidelity can be achieved with compressed samples, while reducing in 96% the size of original constellation samples. Such an extremely large compression rate is not achieved by PCA, which reaches a moderated 60% of compression rate for the same reconstruction fidelity.

Finally, Figure 51c plots the relation between unsupervised features and path length, similarly as for supervised features. For representation purposes, the 32 latent features have been projected into two dimensions by means of applying PCA to the latent space samples. The graph shows the position of the samples in the two-dimensional space obtained with PCA, whereas path length is coded by a color scale. We observe a clear relation between latent features and path length. However, this relation is not as strong as that of the supervised features, which validates the latter for path length analysis.

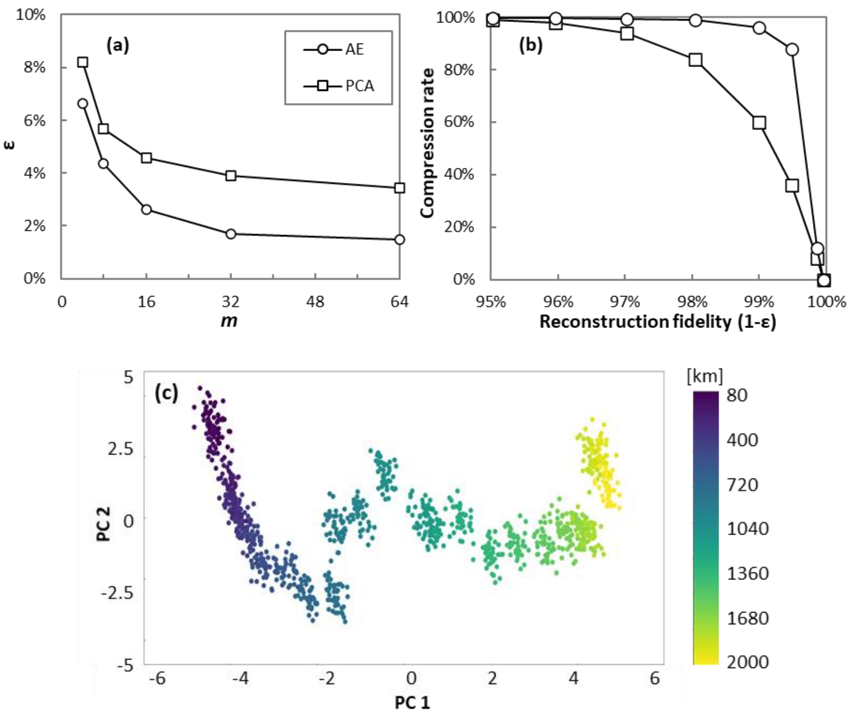


Figure 51. Unsupervised feature extraction performance.

Lightpath Modelling

For lightpath modelling, we selected the number of constellation points to the minimum providing just enough information to capture the overall constellation characteristics; for 16QAM, specifically, two outer (-3+3i, 1-3i) and two inner (1+1i and -1-1i) constellation points were selected. In addition, we considered that both DNN models for the optical fiber links and for ROADMs follow the same architecture characterized by: *i*) 20 input neurons (5 features per constellation point); *ii*) two hidden layers, each one with 12 neurons and *tanh* activation function; and *iii*) one output layer with 24 neurons to estimate the output features. These component models were trained during 5,000 epochs and tested with data from the multiple link data set.

The overall absolute and relative errors for all link configurations, lightpath lengths, and selected constellation points are shown in Figure 52, where average and maximum errors of features μ (Figure 52a) and σ (Figure 52b) are plotted. We observe negligible μ prediction errors (max error < 2%) independently of the link length. In contrast, σ max error is around 30% for low σ values although decreases when path length increases, becoming under 15%, which is, in general, a good enough performance to validate the models. For illustrative purposes, Figure 52c plots the Gaussian distributions for the selected constellation points, obtained with the concatenation model and simulator for a 1,600-km lightpath (4 optical links of 400km). For the sake of clarity, we reduced the number of level curves and removed colors. It is worth noting that strong similarities between both cases are evident.

The reconstruction of the features of the non-selected constellation points can be carried out by means of the proposed linear model with a reconstruction accuracy of 97%, which indicates the proper choice of the selected points.

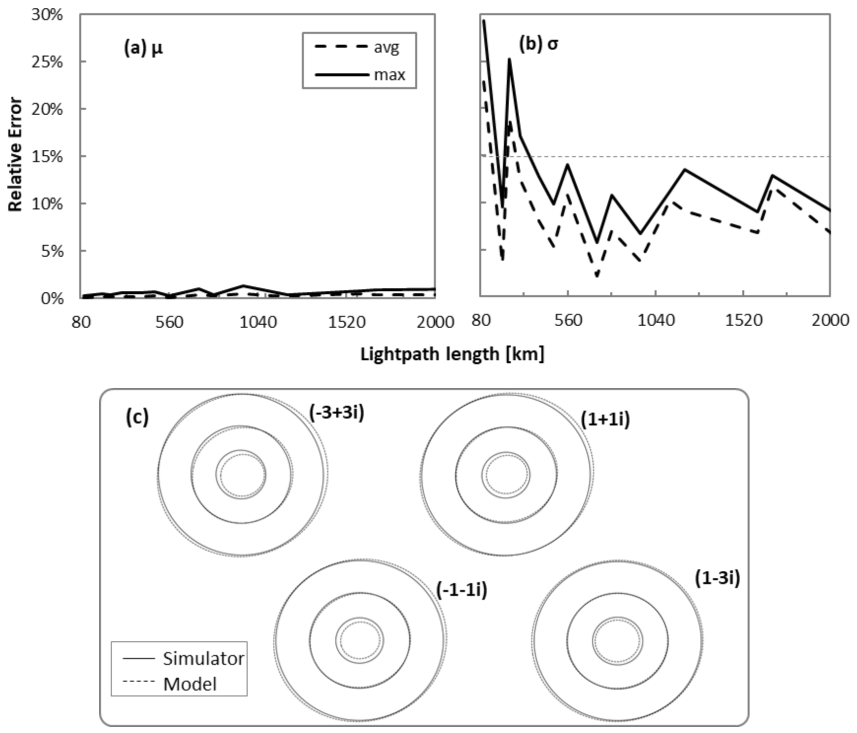


Figure 52. Lightpath modelling performance.

Length Analysis Use Case

Let us now analyze the performance of Algorithm 9. Specifically, we focus on the performance of models *sca* and *pme*. To this aim, the features of the reference samples stored in *sca* model (*model-based*) are compared to those extracted from the validation samples of the multiple link data set (*simulator-based*). The comparison between model-based and simulator-based features was performed using the proposed *chi2* test. We first compared the case when the simulation and model were configured with the same lightpath length and link configuration. For all the combinations, the maximum observed threshold for the *chi2* test never exceeded 0.5 in logarithm scale. Therefore, we use such a threshold for unexpected length detection.

Next, we compared different configurations of 4-link hop lightpaths to check whether the value of the *chi2* test serves as a good indicator of misleading length. Figure 53a reports the results, where we observe that the selected threshold of 0.5 allows us to clearly distinguish all cases when simulation and model were configured differently (orange) from cases with the same length (green). Additionally, the impact of considering just slightly different scenarios in the simulation and concatenation model was tested. Specifically, a 4-link hop lightpath with 240-km links was configured in the simulation, whereas the model was configured with the same number of hops and link configuration except for only one of the hops, where a 400-km link was selected. The four different positions in the path for the 400-km link were evaluated; Figure 53b shows that all cases stayed above the 0.5 threshold, which implies that the small difference was correctly detected. Note that localization of the longer link can be done by performing the test in the intermediate links. Finally, Figure 53c shows the result of applying the intermediate analysis when the 400 km is in the third fiber span. We observe that the link is localized as the *chi2* test value exceeds the selected threshold when evaluating the features right after the third fiber span.

The performance of *pme* was eventually evaluated. The structure of *pme* DNN was as follows: *i*) 20 input neurons (5 features per selected constellation point); *ii*) two hidden layers, each one with 12 neurons and *tanh* activation function; and *iii*) one output layer with one single neuron that predicts lightpath length. The *pme* DNN model was trained with the single link data set and during 5000 epochs. Figure 54 shows the average and maximum relative estimation error. Here, we observe average error below 5% regardless of lightpath length and maximum error under 10% for lightpaths longer than 100km. These results validate *pme* as accurate real lightpath length estimation.

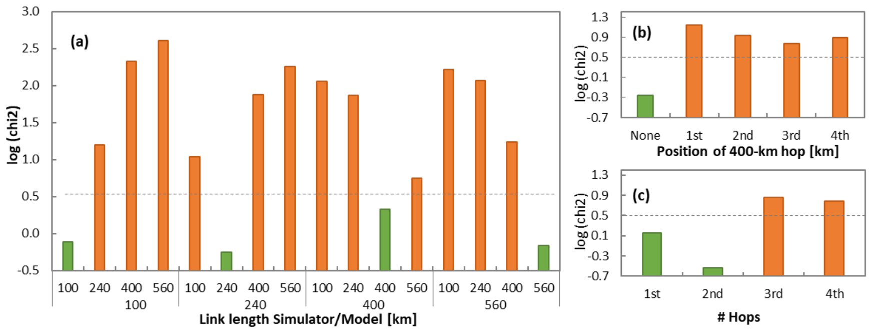


Figure 53. sca model performance.

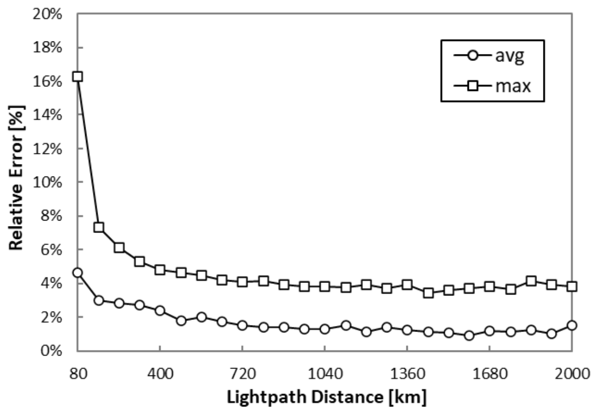


Figure 54. pme model performance.

Relevance Analysis

Finally, we conducted a numerical study to evaluate the models involved in Algorithm 10. Specifically, we focused on illustrating how the *ae* model can discern between different power scenarios and on showing how the input relevance varies with power degradation.

For power scenario discrimination, let us first inspect the examples of constellation point -3+3i for each scenario reproduced in Figure 55. We observe clear differences on the Gaussian features *Y* among power scenarios. Figure 56 shows each of the samples projected on the reduced two-dimensional PCA space from the *Y* space. We realize that the observed differences do not support an easy discrimination since the three classes are not separable. Nonetheless, as part of the forward analysis, the same projection can be performed using latent space samples *Z* computed with the *ae* model trained with all power configurations. This AE produces a reconstruction error under 2%, similar to the one presented above for unsupervised feature extraction. Figure 57 presents the obtained results where the three power scenarios are clearly distinguishable. In consequence, we conclude that the proposed AE-based model for constellation analysis allows for accurate discrimination of power configurations producing small changes in the received constellation.

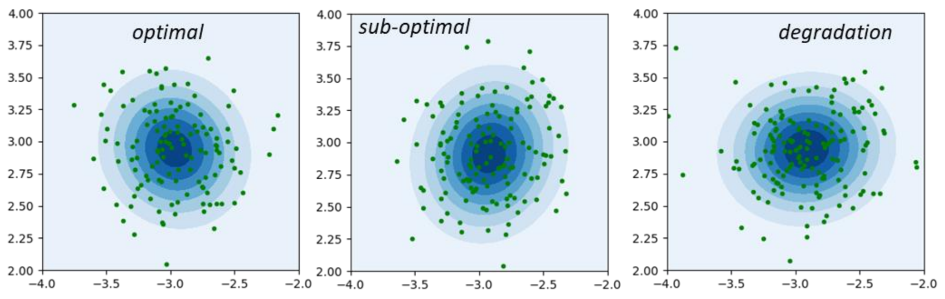


Figure 55. Constellation point -3+3i examples for power scenarios.

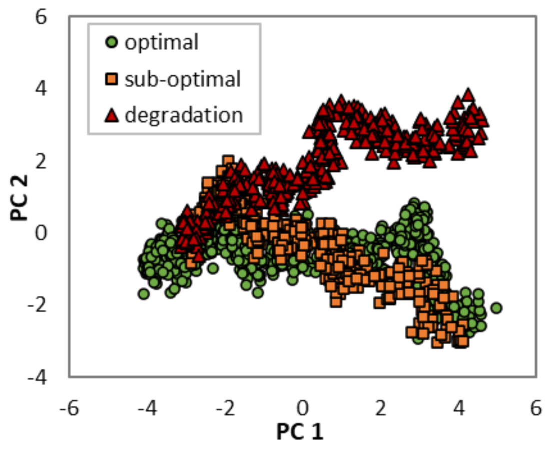


Figure 56. Power scenario discrimination with Y.

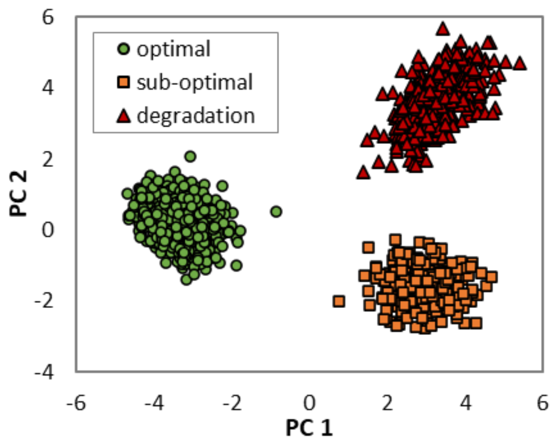


Figure 57. Power scenario discrimination with Z.

Regarding relevance analysis, let us compare two different ways to aggregate relevance of constellation points: *i*) quadrant-based, e.g., right-upper and left-bottom quadrants (Figure 58a); and *ii*) energy-wise, i.e., inner and outer constellation points (Figure 58b). In view of the figures, we conclude that energy-wise aggregation gives more information, since the relevance of inner constellation points clearly reduces when power degrades. Hence, relevance analysis can be potentially used to early power anomaly detection, which could eventually lead to hard failures.

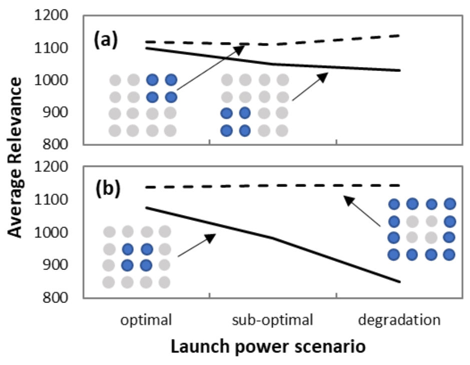


Figure 58. Relevance vs power scenarios.

Table 14. Input relevance variation due to power degradation.

I/Q	-3	-1	1	3
3	7%	-2%	-4%	-9%
1	-4%	-17%	-12%	13%
-1	-3%	-9%	14%	-4%
-3	10%	5%	-3%	22%

The relative relevance variation per constellation point when moving from optimal to suboptimal scenario is detailed in Table 14. Increase/decrease of relevance in more than 5% above/below the reference optimal configuration is highlighted in green/red. In line with the conclusions from Figure 58, we can easily verify that outer constellation points (mainly those in the corners) become more relevant, since more NLI and LI noise is expected under sub-optimal power configuration, which makes the shape of outer points more elliptical than those with lower energy. Hence, the importance of the symbols on these outer constellation points in the latent space is higher, since the overall shape of the constellation is more complex.

5.5 Concluding Remarks

A comprehensive DL-based IQ constellations analysis for in-operation lightpath modelling and power analysis has been proposed. DL models propagating IQ constellations features were trained in a sandbox domain for modelling optical components, such as optical links, OAs, and ROADMs. Then, a target lightpath can be modelled by concatenating specific DL models, to reproduce the propagation of IQ constellations from Tx to Rx. Two methods for feature extraction were proposed, based on GMM (supervised) and on AE (unsupervised). By using those models at the node agent, real-time analysis of the received optical signal can be carried out. In addition, constellation samples were compressed into a reduced set of latent space features, which remarkably reduces (more than 95%) the amount of data that needs to be sent to the centralized controller.

Two illustrative use cases of lightpath performance analysis were investigated. First, lightpath length analysis showed noticeable low error for lightpaths longer than 100km, clearly detecting slight differences in lightpath configurations. Next, different power profiles were studied, where extracted latent features from AE models showed accurate discrimination in terms of Tx power configurations. Finally, a relevance constellation analysis for the AE input parameters was carried out, providing clear understanding about which constellation points have larger relevance, which might be very useful for different analysis proposes.

6 NEAR-REAL-TIME OPERATION

Autonomous network operation evolves from SDN and promises to reduce operational expenditures by implementing closed loops based on data analytics. Such control loops can be put into practice based on policies that specify the action to be taken under some circumstance (policy-based management), e.g., allocate the capacity of a packet flow so that the ratio traffic volume over capacity is under 80%. In packet flows, this ratio is related to the average delay that the packets in the flow will experience because of queuing, and thus to the Service Level Agreement (SLA) in the case that the packet flow is related to some customer connection. Although such policies can be modified, they are purely reactive. Under high traffic variations, they might entail either poor QoS (e.g., high delay or even traffic loss) and SLA breaches or poor resource utilization, which in both cases represent large costs for network operators. Note that policy-based management does not define the desired performance and thus, agents implementing those policies are unable to learn the best actions to be taken. Another approach for network automation is Intent-Based Networking (IBN), which allows the definition of operational objectives that a network entity, e.g., a traffic flow, has to meet without specifying how to meet them. IBN implements and enforces those objectives, often with the help of ML.

