 DOI 10.1364/JOCN.493347

Distributed Intelligence for Pervasive Optical Network Telemetry

LUIS VELASCO*, POL GONZÁLEZ, AND MARC RUIZ

Universitat Politècnica de Catalunya, Barcelona, Spain
**Corresponding author: luis.velasco@upc.edu*

Optical network automation and failure management require measuring the status and the performance of the different network devices to anticipate any degradation and ensure the quality of the provided services, i.e., optical connectivity. Such pervasive network telemetry entails collecting large amounts of measurements and events from different sources and with very fine granularity, which given the amount and variety of telemetry sources and the size of each measurement and event, imposes requirements that are hard to be achieved without large investments. In this paper, we analyze the main limitations of telemetry architectures relying exclusively on centralized systems for data analysis and propose an architecture with distributed intelligence. Data aggregation techniques, especially conceived for optical network telemetry, are presented with the objective of reducing data dimensionality. Illustrative results from our experimental telemetry system reveal reduction of three orders of magnitude in terms of total data volume, without introducing significant error and processing delay, and more importantly, helping network automation algorithms to identify meaningful changes in the network status. © 2023 The Authors

1. INTRODUCTION

Telemetry is an important field of research and thus, extensive work can be found on the topic [1] revealing their benefits for optical networking, including network automation and failure management ([2-5]). Note that network digital twins largely rely on telemetry data for accurate modelling and prediction. For instance, optical spectra can be obtained from Optical Spectrum Analyzers (OSA) installed in optical link and be used for the detection of filter failures [6], whereas in-phase and quadrature (IQ) constellations can be measured in the optical transponders and used for anomaly detection [7] and to estimate the distance traversed by the signal or its Bit Error Rate (BER) [8,9] just to mention a few.

As Big Data, optical network telemetry data are a collection of data from many different sources and can be described by means of a number of characteristics [10], known as the *5 V's*, standing the five V for *volume*, *velocity*, *variety*, *veracity*, and *value*. Such characteristics can be seen as different tiers of a pyramid: *i*) at the bottom of the pyramid, *volume* refers to the size and amount of data that needs to be collected and analyzed; *ii*) *velocity* refers to the speed at which data are collected, stored and managed. Volume and velocity together impose requirements that need to be carefully considered, e.g., sometimes