In this section, we present solutions for the near-real-time control of network resources, thus relieving the SDN controller from such operations.

6.1 LOW COMPLEXITY OPTICAL POWER OPTIMIZATION

With the deployment of complete telemetry solutions, time series of measurements performed at network elements, such as BER, power, etc., are now customarily transmitted to third entities and jointly processed. Besides the obvious usage for troubleshooting, a lot of effort has been focused on leveraging this data to refine the knowledge of the network physical parameters, and thus to help construct a DT, i.e., a timestamped digital replica of the optical network. Operators can leverage the DT to perform several different optimization operations and extract the highest possible value from their network at a given time. In recent work [An24], we focus on the problem of using monitoring data to a) refine the knowledge, and b) optimize in a closed loop, the *optical power* launched in the fiber, in optical transmission impaired by polarization dependent loss (PDL) [Lo21]. The main idea is that the BER of established lightpaths (or the equivalent SNR assuming GN) is a random variable due to the joint effect of PDL and random phase rotations along the line, with different regimes (i.e., linear or nonlinear) resulting in different shapes of the SNR probability density function (PDF).

In [An24] we extended the work of [Lo21] by numerically exploring ML methods and we optimized the input features of the SNR PDFs which are fed to the ML algorithm. A result of this investigation was that 70% reduction of the input feature vector can be achieved while reaching an accuracy above 98% in all cases, while we also validated the numerically trained algorithms on limited experimental samples, resulting in a fair classification accuracy of the transmission regime.

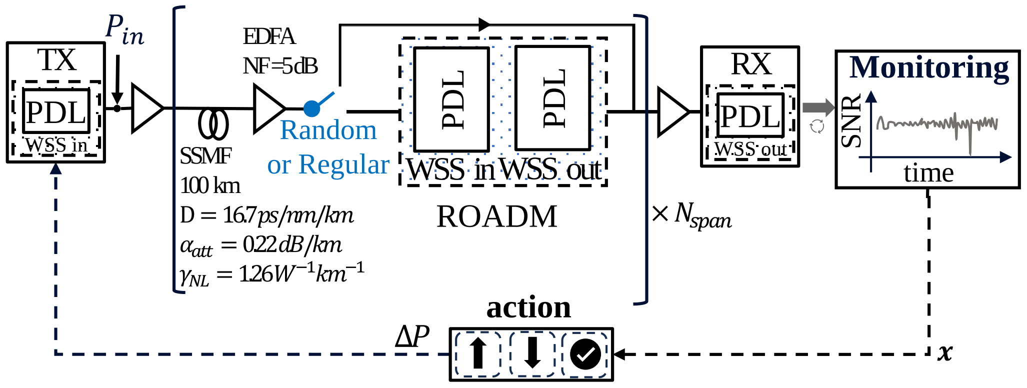


Figure 59: Setup used to collect the SNR samples. The optical link consists of a repetition of an SSMF+EDFA span, followed by either a “regular” or “random” location of the ROADM (two WSS cascade).

In Figure 59 we show the system setup, where the loop between the Rx and Tx is for now switched off. We consider that WSSs in ROADMs add PDL, drawn from different distributions. Two configurations were investigated, the “regular” and the “random”, as shown by the switch in Figure 59. In the regular configuration a ROADM was placed after three consecutive spans with PDL values of each element randomly selected from a uniform distribution ranging between 0.1 and 1 dB. In the “random” configuration ROADMs were inserted randomly after each span with a 30% likelihood, with PDL elements drawn from a chi-square (χ^2) distribution with three degrees of freedom, with a mean of 0.2 dB and a probability of exceeding 0.8 dB set at 1.05%. We considered 21 Gaussian-modulated channels transmitted with symbol rates R=49 and channel spacings W=50GHz or R=69 Gbaud and W=75GHz. The transmitted power varied in the range [-10,10] dBm with steps of 0.5 dB, while the system length was 12, 15, 18 or 21 spans. One million SNR samples were collected for each simulation using the model of [Se20] and then PDFs were a) *normalized* to takes values in the range [p_{lim},1] and b) *downsampled* to a number of bins equal to *N*. Consequently, four ML algorithms have been tested: a) k nearest neighbors (KNN), b) random forest (RF), c) support vector machine (SVN) and d) shallow artificial NN (details found in [An24]).

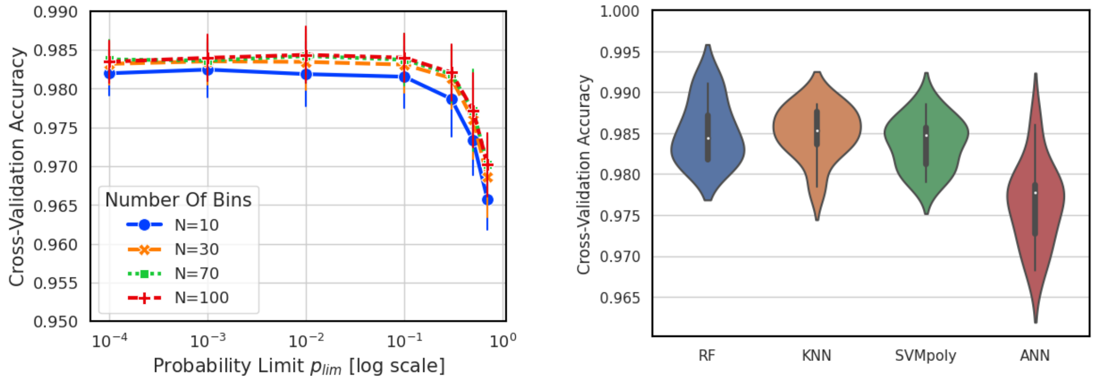


Figure 60: (a) RF cross-validation accuracy as a function of probability limit, p_{lim} for variable number of bins N. (b) Cross-validation accuracy violin plots for RF, KNN, SVM and ANN.

In Figure 60a, we plot the cross-validation accuracy for the RF algorithm as a function of p_{lim} and a variable number of number of bins *N*, while qualitatively similar results are obtained for the other three algorithms (not shown here). We note that for p_{lim}<10^{-1}, the cross-validation accuracy is almost constant, while in a similar manner, the benefits of increasing the number of bins *N* above 70, does not bring considerable benefits. Using the optimal parameters p_{lim} and *N* for all algorithms, in Figure 60b we show the cross-validation accuracy violin plots. We note that all algorithms yield similar accuracies, with RF being slightly more accurate than the others. Finally, we measured the testing time in our server to be 0.369±0.018 sec for KNN, 0.062±0.0002 seconds for RF, 0.0919±0.0002 seconds for SVM and 0.0012±0.00001 seconds for ANN, suggesting that RF achieves a fair tradeoff between accuracy and runtime complexity.

In [AnJOCN24] we consider again the system of Figure 59, but this time with the closed loop between Rx and Tx *switched on*, to achieve optimization of the transmitted power. To do this we propose a low-complexity method based on 2 RF classifiers, one "fine" classifier A and a "coarse" classifier B. For A and B, power is classified in three regimes, linear regime (LIN), close to the optimal power NLT (MID) and nonlinear regime (NLIN), with the only difference that the MID regime is narrower for classifier A compared to classifier B, as detailed in Figure 61a. Furthermore, we investigate the usage of statistical moments of the SNR samples as input features of the ML algorithms, namely the average μ, the variance σ^2, the skewness γ and the kurtosis κ. For comparison, we also introduce a *naive power optimization* method which can be briefly described as follows: 1) adjust power at Tx and establish which adjustment direction (i.e., increasing or decreasing power) results in an *average SNR* E[SNR] increase at Rx, 2) continue adjusting power at Tx with small steps and measuring E[SNR] at Rx until E[SNR] decreases, 3) go one step back and stop power adjustment.

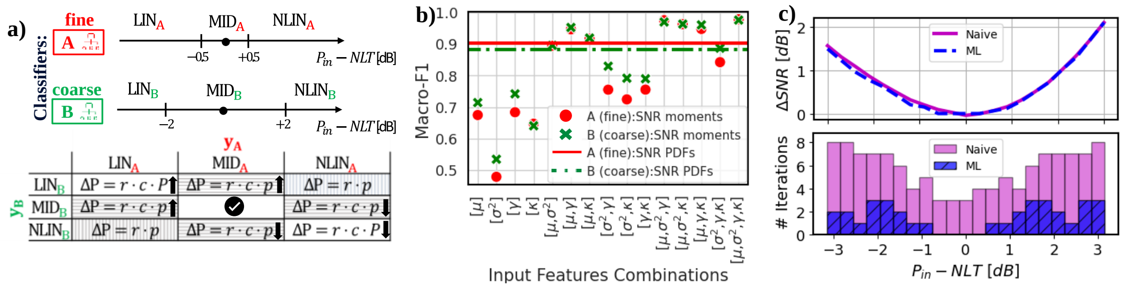


Figure 61: a) Classifiers power ranges and corresponding corrections, with c, P and p constants and $r = \pm 1$ b) Cross-validation ML performance for Classifiers A and B using combinations of SNR moments (markers) or normalized PDF (lines). c) ML vs. naive optimization in terms SNR improvement (upper) and required iterations (lower).

In Figure 61b, we show the macro F1 cross-validation performance of classifiers A and B as a function of different combinations of moments (markers), or using the entire normalized PDF (lines), as in [An24]. We note that optimal results arise when using all four input features [μ, σ^2, γ, κ] but the F1 score is only marginally higher compared to using only [μ, σ^2, γ] or [μ, γ]. Therefore, skewness and average SNR are the most impactful moments, while, using moments generally outperforms using the entire PDF. Finally, in Figure 61c we show the optimization results for the two strategies. In the top chart, SNR improvements *ΔSNR* are plotted against the difference between initial power P_{in} and NLT, showing that SNR improvements are quite similar, with the naïve strategy slightly outperforming the ML strategy in the ASE-dominated range. In the bottom histogram the number of iterations is plotted against power, illustrating that the ML-based strategy offers a significant reduction of the number of iterations, both near and far from the NLT. Interpreting this result, we suggest that, near NLT, a minimal number of iterations is needed thanks to ML's ability to efficiently identify the optimal range, while far from NLT, fewer iterations are required due to a faster power correction achieved by larger steps.

6.2 HASH-BASED TECHNIQUES FOR THE DETECTION OF LARGE FLOWS IN P4 SCENARIOS

Packet-optical nodes employing open operating systems like SONiC featuring high-speed long-reach coherent optical pluggables (e.g., 400G ZR+) have appeared in the optical arena to challenge classical transponder-based systems.

In addition to having high-speed pluggables, such packet-optical boxes may include programmable ASICs leveraging P4 technology [Bos14]. P4 has demonstrated its potential in a wide range of scenarios, including advanced monitoring and telemetry, latency-aware scheduling and forwarding, 5G function acceleration, cyber-security, in-network AI, etc. A broad summary of these use cases can be found in [Cug22, Cug23].

However, monitoring packet streams at line rates equal or above 100 Gb/s is extremely challenging, especially given the fact that flow size typically follows a Pareto distribution. This means that a few minor flows (the elephants) contribute with the majority of packets and traffic in general and vice versa, most of the flows (the mice) contain only a few packets [Fan99, Mol11].

Indeed, the authors in [Jur21] examine a large number of flows in a university campus in Poland, and show that almost two thirds of flows contain only 1 or 2 packets. This means that sampling packets randomly will very likely not observe a great portion of the packet flows, thus very inaccurate to identify flow cardinality or estimate heavy hitters. Thus, sampling is not possible and all packets must be inspected to identify all flows, and only hash-based techniques can be used since they are compact in memory and processing requirements.

There exist different probabilistic data structures based on hashing techniques (i.e., filters and sketches mainly) that provide accurate summaries (not exact) efficiently in terms of time and memory requirements, which can be used to query streams of packets. Some examples of such techniques include Bloom Filters, HyperLogLog and Count-Min Sketch and allow:

- To test if a packet belongs to a group of flows or not, for instance, a black list (using Bloom Filters).
- To obtain cardinality of flows, i.e., how many different flows are traversing a given port (using HyperLogLog algorithm).
- To identify the top-K heaviest flows/heaters, i.e., top 10 heavy-hitters or top-20, etc (using Count-Min Sketch or CMS).

These summaries allow to perform specific operations on all flows at the data plane (in a programmable data plane like P4) with reduced memory requirements. In fact, the authors in [Nam22] have implemented many of these hash-based algorithms in P4 and made the code open-source (SketchLib) for further experimentation by the research community. The next sections briefly review the design guidelines for P4 programmable dataplanes willing to include CMS in its pipeline to perform heavy-hitter detection and processing. In addition, the use cases and applications of CMS in the context of IPoWDM optical networks with P4-based packet-optical nodes are also studied, along with a real implementation on an Intel-based P4 Tofino scenario.

6.2.1 Packet and flow statistics in aggregated IP nodes

The authors in [Jur21] released detailed statistics for the research community regarding packet traces collected at a university campus in Poland in year 2021. Some of the results observed include: (1) average packet size of 870.6 Bytes, (2) Average flow size 68410 Bytes (78.5 packets), (3) Flows with only 1 packet: 47.8%, and (4) Flows with only 1 or 2 packets: 65%.

Another important observation is that flow size follows a Pareto-like distribution, where the majority of flows contain very few packets, while very few heavy hitter flows comprise most of the traffic (in terms of packets), confirming past observations in different scenarios [Fan99, Mol11]. Following these numbers, Table 15 summarizes both flow and packet rates per switch port for a packet-optical node whose ports operate at line rates of 100 Gb/s and 400 Gb/s at different loads.

Table 15. Flow and packet rate per port at 100 and 400 Gb/s (for different loads)

100 Gb/s			
Load	10%	40%	70%
Flow rate	18.25 K flow/s	73 K flow/s	127.75 K flow/s
Packet rate	1.54 M packet/s	6.17 M packet/s	10.8 M packet/s
400 Gb/s			
Load	10%	40%	70%
Flow rate	73 K flow/s	292 K flow/s	511 K flow/s
Packet rate	6.16 M packet/s	24.64 M packet/s	43.12 M packet/

For instance, taking a 1-sec time window, a 100G port will experience the following traversed traffic:

- 73K different Flows, most of them with just 1 or 2 packets
- 6.17M total packets on average.

6.2.2 Count-Min Sketches for traffic flows detection

The CMS is a data-structure that allows to store the frequencies of each flowID in a compact manner [Cor05]. It is very similar to having multiple Counting Bloom Filters. The challenge is again to store the frequencies of flows (especially the heavy hitters) where most of them are unique.

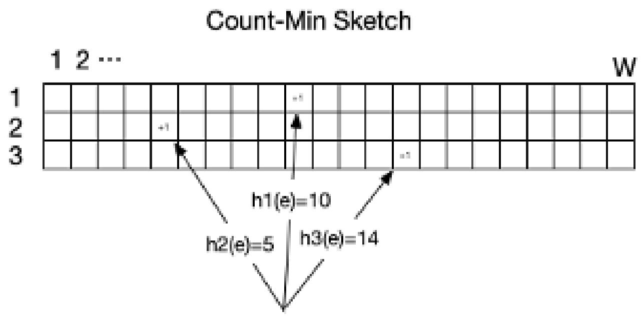


Figure 62. Example of a CMS structure, d=3 hash functions and W columns

To do this, the CMS comprises a matrix with *d* rows (one per hash function) and *w* columns, as shown in Figure 62. When an element arrives (a packet in this scenario), the CMS computes all *d* hash functions (one per row) and increases by one each position in the appropriate column. For instance, in the example of Figure 62 with *d=3* rows and *w=20* columns, let us assume that the hashing of packet *e* gives the following results: $h_1(e)=10$, $h_2(e)=5$ and $h_3(e)=14$. These values imply that the positions *[1,10]*, *[2,5]* and *[3,14]* in the CMS matrix should be increased by one, as shown in the figure. Thus, every single packet traversing the port requires computing *d* different hash functions of it and increasing the corresponding positions $[d,h_d(e)]$ of the CMS sketch.

After all elements (packets) are introduced, the CMS can be queried to get an approximate of frequency for a particular element. Consider we want to estimate the popularity of element *y*. To do this, the procedure is to hash element *y* and take the minimum value. Consider for example, that the CMS array returns the following results: $h_1(y) = 6$, $h_2(y) = 7$ and $h_3(y) = 2$, and looking at positions *[1,6]*, *[2,7]* and *[3,2]* of the CMS matrix, we observe values 5, 7, and 5 again. The number of packets arrived so far with *flowID = y* would then be *min(5,7,5) = 5 packets*. This is an estimate and some errors may have occurred due to hash collisions (as it is probably the second counter *[2,7]* since the other two counters output 5 packets). For this reason, it is necessary to accurately dimension the CMS structure for a given expected number of elements, that is, sufficient rows and columns.

In general, CMS is an *(e,d)* probabilistic data structure, where the result returned is bounded to at most $\varepsilon\|Count\|$ with probability $1-\delta$, while both e and *d* can be obtained from the CMS design parameters *d* and *w* as it follows:

$$d = \lceil ln(1/\delta) \rceil \quad \text{and} \quad w = \left\lceil \frac{e}{\varepsilon} \right\rceil \tag{8}$$

where *e* is the estimation error.

Table 16 shows a number of typical configurations of CMS (i.e. *d* and *w*) and associated accuracy and error.

Table 16.CMS dimensioning examples

Hash	Acc	256	512	1024	2048	4096	8192
3 hash	95%	1.1%	0.6%	0.27%	0.14%	0.068%	0.034%
5 hash	99.32%	1.1%	0.6%	0.27%	0.14%	0.068%	0.034%
7 hash	99.91%	1.1%	0.6%	0.27%	0.14%	0.068%	0.034%
9 hash	99.988%	1.1%	0.6%	0.27%	0.14%	0.068%	0.034%
11 hash	99.998%	1.1%	0.6%	0.27%	0.14%	0.068%	0.034%

The way to read Table 16 is as follows: Consider we are expecting 70,000 flow IDs in a time window of 1 second (as an example of 100 Gb/s port), where the largest data flow is expected to contribute with 2,000 packets. Then, with probability 0.9991, we are interested in error estimated below 1%. Hence we need:

$$\begin{gathered} d = \left\lceil \ln\left(\frac{1}{1} - 0.9991\right) \right\rceil = 7\ hash \\ w = \left\lceil \frac{2.71}{0.01} \right\rceil = 272\ columns \end{gathered} \tag{9}$$

Looking at the table, we decide to use a CMS with 7 hash functions and 512 columns, where each column can allocate 8 bits of counting. The total amount of memory consumed is approximately *7 x 512 x 8 =28,672bit*, i.e., *28Kbit.*

6.2.3 Scenario and experiments

<u>*Network setting and CMS dimensioning*</u>

Let us consider a monitoring time window of 0.1 seconds, where a blank CMS is zeroed at time $t_0=0$ and is populated with the number of packet arrivals until $t_w=0.1s$. Following Table 15 (100 Gb/s, 40% load), we should expect around 7,000 different flows whose size follows a zipf distribution. In this regard, we have simulated a packet trace with N=7,000 different flow IDs and packet sizes following a zipf distribution with scale *a=1.1* (following the examples of [Her19]. This means that the *k*-th flow has a frequency f_k following:

$$f_k = \frac{\frac{1}{k^\alpha}}{\sum_{n=1}^{N} \frac{1}{n^\alpha}}, \quad k = 1, \dots, N \tag{10}$$

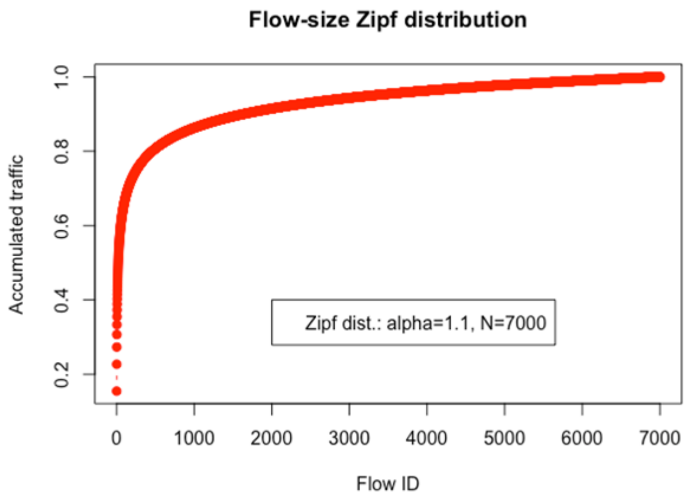


Figure 63. Flow-size distribution: Zipf-like CDF

As shown in Figure 63, the largest flow contributes with 15% of the total traffic while the top-20 heaviest hitters comprise 49% of the total traffic (almost the same as the other 6,980 flows). In this scenario, identifying the top-20 flows allows operators to take decisions regarding their treatment, i.e., using different optical bands or alternative routes for load balancing and QoS guarantees.

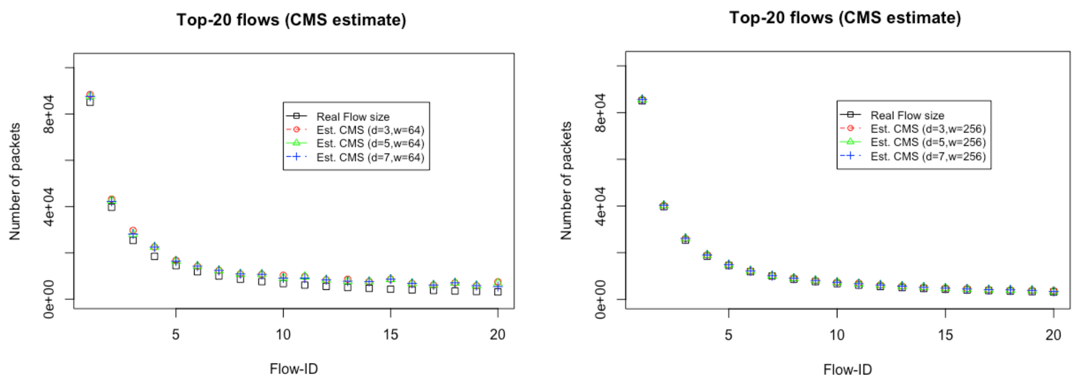


Figure 64. CMS accuracy at estimating top-20 heavy-hitter flows for different configurations of CMS sketch: (left) d = {3,5,7} hashes and W=64 columns and (right) d = {3,5,7} and W=256 columns

Figure 64 further shows the estimated flow size (in no. of packets) for the top-20 heaviest hitters for different CMS configurations, i.e. values of *d = {3,5,7}* hashes and *w={64,256}* columns.

6.2.4 Experimental validation

The CMS-based solution has been validated using a network switch, equipped with P4 ASIC, designed for data center operations and potentially suitable for IPoWDM implementations (as shown in Figure 65. The traffic flows have been generated using a Spirent N4U. The P4 implementation uses five hash functions on the IP source address, each one producing a result between 0 and 4,096 bit (=2^{12}), while the total CMS occupies one register of size 65,536 bit.

In a first experiment, with the aim of collecting baseline performance, the traffic is sent from one source to one destination host, at a maximum rate of 9.85 Gb/s over a 10G port with fixed packet size of 512 bytes. The experienced latency to traverse the switch (including hashing and CMS processing) is *0.95 ms*. Then, traffic is generated by 5 different sub-networks each one comprising 256 hosts (i.e 1,280 IP source addresses) that send traffic to another destination sub-network comprising 256 hosts. The average latency to traverse the switch interconnecting both source and destination subnetworks is again *0.95 ms*. This values slightly increases to *1.35 ms* for packet size values of 1,024 bytes. Thus, following the implementation of the CMS in a real P4 ASIC switch, we observe that every single packet must traverse all the stages in the pipeline of the P4 implementation, resulting in a total latency that varies between *0.95 ms* and *1.35 ms* of processing time.

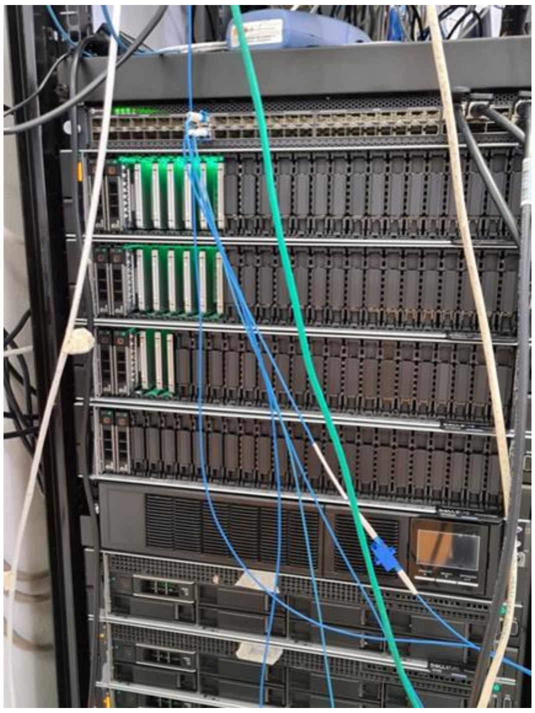


Figure 65. Experimental setup in lab

6.3 Autonomous Capacity Operation

With the deployment of network slicing and the support to time-sensitive applications, the flow capacity autonomous operation (hereafter referred to as CRUX) focuses on answering a major problem that network operators are facing nowadays: how to allocate the right capacity to every traffic flow, so as to provide the desired QoS (e.g., by preventing traffic loss and ensuring a given maximum average delay), while minimizing overprovisioning (i.e., *capacity–traffic*). Although many ML techniques could be potentially applied for the autonomous capacity operation of traffic flows, in this section, we rely on RL. Several works have proposed the application of RL algorithms for autonomous network operation, e.g., *RL*-based algorithms running in the SDN controller can be used for dynamic lightpath provisioning or for autonomous bandwidth allocation in the context of multilayer optical networks.

One of the key issues in the previous works is the time needed to learn optimal policies, as exploration entails low-reward decision making (i.e., far from optimal operation). In view of this, the performance of RL methods is typically evaluated after some training phase, i.e., when reward achieves a stationary behavior. However, a subject that is poorly or not even considered so far is when RL algorithms need to operate before they are properly trained. Note that this happens in our case, as the actual characteristics of the traffic flow are unknown until it is provisioned. Moreover, it is not realistic to assume, in general, the same conditions during training and operation phases, due to mid/long-term traffic evolution, which makes it difficult to reproduce highly accurate operation conditions during training. To solve these issues, a general learning lifecycle can be used that included both offline training (e.g., in a sandbox domain) and online learning, in the context of supervised ML, with the objective to accelerate autonomous operation by deploying accurate models that are firstly trained offline and fine-tuned while in operation, thus adapting pre-trained models to actual in-field operation conditions.

In this section, we apply the main lessons learnt from the previous works to IBN agents based on RL. We assume that a packet flow (alternatively, referred as traffic flow or simply flow) conveying traffic with unknown characteristics is established and the allocated capacity needs to be set to ensure the required QoS (from the set-up time), while minimizing resource utilization. To this end, a policy-based management is used at the set-up time to start operating the capacity of the flow; meanwhile, traffic measurements are collected to characterize the traffic. Note that policy-based operation can be highly reliable, as it is based on specific rules that can be defined and understood by human operators. However, such an operation usually obtains poor resource utilization. Therefore, it would be useful to substitute policy-based operation by an RL model as soon as possible. To that end, pre-trained generally applicable models for the *partly* observed traffic characteristics are loaded and the RL algorithm starts operating. A per-flow algorithm supervises the performance of the RL algorithm and tunes model parameters to ensure the required QoS. Once enough traffic measurements are available, offline training is carried out in a sandbox domain to produce a specific model well-suited for that particular traffic flow, which then substitutes the generic one. Since the traffic characteristics can change over time, analysis must be continuously performed to detect them and change the operating model when needed. By iterating that cycle, the proposed RL agent will be able to adapt to traffic changes that would otherwise degrade performance.

6.3.1 The Flow Capacity Autonomous Operation (CRUX) Problem

Flow Capacity Autonomous Operation

Let us start by analyzing the result of the autonomous capacity operation of a traffic *flow*. A *flow manager* might collect monitoring data from the network and expose some interface, so an RL algorithm can take action. For illustrative purposes, Figure 66 shows a typical RL framework, where the learning agent is separated into two different blocks, the *learner* and the *agent*.

Let us assume that the monitoring data include the byte count since the last monitoring sample (amount of traffic) and the actual capacity allocated to the flow. The actions to be taken are related to the actual capacity allocated to the flow, which can be increased or decreased as needed with some *granularity* to meet the required QoS. For instance, a customer connection can manage the capacity of the flow with granularity 1 Gb/s by configuring some packet node, whereas in a virtual link supported by the optical layer, the capacity can be increased/decreased by establishing or tearing down parallel lightpaths, each with a capacity of 100 s Gb/s. It seems clear that the time to change the capacity is also different, ranging from seconds to minutes. The RL algorithm should then decide the capacity to be allocated to the flow to absorb *variation* in the traffic from one monitoring sample to the next, plus the time to increase the allocated capacity, with the objective to avoid any traffic loss and ensure some additional QoS metric.

Then, the *traffic variation* becomes a major feature for a flow, together with the *traffic pattern*, i.e., the evolution of the mean traffic with time.

This approach can provide excellent performance once the policies that avoid traffic losses meet the desired QoS, and minimize overprovisioning are learned; however, online learning of such policies requires time. In addition, there are several issues that can impact the aforementioned online learning performance, e.g., (1) changes in traffic variability might produce loss before new policies are learned; (2) smooth model fine tuning could not be enough to mitigate persistent errors in taking some specific actions; and (3) online learning tends to forget valuable learning in the long run, thus reducing the model's accuracy.

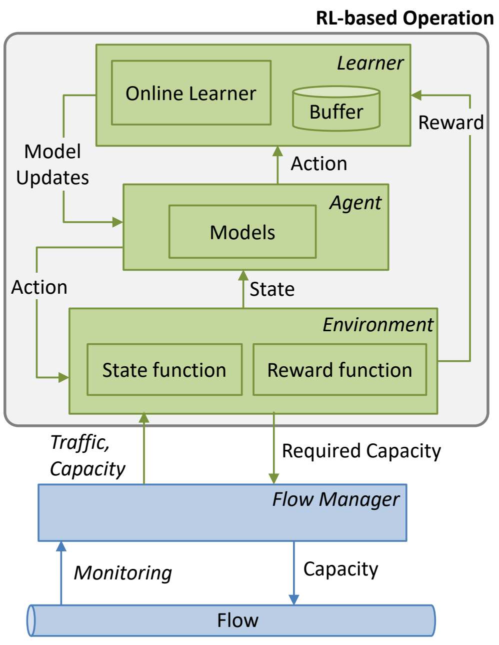


Figure 66. Flow capacity autonomous operation. RL framework with learner, agent, and environment.

Figure 67a represents a possible evolution of the traffic variation (the traffic pattern is omitted here for simplicity) and the obtained performance—overprovisioning, traffic loss, and some other QoS metric. The path supporting the flow is established at time t_0 and the desired QoS is specified, so the RL algorithm needs time to learn the traffic variation (and the traffic pattern); meanwhile (until *ta* in Figure 67a), poor performance, including traffic loss, can be expected. Once a good model is obtained, it is expected that an RL algorithm can provide the target performance. However, a steep change in the variation of the traffic (times t_1 to t_2) can impact the performance until the new variation is learned. Nonetheless, it might happen that the performance does not converge to the desire level even after learning the new traffic variation.

It seems clear that the above behavior is unacceptable for network operators, as it would provide poor performance and might incur penalties due to SLA breaches. Specifically, it seems of paramount importance to start the operation with already trained models. To that end, an initial model can be trained offline using a network simulator in a sandbox domain. Once in operation, the model will be improved by the online learner. However, there are traffic characteristics, e.g., traffic pattern, that are observed after a long period of time, e.g., several days. Therefore, some alternatives are needed to operate the flow during that initial time.

Our solutions go beyond training offline and propose implementing offline–online learning cycles to deal with large changes in traffic flow, i.e., to provide guaranteed performance during the whole lifetime of the traffic flow (Figure 67b). Specifically, (*i*) a policy-based management implemented as a self-tuned *threshold-based* algorithm is in charge of managing the flow capacity during the time immediately after the path is set up (*Phase I*: time interval [t_0, *ta'*], where *ta'*-t_0 should be short, e.g., 1 h). That algorithm tunes a threshold for the flow capacity and accurately determines the traffic variation. This approach enables dynamic flow capacity allocation by fixing the right values for the threshold that minimize overprovisioning. However, avoiding traffic loss and guaranteeing that the required QoS is met is not a straightforward task, as it depends on the variance in the traffic flow—defined as the difference between maximum and minimum amount of traffic during some period. Therefore, during this period, the threshold is set conservatively to avoid *underprovisioning* (i.e., traffic exceeds capacity, and some traffic is loss) at the expense of large overprovisioning. (*ii*) Once the variation of the traffic has been determined, a pre-trained *generic model* can be used for flow operation (*Phase II*: time interval [*ta'*, *tb'*]). The model is general as it has been pre-trained assuming a given traffic pattern, e.g., sinusoidal with daily periodicity, but supporting the measured traffic variation. Once in operation, the pre-trained model starts to fine tune with the observed samples. (*iii*) Once enough measurements are available to determine the characteristics of the flow, including the traffic pattern, a specific model can be trained in a sandbox domain by using a simulator set to operate at time *tb'* (*Phase III*). That model should improve the performance or be easier to operate than the pre-trained one. (*iv*) Assuming that the traffic pattern does not change, any change in the traffic variation that cannot be absorbed by the current model can trigger returning to Phase II for more intensive parameter tuning for the new traffic variation, while a new specific model is trained and is set to operate at time *tc'* (Phase II—Phase III cycle).

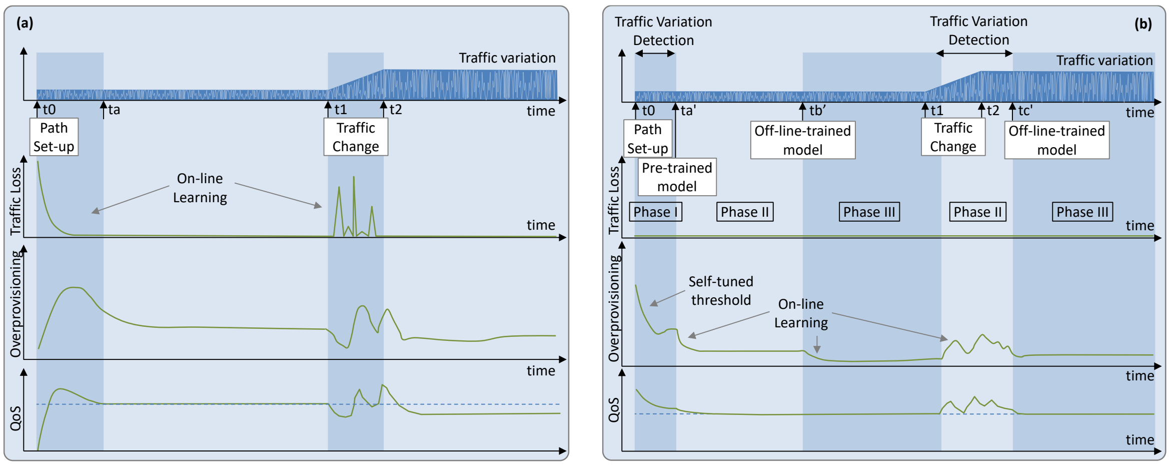


Figure 67. Operation lifecycle. (a) Online learning RL operation. (b) Offline training with online fine tuning RL operation.

2.2. Proposed Architecture

This work extends the basic RL-based flow capacity operation (Figure 66) and proposes a scheme based on (Figure 68): (*i*) analyzing the traffic to obtain meaningful traffic characteristics; (*ii*) making decisions regarding the allocated capacity when no model is in operation (Phase I); (*iii*) selecting pre-trained models that fit with the observed traffic characteristics; (*iv*) training new models in a sandbox domain (*offline learning*), where real traffic measurements are used to generate traffic in a simulation environment and the QoS can be realistically estimated; a replica of the RL algorithm in operation is used here for training new models; and (*v*) once accurate models are obtained, they are used for flow operation and will be progressively fine-tuned online. As in Figure 66, a *Flow Manager* collects monitoring data from the forwarding plane and enforces flow capacity.

The models include some parameters that need to be tuned as a function of the traffic, to provide the desired performance while meeting the required QoS. Such parameter tuning can be carried out during offline learning, as well as during online operation to deal with small traffic changes. Based on the analysis of the traffic and the reward, the Analyzer block decides when to tune parameters and when to update the model with an offline learned one (labeled *Set*() in Figure 68) to meet the given QoS. Note that both parameter tuning and the offline–online cycle can be completed several times during the operation to improve the learned models, which will also enable adaptability to changes.

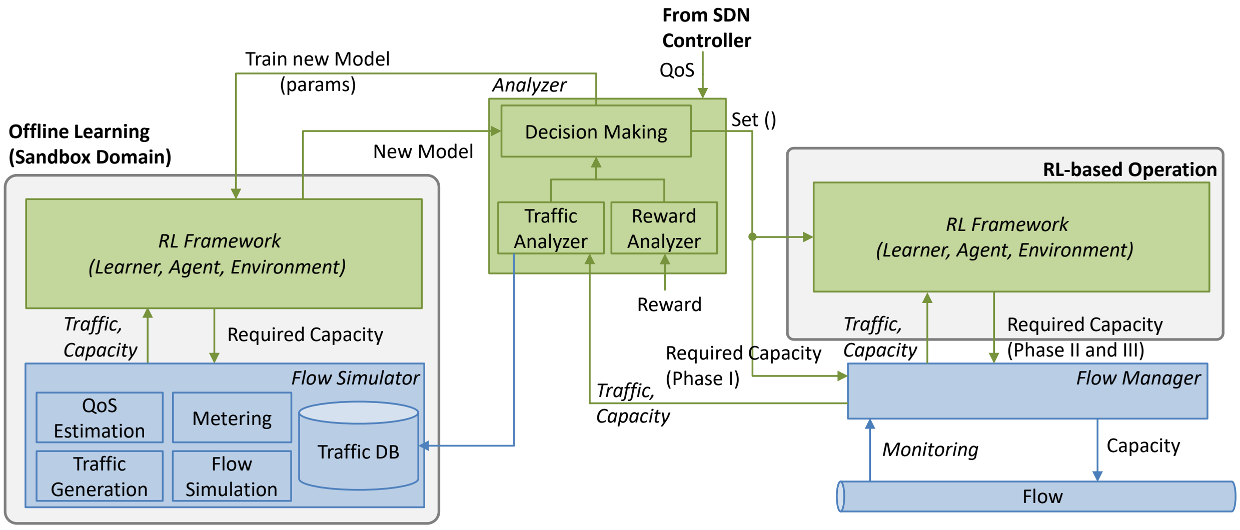


Figure 68. Extended architecture for flow capacity autonomous operation w/ offline learning.

The next section details the RL approaches used to solve the autonomous flow capacity management problem, CRUX.

6.3.2 CRUX Problem Definition and RL Methodology

This section formally defines the CRUX problem, and introduces the main parameters and variables used hereafter. Next, it introduces the methodology to solve the problem using RL, and finally defines the different RL approaches under study. The used notation is summarized in Table 17, where parameters and variables are defined.

Table 17. Notation - Autonomous Capacity Operation.

	Capacity and QoS Params for the Flow
z_{max}	Maximum capacity (Gb/s)
d_{max}	Target maximum delay (s)
l^*	Optimal load (unleashing d_{max}) ∊ [0, 100]%
qa	QoS Assurance (%)
	Traffic and Capacity
$x_{max}(t)$	Maximum traffic at time *t* (Gb/s)
$x_{var}(t)$	Traffic variation at time *t* (Gb/s)
$z(t)$	Capacity allocated at time *t* (Gb/s)
$o(t)$	Capacity slack/surplus at time *t* (Gb/s)
$y(t)$	Overprovisioning margin (Gb/s)
ρ	Traffic variance multiplier
b	Granularity of capacity allocation (Gb/s)
$w(t)$	Traffic loss margin (Gb/s)
	Autonomous Capacity Allocation
$a(t)$	Action time *t* (b/s)

n_a	Number of discrete actions
$s(t)$	State at time t
n_s	Number of discrete states
$r(t)$	Reward at time t
k	Threshold-based scaling factor
β_i	Reward function coefficients (≥0)

Problem Definition and Basic Modeling

Let us consider that the autonomous flow capacity management problem is solved periodically, when a new set of measurements, statistics, and parameters for the traffic flow *x* are collected and computed. Along this section and the following, we adopt the *informational representation* of time, where *t* represents a point in time that refers to the time interval [*t* - 1, *t*). Specifically, $x(t)$ represents the traffic measurements collected in [*t* - 1, *t*), and statistics, such as the maximum ($x_{max}(t)$) and variation ($x_{var}(t)$), summarize traffic dynamicity during that time interval. Additionally, decisions made at time *t*, e.g., the capacity to be allocated ($z(t)$), consider data that arrived up to time *t*.

The main objective of the CRUX problem is to find the optimal (minimum) capacity $z(t)$ at time *t* that satisfies a desired QoS. In this work, we assume that that the QoS is defined by a desirable maximum end-to-end delay d_{max}; this also entails that packet loss is not tolerated during flow operation.

Without loss of generality, we assume that the delay can be modeled as a function of the traffic volume, the allocated capacity for the flow, the *load* (ratio traffic/capacity), and other components such as the transmission delay (*load–delay* models). Such load–delay models can be obtained during the commissioning testing phase using, e.g., active monitoring techniques. Once the model is available, a target load l^* unleashing the target maximum delay d_{max} can be selected. Figure 69 illustrates an example of a load–delay model, where d_{max} has been selected to a value where queueing delay becomes the predominant delay component, e.g., for l^* = 80%.

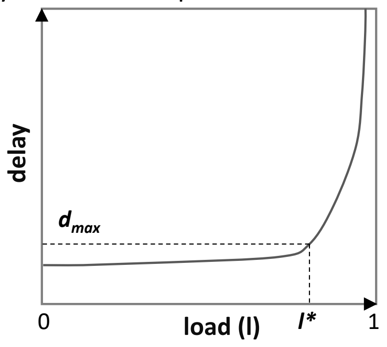


Figure 69. Load-delay model example.

A policy (threshold) -based approach can be used to make decisions from the currently available monitoring data, as defined in eq. (11), where *k* is a constant factor that is related to the traffic dynamicity and variability; *k* needs to be tuned to guarantee the required QoS. However, finding the proper value of *k* is not a straightforward task: if the value of *k* is high, QoS is ensured at the cost of high overprovisioning, whereas if the value of *k* is low, QoS requirements might be not met. In addition, decisions are reactive, so altogether, *sub-optimal solutions* are usually obtained.

$$z(t) = k \cdot x_{max}(t)/l^* \tag{11}$$

The optimal capacity allocation to the CRUX problem requires knowledge of the expected traffic to allocate the capacity of the flow at time *t*-1 to the value that fits the expected maximum load for the period [*t*-1, *t*) (proactive decision making), i.e.:

$$z^*(t-1) = x_{max}(t)/l^* \quad (12)$$

In the case that the capacity allocation is not optimal, some capacity slack / surplus (*o*) will exist, which can be formally computed at time (*t*) as follows:

$$o(t) = z(t-1) - x_{max}(t)/l^* \quad (13)$$

Figure 70 sketches an example of a traffic flow *x*(*t*) for which some capacity allocation *z*(*t*) is required. In the figure, the optimal capacity *z**(*t*) that should be allocated at time t_{i-1} is shown. The different colors provide a visual representation of the values of *o*(*t*). In particular, two different sub-optimal capacity allocations can be distinguished (see labels in Figure 70a): *i*) if *z*(*t*) > *z**(*t*) (i.e., *o*(*t*)>0), QoS requirements are met at the expense of an excess of overprovisioning; *ii*) if *z*(*t*) < *z**(*t*) (i.e., *o*(*t*)<0), QoS requirements are violated.

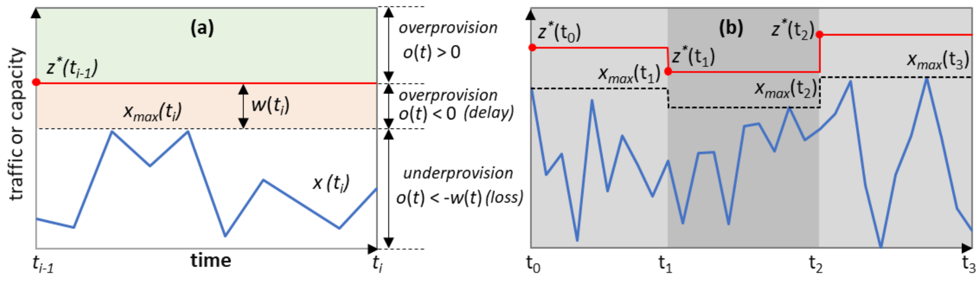


Figure 70. Capacity allocation definition (a) and evolution (b).

The width *w*(*t*) of the high delay area is formally defined as a function of the maximum traffic in eq. (14). Therefore, traffic loss appears if *o*(*t*) ≤ -*w*(*t*).

$$w(t) = x_{max}(t) * \frac{1-l^*}{l^*} \quad (14)$$

It is worth noting that the quality of a solution taken at time *t* - 1 can only be evaluated at time *t*, which motivates the use of RL to learn the optimal policy that allocates the minimum value of *z*(*t*) to meet the QoS requirements. The details of the RL-based methodology are presented in next subsections.

Generic RL-Based Methodology

Figure 71 illustrates the RL workflow, where the main three elements involved are represented, namely: (*i*) the learner in charge of learning the optimal policy; (*ii*) the agent in charge of taking actions to adjust the capacity allocated to the flow; (*iii*) the environment adaptation module in charge of implementing and evaluating the actions taken; and (*iv*) the flow manager, which enforces the capacity and collects traffic measurements. Three time periods are specified, from t_0 to t_2; let us assume that some initial policy model has been set in the agent before operation starts at time t_0, when the agent applies the first action $a(t_0)$.

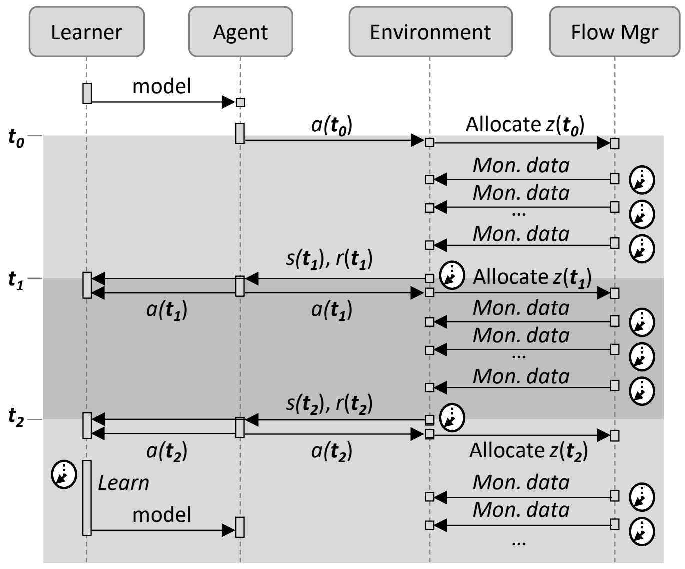


Figure 71. General RL workflow.

For the sake of simplification and to reduce complexity, action $a(t)$ is defined in eq. (15) as the differential capacity with regard to the current one. Actions are processed by the environment, which computes the new capacity $z(t)$ to be allocated.

$$a(t) = z(t) - z(t-1) \tag{15}$$

The flow manager periodically sends traffic monitoring data to the environment, which processes them at the end of every time interval to compute state $s(t_i)$ and reward $r(t_i)$. Upon receiving the state, the agent finds the action $a(t_i)$ to be taken with the current policy. In addition, the learning process uses state, reward, and action to improve the model, which is updated in the next time interval. The state function $s(t)$ is defined in terms of $o(t)$ normalized by a parameter $y(t)$ (eq. (16)(17)), which is conveniently set up using parameter ρ to absorb the traffic variation observed in the flow (eq. (17)).

$$s(t) = o(t)/y(t) \tag{16}$$

$$y(t) = \rho \cdot x_{var}(t) \tag{17}$$

As for the reward $r(t)$, the objective is to minimize overprovisioning, without providing high delay. To ensure that, the higher delay must be obtained by producing some overprovisioning. To that end, we have defined a piece-wise function with four different regions in the range of $o(t)$, representing the sub-optimal cases. Such division allows for individual modes of operation that correspond to an adequate reward in each case. The reward function $r(t)$ is formally expressed in eq. (18) and illustrated in Figure 72. The first and second components of $r(t)$ penalize traffic loss and high delay, respectively. Both components are linear functions of $o(t)$, where coefficients β_1 and β_2 can be tuned to penalize traffic loss and high delay. The third component gives the maximum reward, which is slightly shifted to the positive values of $o(t)$ to reduce the risk of QoS violation. This segment is concave quadratic with regard to the relation $o(t)/y(t)$, with the maximum value weighted by coefficient β_3. Finally, overprovisioning above $y(t)$ is linearly penalized by coefficient β_4.

$$r(t)=\begin{cases}\beta_1\cdot\big(o(t)-w(t)\big)+\beta_2\cdot w(t), & o(t)<-w(t)\\ \beta_2\cdot o(t), & -w(t)\le o(t)<0\\ \beta_3\cdot\left(1-\dfrac{o(t)}{y(t)}\right)\cdot\dfrac{o(t)}{y(t)}, & 0\le o(t)<y(t)\\ -\beta_4\cdot(o(t)-y(t)), & o(t)\ge y(t)\end{cases}\tag{18}$$

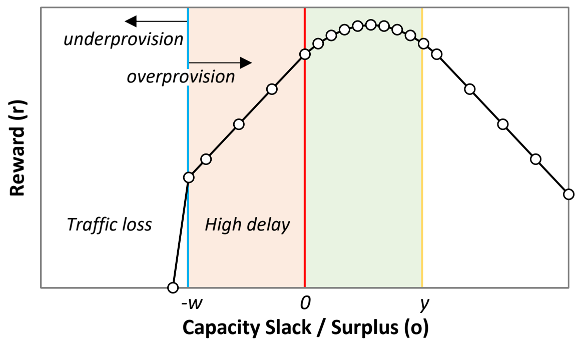


Figure 72. Reward function vs. capacity slack/surplus.

Specific Adaption of RL Approaches

As introduced in Section 1, three RL methods are considered to solve the CRUX problem, namely: (*i*) Q-learning, (*ii*) D3QN, and (*iii*) TD3.

For each method, the main adaptation of the generic problem definition is to discretize state and action spaces. Q-learning requires discretizing the continuous state function *s*(*t*) in eq. (16) into a number of discrete states (n_s). Such discrete state function *s'*(*t*) can be formally expressed as eq. (19). Discrete states 0 and n_s - 1 indicate underprovisioning and overprovisioning above margin *y*(*t*), respectively, whereas the rest states n_s - 2 are used to evenly discretize the overprovisioning below margin *y*(*t*).

$$s'(t)=\begin{cases}0, & s(t)<0\\ \lceil(n_s-2)\cdot s(t)\rceil, & s(t)\in[0,1]\\ n_s-1, & s(t)>1\end{cases}\tag{19}$$

In addition, Q-learning and all DQN variants require a discrete space of n_a actions. Let us define *a'*(*t*) as the set of discrete actions, where a discrete action is defined by an integer number of units of capacity *b* to update (add or subtract) the current capacity. The discrete set of actions that depend on both $n_a \in 2 * \mathbb{N} - 1$ (natural odd number) and *b* can be formally defined as:

$$a'(t)\in\left\{b\cdot i, i\in\left[-\frac{n_a-1}{2},\frac{n_a-1}{2}\right]\right\}\tag{20}$$

Table 18 summarizes the main characteristics and parameters to be configured for each method.

Table 18. Summary of RL approaches.

Approach	**State Space**	**Action Space**	**Parameters**
Q-learning	Discrete, eq. (19)	Discrete, eq. (20)	n_s, n_a, *b*
D3QN	Continuous, eq. (16)	Discrete, eq. (20)	n_a, *b,* DNN config, Replay buffer
TD3	Continuous eq. (16)	Continuous, eq. (20)	Actor/critic DNN config Replay buffer

6.3.3 Cycles for Robust RL

This subsection is devoted to the details of the operation lifecycle presented in Figure 67b. Let us assume that when a new path for a flow is set up, an instance of every element in the architecture in Figure 68 is instantiated. The characteristics of the instances depend on the flow requirements, e.g., pre-trained generic models loaded in the *Offline Learning* block are those that were trained with similar QoS requirements (d_{max} and q_a) to those of the current flow. When the operation starts, no *online RL* models exist and the maximum capacity for the flow z_{max} is allocated. All the algorithms presented next run in the *Analyze*r block (see Figure 68), which makes decisions and orchestrates the rest of the blocks based on some analysis results.

Algorithm 11 shows the Analyzer initialization that receives the pointers to external modules that interact with the *Analyzer*, i.e., *offline learner, online RL*, and *flow manager*, and stores them (line 1 in Algorithm 1), and initializes the main variables used by the rest of the procedures. In particular, *DB* contains the needed traffic-related data for the analysis carried out at every phase (line 2), *params* is a vector with the parameters that characterize flow's requirements and some configuration and that are used during the different phases (line 3), and *phase* records the current phase, which is initialized to Phase I (line 4).

Before describing the algorithms for the different phases, let us present a specific procedure for traffic variance analysis. Algorithm 12 is applied to the set of observed traffic-related measurements, stored in *DB*, with the objective of characterizing and quantifying the fluctuation of the traffic around its observed average. After retrieving data time series contained in *DB*, the average pattern on the given traffic time series *X* is computed (lines 1–2 in Algorithm 2). Note that the result of this operation produces time series X_{avg}, with the smoothed average that better fits *X*. Without loss of generality, we assume that a combination of regression techniques including polynomial fitting, spline cubic regression, and sum-of-sin regression is applied, returning the best result in terms of accuracy and model complexity. The relative residuals are computed (line 3) and the difference between maximum and minimum of these residuals (*var*) is considered as the traffic *variance* measurement (line 4). This value together with the previous *var* measurements stored in *DB* (time series *Y*) are used to compute the derivative *drv* of *var* in time, i.e., the first-order difference (lines 5). The last value in *drv* denotes the current derivative, which is later used for the identification of the traffic variance (line 6). In addition, a variance *score* is computed as the maximum absolute derivative value normalized by the observed variance (line 7). The score approaches 1 if the traffic fluctuates around its maximum range between two consecutive time measurements. This score will be used later for generic model tuning purposes. The computed variance results are eventually returned (line 8).

Algorithm 11. Analyzer Initialization

```
INPUT: offlineLearn, onlineRL, flowMgr          OUTPUT: -
1:  store(offlineLearn, onlineRL, flowMgr)
2:  initialize DB
3:  params ← [<l*, $z_{max}$, k, eps>,         // Phase I
              <qa, cfl, $\Delta\rho$>,              // Phase II
              <var_l, var_h, r_l, m>]     // Phase III
4:  phase ← PhaseI
```

Algorithm 12. varianceAnalysis().

```
INPUT: DB                          OUTPUT: V
1:  $<X,Y> \leftarrow <DB.\text{traffic}, DB.\text{var}>$
2:  $X_{avg} \leftarrow \text{computeAveragePattern}(X)$
3:  $X_{res} \leftarrow (X - X_{avg}) \odot X^{-1}$
```

```
4:  $var \leftarrow \max(X_{res}) - \min(X_{res})$
5:  $Y$.append($var$)
6:  $drv \leftarrow$ computeDerivative($Y$)
7:  $score \leftarrow \max(|drv|)/var$
8:  **return** <var = $var$, curdrv = $drv$[-1], score = $score$ >
```

Algorithm 13 specifies the main procedure running in the Analyzer block and it is called periodically every time *t*. First, new traffic monitoring data are gathered from the flow manager (line 1 in Algorithm 13). Then, the specific procedure for each phase is called with the current time and the collected data, and the phase changes only when the called procedure returns *True* (lines 2–10); *DB* is initialized every time phase changes (line 11).

Algorithm 13. Main Analyzer Procedure

```
INPUT: t                    OUTPUT: -
 1:  x(t) ← flowMgr.getMonitoringData(t)
 2:  if phase = PhaseI then
 3:      changePhase ← thresholdBased(t, x(t))
 4:      if changePhase then phase ← PhaseII
 5:  else if phase = PhaseII then
 6:      changePhase ← modelSelectionAndTuning(t, x(t))
 7:      if changePhase then phase ← PhaseIII
 8:  else // phase = PhaseIII
 9:      changePhase ← specificModel(t, x(t))
10:      if changePhase then phase ← PhaseII
11:  if changePhase then initialize DB
```

Algorithm 14. thresholdBased() (Phase I)

```
INPUT: t, x(t)                              OUTPUT: changePhase
 1:  DB.traffic.append(x(t))
 2:  V ← varianceAnalysis(DB)
 3:  if |V.curdrv| < eps then
 4:          <$f_0$, $\rho_0$, $sc_0$> ← offlineLearn.getGenericModel(V.var)
 5:          $\rho_1 \leftarrow \rho_0 \cdot (sc_0$ / V.score)
 6:          onlineRL.setModel($f_0$, $\rho_1$)
 7:          return True
 8:  DB.var.append(V.var)
 9:  $k(t) \leftarrow \max(1, V.var / (1-l^*))$
10:  if $k(t)>k$ then $k \leftarrow k(t)$ else $k \leftarrow k - (k(t)-k)/2$
11:  flowMgr.setupCapacity($\max(k \cdot \max(x)/l^*, z_{max})$)
12:  return False
```

Algorithm 14 defines the operation of the *Analyzer* block during Phase I. Recall that during this phase, flow capacity allocation is managed following a threshold-based procedure, defined by eq. (11). First of all, *DB* is updated with the new monitoring data and the variance analysis described in Algorithm 12 is executed, storing the result in variable *V* (lines 1–2 in Algorithm 14). Then, the absolute value of the current derivative is compared with a small epsilon value (param *eps*) to decide whether enough traffic data have been already analyzed to estimate variance with high accuracy (line 3). If so, a generic pre-trained model f_0 for the computed variance is retrieved from the *offline learner* and factor ρ_0 is scaled with the ratio of scores between the generic model and the observed traffic (lines 4–5); the scaled factor increases (decreases) if the computed score is higher (lower) than the score of the generic model. The rationale behind such factor correction is to achieve a more robust and conservative operation of the generic model under the actual traffic variance behavior. Then, the *online RL* module is updated with new model f_0 and scaled factor ρ_1 and Phase I ends (lines 6–7).

In case the current derivative is still high, the threshold-based capacity allocation procedure continues. Here, factor k in eq. (11) is adapted from its input value as soon as more traffic data are available and traffic variance is better estimated. With the estimated variance and target load l^*, factor $k(t)$ is computed. Then, k is updated in two different ways: (*i*) reducing by half between k and $k(t)$ if k is larger than needed; or (*ii*) replaced by $k(t)$ if is k is lower than needed (lines 8–10). The flow manager is requested to modify the flow capacity to the computed $z(t)$, which is bounded by the maximum capacity z_{max} (line 11) and Phase I continues (line 12).

Algorithm 15 details the procedure during Phase II. Traffic data collected from the *flow manager* are sent to the *offline learner* block for updating a historical traffic database used to train RL models offline in the sandbox (line 1 in Algorithm 15). The offline model training procedure runs in parallel in the sandbox domain. When enough traffic measurements are collected and processed and an accurate and robust offline-trained RL model is available, it is sent to the *online RL* block to be used for flow capacity operation; this ends Phase II (lines 2–5). Otherwise, Phase II continues, aiming to identify whether the RL model currently in operation needs some parameter tuning (in addition to model updates that the RL algorithm performs during operation). In particular, this procedure aims to supervise the degree of QoS assurance (as compared with the target value, *qa*) obtained by the current model and modifying factor ρ when needed to achieve the target performance. Note that low overprovisioning is the secondary objective and therefore, QoS assurance analysis requires computing whether the current capacity violated maximum delay ($o(t) < 0$) or not ($o(t) \geq 0$). The result is stored in *DB* (lines 6–10).

Algorithm 15. ModelSelectionAndTuning() (Phase II)

```
INPUT: $t$, $x(t)$                          OUTPUT: changePhase
1:  offlrn.updateTrafficDB($x(t)$)
2:  if offlineLearn.newModelAvailable() then
3:     <$f$, $\rho$>← offlineLearn.getModel()
4:     onlineRL.setModel($f$, $\rho$)
5:     return True
6:  $x_{max}(t)$=max($x$)
7:  $z(t-1)$← flowMgr.getCurrentCapacity($t$)
8:  $o(t)$←computeSlackSurplus($x_{max}(t)$, $z(t-1)$) // eq. (13)
9:  if $o(t)<0$ then DB.QA.append(0)
10: else DB.QA.append(1)
11: $p_{obs}$←avg(DB.QA)
12: pval_l←BinomialTest1("$p_{obs}<qa$")
13: pval_g←BinomialTest2("$p_{obs}>qa$")
14: if min(pval_g, pval_l)>cfl then
15:    DB.QA←∅
16:    if pval_l<cfl then onlineRL.tuneParam('$\rho$', $\Delta\rho$)
17:    else onlineRL.tuneParam('$\rho$', -$\Delta\rho$)
18: return False
```

Next, two different one-sample proportion binomial hypothesis tests are conducted to detect whether the observed degree of QoS assurance is significantly below (test 1) or above (test 2) the target value *qa* (lines 11–13). In the case that some of the hypotheses can be confirmed (on the contrary, it is assumed that QoS assurance is in the target), ρ needs to be tuned. If hypothesis test 1 is confirmed, some extra capacity allocation is needed, which is achieved by increasing ρ

with a given step size *Δρ*. On the contrary, if hypothesis test 2 is confirmed, allocated capacity can be reduced, so *ρ* is decreased by the same step size.

Algorithm 16. specificModel() (Phase III)

```
INPUT: t, x(t)                OUTPUT: changePhase
1: $rw(t) \leftarrow onlineRL$.getReward($t$)
2: $DB$.traffic.append($x(t)$)
3: $DB$.reward.append($r(t)$)
4: if |DB.traffic|>m then DB.traffic.pop(0)
5: if |DB.reward|>m then DB.reward.pop(0)
6: V←varianceAnalysis(DB)
7: return V.var NOT IN [var_l, var_h] OR rw(t) < rw_l
```

Finally, Algorithm 16 describes the procedure running in the Analyzer during Phase III. This algorithm analyzes the last m traffic measurements and the reward obtained by the *online RL* (lines 1–6 in Algorithm 16). The objective of this analysis is to check whether both the current traffic and reward follow the expected behavior (line 7). Let us assume that an extended estimation of the working variance with range [*var_l*, *var_h*] is found during the offline training phase—with a minimum and maximum variance that the RL model can support without losing either robustness or desired performance. Bear in mind that operating a traffic flow with more variance than what is supported by the model can lead to poor QoS assurance and even traffic loss. On the contrary, a traffic flow with less variance can produce large overprovisioning, which the *online RL* can hardly decrease with its fine adaption configuration. In fact, *online RL* continuously adapts the model to smooth traffic changes with controlled reward fluctuations, so that a minimum reward (*rw_l*) can be considered as the reasonable limit of a normal RL operation. Therefore, Phase II is triggered back when traffic variance leaves the working range of the RL model or the observed reward goes below that limit; otherwise, Phase III continues.

6.3.4 Illustrative Results

For the ongoing evaluation, a Python-based simulator reproducing the modules described in Figure 68 was implemented. Realistic traffic flow behavior was accurately emulated using a simulator based on the CURSA-SQ engine, which combines statistically based traffic flow generation and continuous G/G/1/k queue model based on the logistic function; multiple queuing systems can be numerically analyzed and related statistics, such as traffic magnitude, queuing delay, and packet loss, be computed. In the context of this work, CURSA-SQ is used as: (*i*) a lightweight network simulator to emulate a flow manager and the forwarding plane; and (*ii*) a flow simulator running in the ML sandbox domain for offline RL training purposes. It is worth highlighting that both CURSA-SQ instances have been independently configured and managed in order to reproduce the actual separation between the physical network and the sandbox domain.

Traffic was randomly generated according to different *traffic configurations*. Each traffic configuration is the combination of traffic pattern and variance. Two different daily patterns were considered: a simple *sinusoidal* pattern for offline training purposes, and a *realistic* pattern to emulate the real traffic in the forwarding plane. In both cases, traffic fluctuates between 5 Gb/s (valley) and 40 Gb/s (peak) throughout the day. Regarding variance, it is defined as a percentage of the mean, so the magnitude of traffic oscillations changes in time (heteroscedasticity). For the sake of a wider analysis, we considered five different variance values: 1%, 3%, 6%, 12%, and 25%.

RL algorithms running in the RL-based operation and offline learning modules have been implemented in Python3 using libraries such as *pytorch*. A general epsilon decay strategy was implemented in all the RL methods for balancing between exploration and exploitation, with decay factor equal to 0.00125. Moreover, a discount factor equal to 0.95 was set up. Q-learning was configured with $n_s = 14$ states and $n_a = 3$ actions, as well as capacity allocation granularity b = 1 Gb/s. In the case of D3QN and TD3, every DNN consisted of two hidden layers with 100 neurons each implementing the Rectified Linear Unit activation function. All DNNs were trained by means of the Adam replacement optimizer with learning rate equal to 0.001 and maximum replay buffer equal to 1e6 samples.

Finally, the capacity and QoS parameters for the flow under study are maximum capacity z_{max} = 100 Gb/s, optimal load l^* = 80%, and QoS assurance qa = 99%.

In the next two subsections, we first focus on comparing the different RL methods for the scenario where the pure online learning RL-based operation is performed (see Figure 67a). Next, we evaluate the offline leaning + online RL-based operation (Figure 67b), including the three proposed phases.

Online RL-Based Operation

Let us first analyze the reliability of the RL operation under real traffic; we focus specifically on the traffic loss. For this study, low (1%) and high (25%) variances were considered. Figure 73 plots the traffic loss as a function of time from the path set-up time. We observe extremely poor performance (high loss) at the beginning of operation, as it was anticipated in Figure 67a. Interestingly, we observe that the simplest Q-learning method provides the fastest convergence time to achieve zero loss, although it needs more than one day to achieve zero loss operation when traffic variance is high. Note that D3QN is the most sensitive to traffic configuration (zero loss operation time increases three times from low to high variance). TD3 is the method with the slowest convergence (around 4 days).

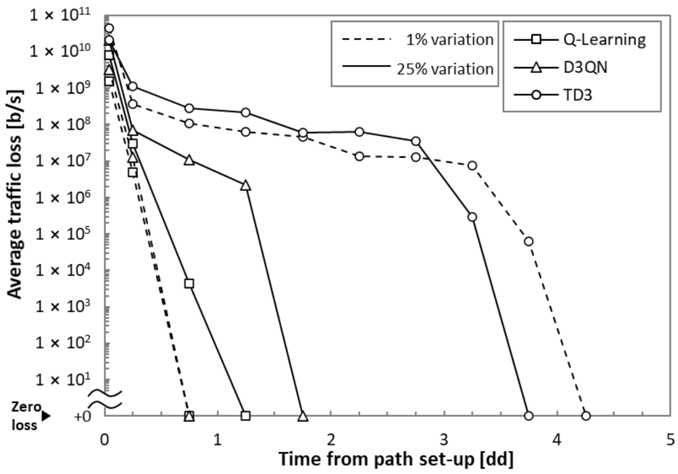


Figure 73. Achieving zero loss operation.

As all the RL methods have achieved zero loss operation, the rest of the results analyze the performance after 5 days of operation. As an illustrative example of the RL-based operation, Figure 74 shows one day of real traffic $x(t)$ and variance from low to high, as well as the capacity $z(t)$ allocated using Q-learning; optimal ρ for each variance is configured. The optimal capacity allocation ($o(t)$ = 0) and the margin for overprovisioning $y(t)$ are also plotted. We observe that the allocated capacity is close to the optimal one, absorbing fluctuations with enough margin to meet the target QoS.

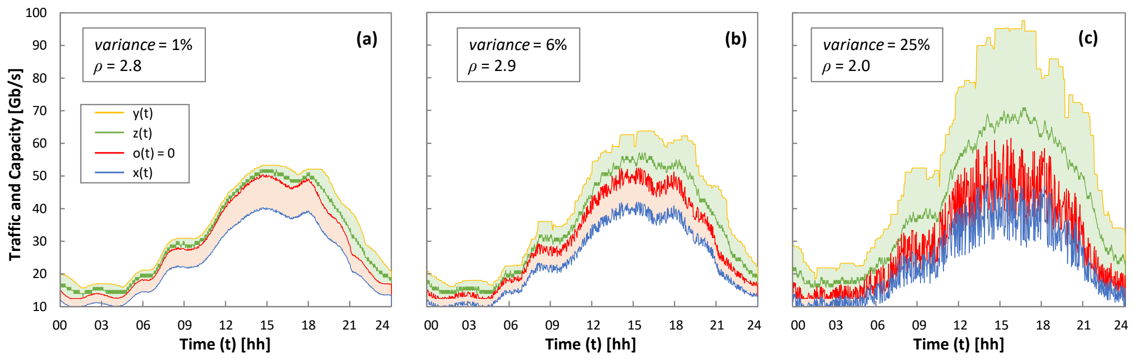


Figure 74. Q-Learning operation. Traffic and allocated capacity for low (a), moderated (b), and high (c) traffic variance.

Let us now analyze the impact of the margin multiplier ρ to achieve the desired QoS. Figure 75 shows the obtained QoS assurance as a function of ρ. For the sake of a comprehensive study, all traffic configurations for sinusoidal (Figure 75a–c) and real (Figure 75d–f) traffic patterns have been analyzed. The minimum ρ value has been set to 1. The ρ values for which the target QoS of 99% is achieved are highlighted with a round marker. We observe in the results that ρ depends not only on the traffic characteristics, but also on the RL method.

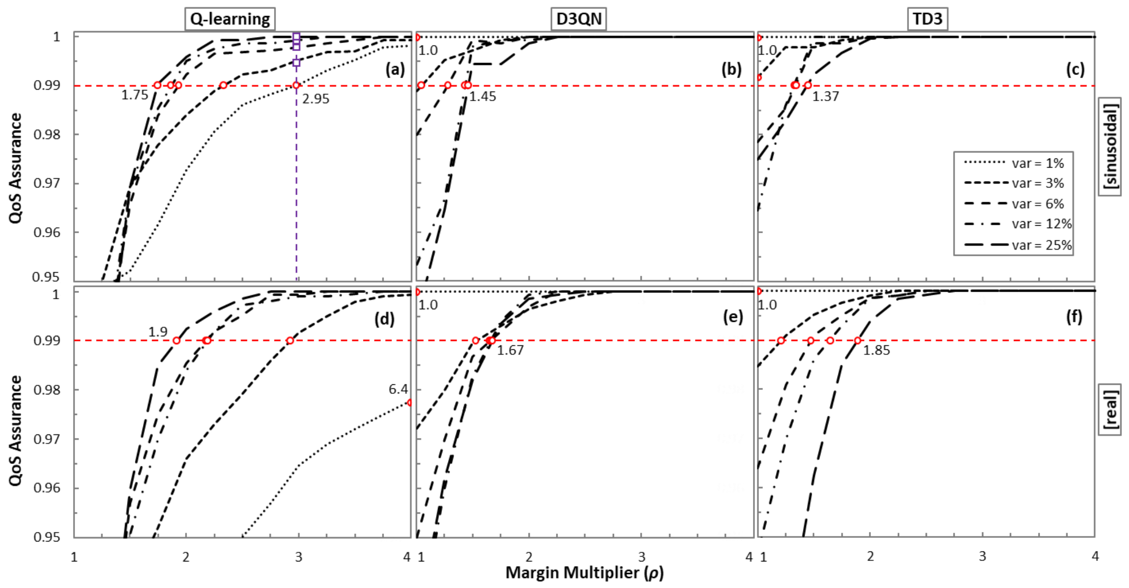


Figure 75. QoS as a function of ρ models trained with a sinusoidal traffic pattern (a–c) and real traffic (d–f).

Interestingly, Q-learning needs the widest range of values for all traffic configurations, requiring a smaller ρ as soon as variation increases. It is worth noting that the large range of values ([1.9, 6.4]) makes it more difficult to adjust ρ for different traffic configurations. Conversely, D3QN and TD3 show a smaller ρ range and opposite behavior as ρ increases with the traffic variance.

The detailed evolution of the optimal ρ with respect to the traffic variation is plotted in Figure 76a, and Figure 76b shows the total overprovisioning introduced by every RL method operating with the optimal ρ. We observe that overprovisioning increases with traffic variation, with a slightly different trend depending on the traffic pattern (sinusoidal and real). Moreover, every RL method introduces different amounts of overprovisioning as shown in Figure 77, where the relative overprovisioning per RL method with respect to the minimum one for every traffic configuration is represented. Q-learning is the method that requires larger overprovisioning in general terms, whereas D3QN and TD3 show better performance. Interestingly, the differences are proportionally larger when traffic variation is small.

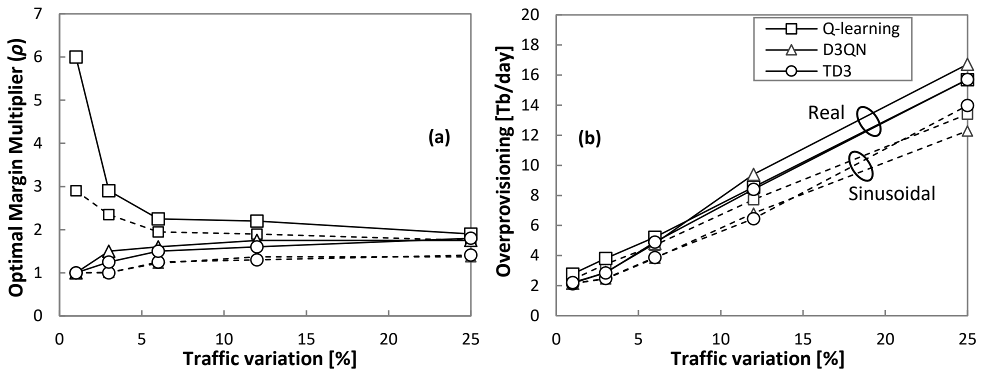


Figure 76. Optimal margin multiplier (a) and overprovisioning (b).

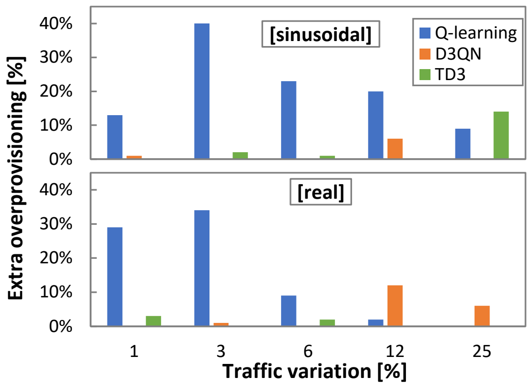


Figure 77. Relative extra overprovisioning.

Let us analyze the effect in terms of extra-overprovisioning when ρ is fixed to a constant value, high enough to assure reliable QoS performance in online RL operation under traffics with a wide range of characteristics. The squared purple markers in Figure 75a indicate the QoS that could be achieved under different traffic characteristics for the largest ρ. Table 19 summarizes the relative and absolute increment of overprovisioning (computed in total Tb per day of operation) produced with the most conservative ρ configuration for all traffic configurations and RL methods. When the fixed ρ happens to be the optimal one for the traffic and RL method, no additional overprovisioning is set up (values in boldface); otherwise, additional overprovisioning is introduced. Q-learning is the method that adds the largest extra overprovisioning (exceeding 200% and 30 Tb/day). In any case, it is worth highlighting that achieving optimal performance in terms of QoS assurance while achieving efficient capacity allocation requires some method to find the optimal ρ for the traffic characteristics.

To conclude this section, we can highlight (*i*) that the CRUX problem proposed in Section 2 can be tackled using RL; under different traffic characteristics and RL methods, the target QoS is assured with reduced overprovisioning; (*ii*) online RL operation leads to traffic loss at the beginning of flow capacity operation; this fact prevents us from using RL until online learning has come up with a robust and reliable model to guarantee autonomous flow operation; and (*iii*) the comparison of the RL methods shows interesting differences among them. While Q-learning learns fast, it also produces larger overprovisioning. D3QN and TD3 need more time to ensure zero loss and adjust QoS at the benefit of reducing overprovisioning.

Table 19. Additional overprovisioning when fixing ρ conservatively.

Traffic Pattern	Traffic Variance	Q-Learning		D3QN		TD3	
		%	Tb/day	%	Tb/Day	%	Tb/Day
	1%	0.00%	0	0.88%	0.02	2.49%	0.05
	3%	19.18%	0.67	6.04%	0.16	20.67%	0.51
Sinusoidal	6%	38.71%	1.85	5.21%	0.21	15.38%	0.63
	12%	44.15%	3.56	0.00%	0.00	4.01%	0.32
	25%	60.40%	8.29	0.00%	0.00	0.00%	0.00
	1%	0.00%	0	0.05%	0.00	4.95%	0.11
	3%	106.68%	3.94	9.52%	0.28	0.39%	0.01
Real	6%	192.50%	9.52	12.72%	0.60	2.90%	0.14
	12%	200.69%	18.37	0.00%	0.00	0.00%	0.00
	25%	219.35%	34.46	0.00%	0.00	0.00%	0.00

Offline Leaning + Online RL-Based Operation

Let us now focus on evaluating the operational approach detailed in Figure 67b, which consists of three phases. Before emulating flow operation, we generated synthetic data for all the traffic variances following the sinusoidal traffic pattern and used them to pre-train generic models independently for each traffic configuration and RL method. We ran every offline RL training for 14,400 episodes to guarantee QoS assurance with minimum overprovisioning.

Starting with the analysis of Phase I, Figure 78a shows an illustrative example of the evolution of the capacity during the operation of the threshold-based algorithm (Algorithm 14) for the real traffic pattern and variation 12%. Actual traffic *x*(*t*), allocated capacity *z*(*t*), and the evolution of the self-tuned *k* parameter are also shown. Note that *k* quickly evolves from its initial value (*k* = 4) to reach a capacity closer to actual traffic. However, as soon as the load exceeds the optimal one *l**, *k* is increased until reaching a stable value (1.47), which happens after 60 minutes of operation. The inset table in Figure 78a details the values of *k* after one hour of operation for all traffic configurations. It is worth noting that the self-tuned threshold-based algorithm operates with zero traffic loss for all the cases.

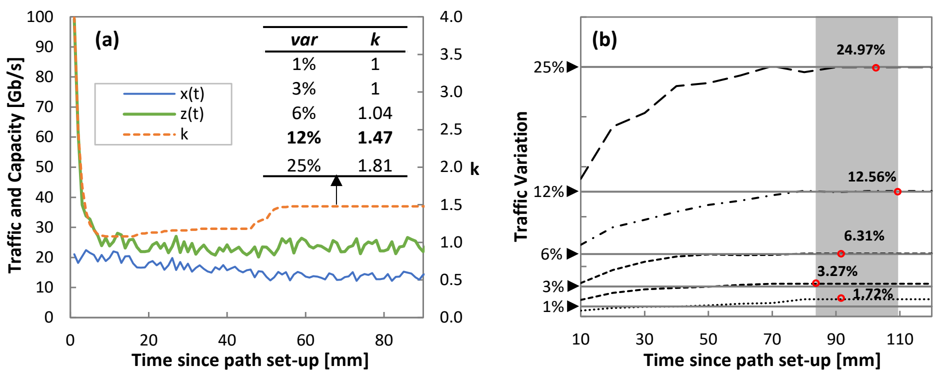


Figure 78. Phase I: Self-tuned threshold (a) and traffic variance analysis (b).

Parallel to the threshold-based operation, traffic variance analysis (Algorithm 12) is conducted in order to compute the true variance of the traffic. Figure 78b shows the computed traffic variation as a function of time for all traffic configurations. Round markers highlight when the derivative of traffic variance reached a small *eps* = 0.01 and labels show the final computed variance value. The low error between computed and true variances is noticeable. Such estimation is achieved within two hours in all the cases.

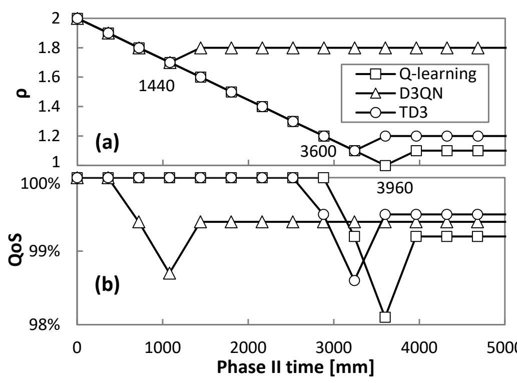


Figure 79. Phase II: QoS (a) and ρ (b) evolution.

After two hours of operation, *Phase II* (Algorithm 15) can start and an RL method with a generic pre-trained model for the true traffic variance is set to operate. The tuning of parameter *ρ* (*Δρ* = 0.1) and the resulting QoS are shown in Figure 79a and Figure 79b, respectively, for traffic variance equal to 12%. To detect whether the measured QoS is considerably below or above the desired *qa* value, a significance level *cfl* = 0.05 was used to be compared against the obtained *p*-value from the binomial tests. We observe that *ρ* decreases up to a magnitude that produces QoS below 99%; just after that, *ρ* increases and remains stable from that point on. As shown in Figure 79a, the time to converge to the best *ρ* is 3960, 1440, and 3600 min (2.75, 1, and 2.5 days) for Q-learning, D3QN, and TD3, respectively.

The above analysis, however, needs to be complemented with the overprovisioning to extract meaningful conclusions. Figure 80 presents the overprovisioning obtained by every RL method before and after tuning *ρ* in *Phase II*. For reference purposes, the overprovisioning introduced by the threshold-based algorithm during *Phase I* is also included as a dotted line. The large benefits in terms of overprovisioning reduction for the RL-based operation with regard to the threshold-based algorithm are remarkable—up to 45% of capacity allocation reduction and 11 Tb/day of total capacity savings for one single flow. After *ρ* tuning, D3QN shows the worst performance, as Q-learning and TD3 achieved significantly lower overprovisioning (24%, ~3 Tb/day). Figure 80 also shows the obtained overprovisioning when the specific model (trained offline with the collected traffic) is loaded in *Phase III* after 10 days of operation. We observe that Q-learning and TD3 reduce overprovisioning slightly, whereas a larger reduction is achieved with D3QN; we conclude that the former RL methods are less dependent on an accurate model of the specific traffic to achieve optimal capacity allocation.

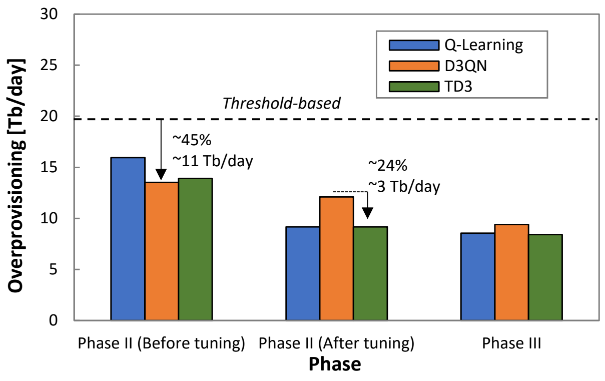


Figure 80. Overprovisioning reduction.

Finally, let us analyze the performance of Algorithm 16 to detect traffic changes while flow is operated in *Phase III*; recall that such detection immediately triggers *Phase II.* To this end, we generated four different scenarios, combining two different types of changes in traffic variance while keeping traffic profile unchanged. We evaluate *gradual* and *sudden/increase* or *decrease* traffic variance changes. Figure 81 illustrates two out of four scenarios: *gradual increase* (variance gradually increases from 1% to 25% along 5 days) and *sudden increasing* (from 1% to 25% in just one minute); an inverse trend is configured for *gradual* and *sudden decrease* scenarios.

Table 20 details the time to detect the traffic change when the variance range was configured as [*var_l*, *var_h*] = [−10%, +10%]; the current traffic variance and minimum reward *rw_l* was set to 5% of the minimum observed reward (see Algorithm 16). We observe that the proposed mechanism ensures prompt reaction under any of the studied changes—immediate detection is achieved when a sudden change happens, and no more than one hour is required for gradual change detection.

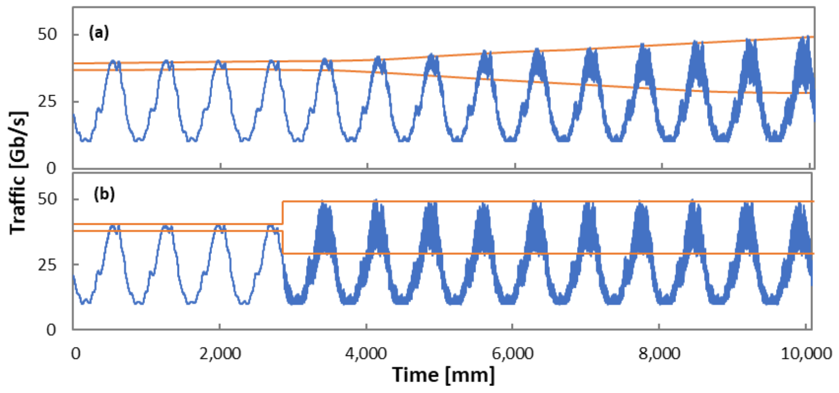


Figure 81. Phase III: Traffic variance change scenarios. Gradual increase (a) and sudden increase (b).

Table 20. Phase III: Analysis under traffic changes.

Traffic Change Scenario	Detection Time (min)	QoS at Detection Time (%)			Reward Degradation (Min from Detection)		
		Q-L	D3QN	TD3	Q-L	D3QN	TD3
Gradual increase	45	99.30	100	100	419	595	585
Gradual decrease	650	99.86	99.44	99.72	3354	2440	2233
Step increase	1	99.17	99.86	99.86	332	413	433
Step decrease	1	99.03	99.44	99.44	494	683	212

To evaluate the promptness of detection, Table 20 considers the observed QoS at the detection time, as well as the elapsed time between detection time and the time when reward begins to degrade (reveals whether the RL module is working properly). Note that the detection happens when the QoS is still above the target value in all the cases. This is proof of anticipation of the change detection, which is key to guarantee robust and reliable RL-based operation.

6.3.5 Concluding Remarks

The Flow Capacity Autonomous Operation (CRUX) problem has been introduced to deal with online capacity allocation of traffic flows subject to dynamic traffic changes; it guarantees precise QoS requirement assurance and minimizes capacity overprovisioning. RL-based operation was proposed to learn the best policy for the traffic characteristics and QoS requirements of a given flow. RL allows adaptive and proactive capacity allocation once the optimal policy is learnt. However, pure RL operation lacks robustness during online learning (e.g., at the beginning of flow operation and in the event of traffic changes) and might result in undesirable traffic loss. However, this can be avoided using simpler reactive threshold-based capacity allocation.

In view of the above, an offline + online learning lifecycle was proposed, aiming at providing guaranteed performance during the entire lifetime of the traffic flow. The proposed management lifecycle consists of three phases. Firstly, a self-tuned threshold-based approach was proposed to operate just after the flow is set up and until enough evidence of the traffic characteristics are available (*Phase I*). Secondly, an RL operation based on models with a pre-trained generic traffic profile but meeting specific traffic variance that was measured during Phase I was executed (*Phase II*). Lastly, an RL operation with models trained for the real measured traffic, while allowing an online RL to adapt to smooth traffic changes (*Phase III*). In addition, during *Phase III* online traffic analysis and RL performance tracking was conducted to detect sharper traffic changes that might require moving back to *Phase II* to keep high reliability.

The proposed lifecycle was implemented under three different RL models, namely, Q-learning, D3QN, and TD3. While Q-learning allows for simple and easy-to-learn definition of policies under discrete spaces of states and actions, D3QN and TD3 enable the application of more complex policies based on deep learning models with continuous state space (D3QN) and continuous action space (TD3).

Numerical evaluation of the proposed offline + online lifecycle under different RL techniques was carried out, reproducing realistic traffic flows in a simulation environment. For benchmarking purposes, comparative results against basic threshold-based operation and online RL operation were also presented. The main conclusions extracted from the numerical evaluation are summarized in Table 21, where colors are used to highlight the results. As expected, online RL produces moderate to high loss (reaching peaks of 1–10 Gb/s) at the beginning of the network operation. Among the different methods, Q-learning reached the required QoS operation earlier (up to 6 times faster than TD3) at the expense of moderate to large overprovisioning (up to 40% larger than TD3). On the other hand, D3QN and TD3 needed more time to converge to the required QoS operation but resulted in considerably better capacity allocation efficiency.

Table 21. Summary of results for policy-based and RL operation with and without offline learning.

Approach	Concept		Threshold-based	Q-Learning	D3QN	TD3
Policy-based	Traffic loss		Zero traffic loss			
	QoS assurance		Since path set-up			
	Over-provisioning		Very large			
Online RL Operation	Traffic loss			Moderated loss	Moderated loss	High loss
	QoS assurance			After 2 days	After 2 days	After 5 days
	ρ range for QoS assurance			Wide	Narrow	Narrow
	Over-provision	conservative ρ		Large	Small	Small
		optimal ρ		Moderate	Small	Small
Offline + Online RL Operation	Traffic loss		Zero traffic loss	Zero traffic loss	Zero traffic loss	Zero traffic loss
	QoS assurance		Since path set-up	Since path set-up	Since path set-up	Since path set-up
	ρ fine tuning effectiveness			Large	Moderated	Large
	Over-provision Gain	Phase I	None			
		Phase I-> Phase II		Moderated	Large	Large
		Phase II		Large	Small	Large
		Phase II-> Phase III		Small	Large	Small
	Reliability (Phase III-> Phase II)			High	High	High

The analysis of the numerical results of the proposed lifecycle leads to several conclusions. Firstly, zero traffic loss and QoS assurance is guaranteed from path set-up regardless of the chosen RL method. Secondly, *Phase II* allows a very efficient and robust operation based on pre-trained generic models that were tuned with specific traffic characteristics. *Phase II* clearly outperformed threshold-based operation in terms of capacity utilization since it remarkably reduced overprovisioning (up to 45%). Thirdly, all the methods reached outstanding capacity efficiency (more than 50% of capacity reduction with respect to threshold-based operation) without losing QoS performance in *Phase III*; Q-learning and TD3 behaved slightly better than D3QN. Finally, the continuous analysis and tracking conducted during *Phase III* to detect traffic changes allowed a prompt detection of sharp changes (between 1 and 650 minutes), triggering *Phase II* from several hours to days before online RL operation suffered any significant degradation.

6.4 Dynamic Control of P2MP connectivity

P2MP connectivity, when supported by DSCM, leads to cost reduction in the presence of large dynamic traffic scenarios if not all SCs need to be active when traffic is low. To dynamically allocate SCs based on the traffic observed at the individual Txs in the leaves of a P2MP connection a centralized module can be deployed in the SDN controller. Such approach can provide gains in the maximum number of leaves that can be supported. However, that centralized approach shows a number of drawbacks: i) it is still heavily reliant on the SDN controller to communicate with the various Txs; ii) it intakes large amounts of observational data, and iii) it requires to process the requests on a synchronous basis, and all near real-time to follow traffic dynamics.

In this section, we distribute decision making down to the very transponders participating in the P2MP connection and relieve the SDN controller from near real-time operation, hence increasing scalability. To this end, we introduce agents with the ability to communicate among them directly to create a MAS. The target is to achieve gains similar to those provided by the centralized approach.

6.4.1 MAS-based Subcarrier Allocation

To show the main difficulty of moving decision making to the transponders, we focus on the direction from the leaves to the hub (MP2P). Here, some sort of coordination is needed to avoid two Txs using the same SC. Figure 82a illustrates the target scenario. Every Tx is tuned on the portion of the spectrum assigned (dotted lines), within which its SCs are allocated.

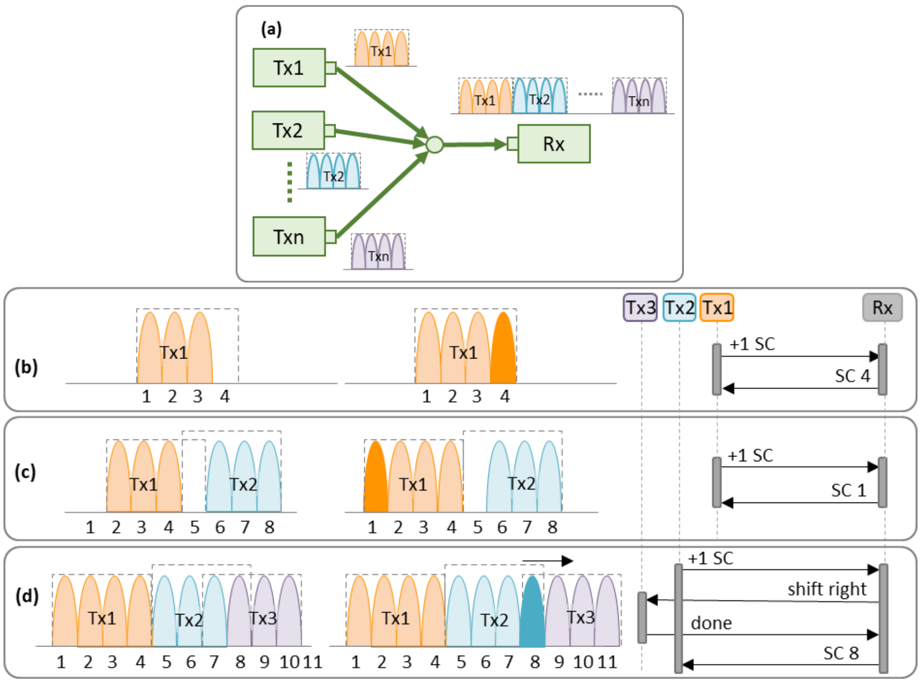


Figure 82: MP2P connectivity based on DSCM (a). SC allocation upon Tx1 request (b) and (c). Neighbor shifting (d).

We define a normative MAS with two types of agents for the Txs and the Rx. There are various interpretations of normative MAS from which we adopt the definition where the system is governed by restrictions on patterns of behaviors of the agents in the system. In the proposed MAS, Tx and Rx agents have distinctive norms that govern their behaviors. The proposed algorithms rely on agent sociability, where agents can share knowledge to achieve their goals; hence, communication is an important aspect of agents' functionality. We define communication channels capable of sending and receiving information between each Tx and the Rx, while avoiding Txs communicating with one another.

The Tx agent is responsible for allocating enough capacity for the incoming traffic. Traffic prediction is used to anticipate traffic dynamicity, and required capacity changes (activation or deactivation of SCs) are requested to the Rx agent. The role of the Rx agent is to mitigate SC oversubscription, which can occur when multiple Txs request the activation of a single SC. As for the Rx agent, we consider two main functionalities, used in combination: i) a *simple request process*, where a Tx agent makes a request and the Rx evaluates the spectrum in order to accept or deny this request. This is shown in Figure 82b, where Tx1 sends a request for an increase in capacity. The Rx replies with instructions for the Tx to occupy SC 4. Another example is shown in Figure 82c, where the Rx is aware of a possible oversubscription of SC 5 and commands Tx1 to use SC 1; and ii) a *neighboring Tx shifting* in order to satisfy a Tx's request. For example, in Figure 82d, Tx2 requests additional capacity, but there is no free spectrum since SCs 5 and 8 are already allocated. In that case, the Rx can ask the neighbor Tx3 to shift the SCs right and liberate SC 8, so that Tx2 can allocate it.

6.4.2 Architecture and Agent Models

In the previous section, Tx and Rx agents played several roles, from capacity and SC management to communication. However, the main role of MAS agents is to communicate with one another to coordinate SC utilization in the MP2P connection and avoid oversubscriptions. The MAS Tx agent also interacts with the local *capacity manager* which manages the capacity available between the local Tx and the Rx to follow traffic dynamicity. Tx and Rx MAS agents also interact with the *transponder agent* responsible for SC operation, such as activation and deactivation to meet capacity requirements, but it does not make any decision on the use of the spectrum. The spectrum management role is now played by the MAS Rx agent. Figure 83 summarizes the relationship among the different agents in the system and includes the main commands that they exchange.

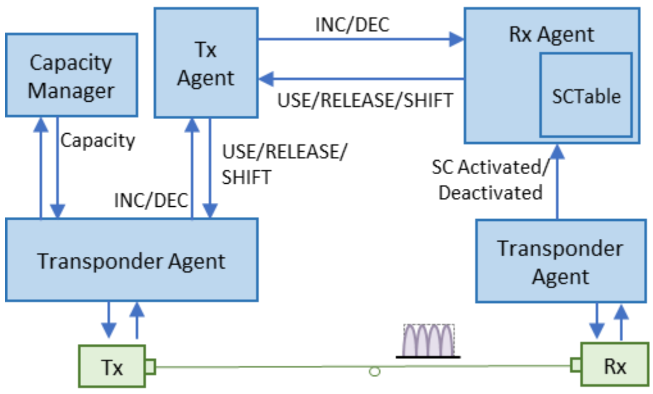


Figure 83: Relation between MAS and transponder agents

The MAS Tx agent role is simply to translate requests to/from the transponder agent to/from the MAS Rx agent. However, the role of the MAS Rx agent is more complex as it has to coordinate spectrum allocation of the whole MP2P connection. Algorithm 17 describes the Rx agent; it receives a command and the ID of the Tx agent that issued it (we used the value -1 to indicate an invalid ID) and returns the command to be executed by a Tx agent on a specific SC.

Algorithm 17. Rx Agent

```
INPUT: cmd, TxId                OUTPUT: cmd, TxId, SCId
 1: if cmd in {SCActivated, SCDeactivated} then
 2:     <cmd, TxId> ← pendingReq.remove()
 3: if cmd == DEC then
 4:     SCId ←find_best(SCTable, TxId, RELEASE)
 5:     return {RELEASE, TxId, SCId}
 6: if cmd == INC then
 7:     SCId ← find_best(SCTable, TxId, USE)
 8:     if SCId <> -1 then return {USE, TxId, SCId}
 9:     shift=<TxId, dir> ← find_neighborShift (
                  SCTable, TxId, SHIFT)
10:     if not shift then return {USE, TxId, -1}
11:     pendingReq.add(<cmd, TxId>)
12:     return {SHIFT, shift.TxId, shift.SCId}
```

The Rx agent processes messages from Tx agents to request or release a SC and from the local transponder agent when a SC has been actually activated or deactivated. Because operation is asynchronous, a list of requests pending to be processed is maintained. Pending actions are processed back after SCs are actually activated or deactivated (lines 1-2). In addition, the Rx agent maintains an internal table with the status of every SC, which is checked and updated when a request to release a SC or to get a free SC is received from a Tx agent.

When a request to release a SC is received (lines 3-5), the Rx agent finds which SC is the best to be released as a function of the allocation of neighboring Txs. The SC selected is returned to be sent to the requesting Tx agent. The allocation of a new SC entails more complexity (lines 6-12).

If a neighboring SC to the Tx current allocation is free, then it is selected (lines 7-8). Otherwise, a possible spectrum shifting (entailing the activation and deactivation of two SCs) of a neighboring Tx is evaluated (lines 9-12). If a spectrum shifting is possible, it is requested to that Tx agent and the current SC allocation request is added to the pending list.

6.4.3 Centralized Subcarrier Allocation

This section proposes an Integer Linear Programming (ILP) model to dynamically solve SC allocation and reconfiguration. The problem statement is:

Given:

- the total spectrum available for the MP2P connection, represented by ordered set $W = \{w_1, w_2, \ldots, w_{|W|}\}$, where $|W|$ is the maximum number of wavelengths supported by the Rx.
- the set of candidate channels described by set $C = \{c_0, c_1, c_2, \ldots c_{|C|-1}\}$, where c_0 is the empty set and every c_i (i:1..|C|-1) is a subset of contiguous wavelengths of size 1..*m*, being *m* the maximum number of wavelengths supported by each Tx. We assume all Tx with the same characteristics.
- a set of Txs $T=\{t_i, t_2, .., t_n\}$, where every Tx *t* has an associated capacity requirement k_t.

Output: the configuration of every wavelength and its assignment to every Tx, and the channel assigned to every Tx.

Objective: 1) minimize the amount of lost traffic; 2) to minimize the number of used SC, thus minimizing energy cost; and 3) minimize the number of SCs that are reconfigured.

The parameters and decision variables of the problem are:

α_i	Weights of the multi-objective function
k_c	Capacity of channel *c*
δ_{tc}	Equal to 1 if channel *c* is a candidate for transponder *t*, 0 otherwise.
v_{tc}	Total number of active wavelengths *w* in channel *c* for transponder *t*.
r_{tc}	Whole number describing the number of wavelengths *w*, that were modified with respect to c_t.
ct	Current channel selected for transponder t.
utc	Whole number describing the difference between kt and kc.
δcw	Equal to 1 if wavelength w is in use for candidate channel c, 0 otherwise.
stw	Equal to 1 if Tx t current occupies wavelength w; 0 otherwise.
xtc	Binary decision variable, equal to 1 if channel c is assigned to Tx t; 0 otherwise.

In order to ensure fast computation time, the number of possible solutions was limited through a pre-calculation, where the set of channels C is calculated as $\delta_{tc} = 1, \forall t \in T, c \in C \mid overlap(s_{tw}, c_t) \mid\mid c == c_0$ and δ_{cw} can be easily calculated for the channels. Additionally, we compute $u_{tc} = \max(k_t - k_c, 0), \forall t \in T, c \in C \mid \delta_{tc} = 1$ to calculate the traffic that could be lost, $v_{tc} = |c|, \forall t \in T, c \in C$ to track the number of wavelengths in use, and $r_{tc} = |\, index(c_t) - index(c)|, \forall t \in T, c \in C$ to measure the number of SC reconfigurations made for a potential new channel.

The mathematical programming formulation for the problem is as follows, where the objective function (eq. (21)) minimizes the three defined objectives, which are weighted by the α_i parameters, where $\alpha_1 > \alpha_2 > \alpha_3$ to maintain the priority of the objectives. Constraint (22) ensures that only one Tx can be assigned to one eligible channel, and constraint (23) ensures that every wavelength can be occupied with, at most, one and only one SC from one single Tx.

$$\min \sum_{t \in T} \sum_{c \in C} [\propto_1 \cdot u_{tc} + \propto_2 \cdot v_{tc} + \propto_3 \cdot r_{tc}] \cdot x_{tc} \tag{21}$$

subject to:

$$\sum_{c \in C} \delta_{tc} \cdot x_{tc} = 1 \quad \forall t \epsilon T \tag{22}$$

$$\sum_{t \in T} \sum_{c \in C} \delta_{tc} \cdot \delta_{cw} \cdot x_{tc} \leq 1 \quad \forall w \epsilon W \tag{23}$$

6.4.4 Results

In order to evaluate the proposed MAS system, a Python-based simulator was implemented to reproduce the MP2P optical connection in Figure 82 and include the agents and communication channels in Figure 83. Each Tx was equipped with 4 60Gb/s SCs (assuming 16QAM at 11 Gbaud). Besides each Tx, a packet traffic generator was used to inject traffic following a typical daily profile varying between 60 and 240 Gb/s, thus leading to capacity requests between 1 and 4 SCs. On the Rx side, 16 SCs were configured, thus leading to a maximum capacity of 960 Gb/s for the whole MP2P connection.

Two traffic scenarios were considered, namely, *in-phase* and *opposition-phase*, with Txs requiring either a similar or different (respectively) number of SCs at a given time. Figure 84 shows the total offered traffic (average and maximum) as a function of the number of Txs. The in-phase scenario presents a high peak/average ratio (1.7) similar to that of a single Tx traffic. Thus, although 4 Txs produce moderate average traffic, the maximum reaches Rx capacity limit. On the other hand, as a consequence of traffic multiplexing, the opposition-phase scenario presents a much lower peak/average ratio (1.1), reaching maximum Rx capacity when 6 Tx are considered.

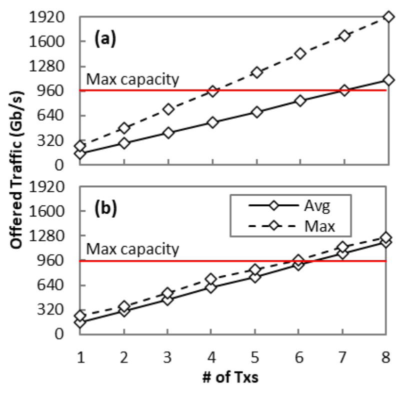


Figure 84: Offered Traffic at Rx

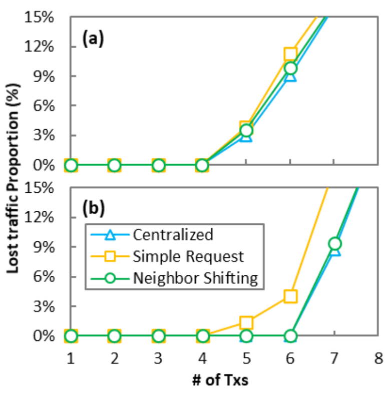


Figure 85: MAS vs Centralized

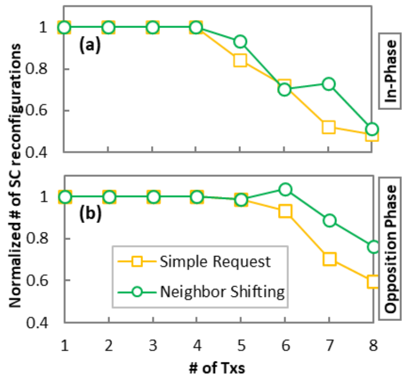


Figure 86: SC reconfigurations

Two different configurations for the MAS Rx agent with increasing functionalities were evaluated: *i*) *Simple Request*, corresponding to the process illustrated in Figure 82b-c and defined by lines 1 to 8 in Algorithm 17; and *ii*) *Neighbor Shifting*, adding the process illustrated in Figure 82d to the previous functionalities and defined by the whole Algorithm 17. Figure 85 shows the performance of the MAS Rx agent configurations as a function of the number of Txs. For benchmarking purposes, the synchronous centralized approach based on the optimization model was implemented and executed every minute with monitoring data received from Txs. In light of Figure 85a, we can conclude that, under the in-phase scenario, both MAS configurations reached a similar performance to the centralized one, accepting 4 Tx without loss and 5 Txs with moderated loss ~3%. However, under the opposition-phase scenario (Figure 85b), the maximum of 6 Tx without loss is achieved by both centralized and MAS with neighbor shifting.

Complementing the previous results, Figure 86 shows the average number of SC reconfigurations per Tx for both traffic scenarios. Values are normalized to the number of reconfigurations with only one Tx. We observe that both methods perform the same number of SC reconfigurations when the spectrum is not at saturation; when the spectrum is near saturation, the MAS with neighbor shifting performs more reconfigurations to accommodate Tx requests.

Finally, Table 22 focuses on MAS with neighbor shifting and summarizes the average number of messages exchanged between a Tx and the Rx at a given time, as well as the percentage of those messages belonging to the neighbor shifting process. We observe that a moderated number of extra messages (up to 24% w.r.t that of the simple request process) are enough to eliminate loss and achieve near-optimal performance.

Table 22. Message Exchange Analysis

# Tx	**In-phase**		**Opposition**	
	avg(msg/Tx)	***% shifting***	***avg(msg/Tx)***	***% shifting***
4	1.67	0%	1.78	0%
5	2.02	17%	1.86	6%
6	1.72	2%	2.35	24%

6.4.5 Conclusions

We showed that dynamic SC allocation brings significant capital and operational cost reduction in P2MP connectivity, as compared to the static SC allocation. A MAS system for near real-time optical SC allocation has been presented. The distributed MAS reaches similar gains to centralized SC management. However, by moving decision making to the transponders in the P2MP connection a much more scalable solution can be created, thus relieving the SDN controller from operation after the provisioning phase.

7 CONCLUSIONS

Different approaches to reduce operation overheads have been shown in this deliverable, targeting: (i) overprovisioning minimization though planning and orchestration; (ii) predictive failure management to minimize operational expenditures; and (iii) simplification of network and service operations, through digital twining solutions and near-real time control of network resources.

Although the assessment of many of the solutions are from simulation and numerical evaluation, experimental implementations and results are also shown, thus demonstrating the feasibility of the autonomous network operation and showing the need for larger efforts to bring such solutions to real network operator infrastructures.

REFERENCES

[Bos14] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," SIGCOMM Comput. Commun. Rev., vol. 44, no. 3, p. 87–95, jul 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890

[AIS23] G. S. Aiswarya, M. Mariah, R. Katragadda and R. Makam, "Control of Self-Driving Cars using Reinforcement Learning," 2023 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2023, pp. 1-6, doi: 10.1109/CONECCT57959.2023.10234763.

[An24] I. Andrenacci, M. Lonardi, P. Ramantanis, E. Awwad, E. Irurozki, S. Clémençon, and S. Almonacil, "A machine-learning-based technique to establish ASE or Kerr impairment dominance in optical transmission," accepted for publication at JOCN, 2024.

[Bac20] J. Bäck et al., "Capex savings enabled by point-to-multipoint coherent pluggable optics using digital subcarrier multiplexing in metro aggregation networks," in 2020 ECOC, (IEEE, 2020), pp. 1–4.

[Bar20] S. Barzegar, M. Ruiz and L. Velasco, "Autonomous Flow Routing for Near Real-Time Quality of Service Assurance," in IEEE Transactions on Network and Service Management.

[Cas23] Castro, A. Napoli, M. Porrega, J. Back, A. Rashidinejad, M. Quagliotti, E. Riccardi, D. Hillerkuss, A. Yekani, F. Masoud, A. Mathur, J. Pedro, B. Spinnler, S. Erkilinc, A. Chase, T. A. Eriksson, and D. Welch, "Scalable filterless coherent point-to-multipoint metro network architecture," J. Opt. Commun. Netw. 15, B53–B66, 2023

[Che21] J. Chen et al., "DRL-QOR: Deep Reinforcement Learning-Based QoS/QoE-Aware Adaptive Online Orchestration in NFV-Enabled Networks," Trans. Netw. Service Manag., vol. 18, no. 2, pp. 1758–1774, jun 2021.

[Cor05] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," Journal of Algorithms, vol. 55, no. 1, pp. 58–75, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196677403001913

[Cug22] F. Cugini, D. Scano, A. Giorgetti, A. Sgambelluri, F. Paolucci, J. J. Vegas Olmos, and P. Castoldi, "Applications of p4-based network programmability in optical networks," in 2022 Optical Fiber Communications Conference and Exhibition (OFC), 2022, pp. 1–3.

[Cug23] F. Cugini, D. Scano, A. Giorgetti, A. Sgambelluri, L. D. Marinis, P. Castoldi, and F. Paolucci, "Telemetry and ai-based security p4 applications for optical networks (invited)," J. Opt. Commun. Netw., vol. 15, no. 1, pp. A1–A10, Jan 2023. [Online]. Available: https://opg.optica.org/jocn/abstract.cfm?URI=jocn-15-1-A1

[Fan99] W. Fang and L. Peterson, "Inter-as traffic patterns and their implications," in Seamless Interconnection for Universal Services. Global Telecommunications Conference. GLOBECOM'99. (Cat. No.99CH37042), vol. 3, 1999, pp. 1859–1868 vol.3.

[Gall22] J. Gallego-Madrid, "Machine learning-based zero-touch network and service management: a survey," Digit. Commun. Networks 8, 105–123 (2022).

[Gar20] A. L. Garcia Navarro, “Packet-Optical Latency-based RL,” https://github.com/alexgaarciia/ PacketOpticalLatencyRL (2023).

[Her19] J. A. Hernandez, R. Sanchez, I. Martin, and D. Larrabeiti, “Meeting the traffic requirements of residential users in the next decade with current ftth standards: How much? how long?” IEEE Communications Magazine, vol. 57, no. 6, pp. 120–125, 2019.

[Her22] C. Hernández-Chulde, R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, "Evaluation of Deep Reinforcement Learning for Restoration in Optical Networks," 2022 Optical Fiber Communications Conference and Exhibition (OFC), San Diego, CA, USA, 2022, pp. 1-3.

[Her22b] C. Hernández-Chulde, R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, "Latency-Aware Routing and Spectrum Assignment with Deep Reinforcement Learning," 2022 18th International Conference on the Design of Reliable Communication Networks (DRCN), Vilanova i la Geltrú, Spain, 2022, pp. 1-4, doi: 10.1109/DRCN53993.2022.9758014.

[Her23] C. Hernandez-Chulde, R. Casellas, R. Martinez, R. Vilalta and R. Munoz, "Experimental evaluation of a latency-aware routing and spectrum assignment mechanism based on deep reinforcement learning," in Journal of Optical Communications and Networking, vol. 15, no. 11, pp. 925-937, November 2023, doi: 10.1364/JOCN.499343.

[Her23b] C. Hernández-Chulde, R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, "DRL for VNF placement in Inter-Data Center Elastic Optical Networks," 2023 Optical Fiber Communications Conference and Exhibition (OFC), San Diego, CA, USA, 2023, pp. 1-3, doi: 10.1364/OFC.2023.Tu3D.7.

[Her23c] C. Hernández-Chulde, R. Casellas, R. Martínez, R. Vilalta and R. Muñoz, "VNF Placement Over Autonomic Elastic Optical Network via Deep Reinforcement Learning," ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 422-427, doi: 10.1109/ICC45041.2023.10278838.

[Hos23] M. M. Hosseini, J. Pedro, A. Napoli, N. Costa, J. E. Prilepsky, and S. K. Turitsyn, “Multi-period planning in metro-aggregation networks exploiting point-to-multipoint coherent transceivers,” J. Opt. Commun. Netw. 15, 155–162 (2023).

[Iac22] O. Iacoboaiea, J. Krolikowski, Z. Ben Houidi, and D. Rossi, ”From Design to Deployment of Zero-touch Deep Reinforcement Learning WLANs,” arXiv:2207.06172, 2022.

[Jan19] B. Jang, M. Kim, G. Harerimana and J. W. Kim, ”Q-Learning Algorithms: A Comprehensive Classification and Applications,” in IEEE Access, vol. 7, pp. 133653-133667, 2019, doi: 10.1109/ACCESS.2019.2941229.

[Jur21] P. Jurkiewicz, G. Rzym, and P. Boryło, “Flow length and size distributions in campus internet traffic,” Computer Communications, vol. 167, pp. 15–30, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366420320223

[Kum22] H. S. Kumhar and V. Kukshal, “A review on reinforcement deep learning in robotics,” 2022 Interdisciplinary Research in Technology and Management (IRTM), Kolkata, India, 2022, pp. 1-8, doi: 10.1109/IRTM54583.2022.9791615.

[Lo21] M. Lonardi, P. Serena, P. Ramantanis, N. Rossi, and S. Musetti, “Kerr Nonlinearity Dominance Diagnostic for Polarization-Dependent Loss Impaired Optical Transmissions,” in 2021 European Conference on Optical Communication (ECOC), 2021-09-18, pp. 1–4. doi: 10.1109/ECOC52684.2021.9605832.

[Mam19] Z. Mammeri "Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches," in IEEE Access, vol. 7, pp. 55916-55950, 2019.

[Mar21] R. Martínez et al., "Autonomous SDN-based global concurrent restoration for high-capacity optical metro networks," in 2021 Optical Fiber Communications Conference and Exhibition (OFC), (2021), pp. 1–3.

[Mol11] S. Molnar and Z. Moczar, "Three-dimensional characterization of internet flows," in 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1–6.

[Mom22] J. Momo Ziazet, B. Jaumard, "Deep Reinforcement Learning for Network Provisioning in Elastic Optical Networks," In Proc. ICC 2022.

[Mus19] F. Musumeci et al., "A tutorial on machine learning for failure management in optical networks," J. Light. Technol. 37, 4125–4139 (2019).

[Nae20] M. Naeem, S. Rizvi, and A. Coronato, "A Gentle Introduction to Reinforcement Learning and its Application in Different Fields," IEEE Access, vol. 8, pp. 209320-209344, Jan. 2020.

[Nam22] H. Namkung, D. Kim, V. Sekar, and P. Steenkiste, "Sketchlib: Enabling efficient sketch-based monitoring on programmable switches," in USENIX NSDI 2022. USENIX, April 2022. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sketchlib-enabling-efficient-sketch-based-monitoring-on-programmable-switches/

[Nap21] A. Napoli et al., "Live network demonstration of point-to-multipoint coherent transmission for 5G mobile transport over existing fiber plant," in 2021 ECOC, (2021), pp. 1–4.

[Nap22] A. Napoli, Z. Stevkovski, J. D. Jiménez, E. J. Zuleta, J. Bäck, J. Pedro, J. Rodriguez, R. Diaz, J. Carrallo, A. Mathur et al., "Enabling router bypass and saving cost using point-to-multipoint transceivers for traffic aggregation," in Optical Fiber Communication Conference, (Optica Publishing Group, 2022), pp. W3F–5.

[Nat20] C. Natalino and P. Monti, "The Optical RL-Gym: An open-source toolkit for applying reinforcement learning in optical networks," in Proc. ICTON 2020.

[Pav22] P. Pavon-Marino et al., "On the benefits of point-to-multipoint coherent optics for multilayer capacity planning in ring networks with varying traffic profiles," J. Opt. Commun. Netw. 14, B30–B44 (2022).

[POI07] Y. Pointurier, F. Heidari, "Reinforcement learning based routing in all-optical networks," In Proc. BROADNETS 2007.

[Se20] P. Serena, C. Lasagni, and A. Bononi, "The Enhanced Gaussian Noise Model Extended to Polarization-Dependent Loss," Journal of Lightwave Technology, vol. 38, no. 20, pp. 5685–5694, 2020-10, doi: 10.1109/JLT.2020.3001722.

[Sga23] Andrea Sgambelluri, Davide Scano, Roberto Morro, Filippo Cugini, Jordi Ortiz, José Manuel Martinez, Emilio Riccardi, Piero Castoldi, Pablo Pavon, and Alessio Giorgetti "Failure recovery in the MANTRA architecture with an IPoWDM SONiC node and 400ZR/ZR+ pluggables", Journal of Optical Communications and Networking (JOCN), Vol. 16, Issue 5, pp. B26-B34, (2024)

[Sko21] Skorin-Kapov et al., "Point-to-multipoint coherent optics for re-thinking the optical transport: case study in 5G optical metro networks," in 2021 ONDM, (IEEE, 2021), pp. 1–4.

[Swe22] N. L. Swenson, "Open XR Concept Introductory White Paper," Tech. rep., Open XR Forum (2022).

[Tac23] T. Tachibana et al, "Metropolitan Area Network Model Design Using Regional

Railways Information for Beyond 5G Research" in IEICE Trans. on Comm., vol E106.B, no. 4, pp. 296-306, 2023.

[Vil20] R. Vilalta et al., "Experimental validation of resource allocation in transport network slicing using the ADRENALINE testbed," Photonic Netw. Commun., vol. 40, no. 2, pp. 82–93, aug 2020.

[Wel21] D. Welch et al., "Point-to-multipoint optical networks using coherent digital subcarriers," J. Light. Technol. 39, 5232–5247 (2021).

[Wel23] D. Welch, A. Napoli, J. Bäck, S. Buggaveeti, C. Castro, A. Chase, X. Chen, V. Dominic, T. Duthel, T. A. Eriksson, S. Erkilinç, P. Evans, C. R. S. Fludger, B. Foo, T. Frost, P. Gavrilovic, S. J. Hand, A. Kakkar, A. Kumpera, V. Lal, R. Maher, F. Marques, F. Masoud, A. Mathur, R. Milano, M. I. Olmedo, M. Olson, D. Pavinski, J. Pedro, A. Rashidinejad, P. Samra, W. Sande, A. Somani, H. Sun, N. Swenson, H.-S. Tsai, A. Yekani, J. Zhang, and M. Ziari, "Digital subcarrier multiplexing: Enabling software-configurable optical networks," J. Light. Technol. 41, 1175–1191 (2023)

[Xu16] Xu et al., "Understanding mobile traffic patterns of large-scale cellular towers in urban environment," IEEE/ACM Transactions on networking 25, 1147–1161 (2016)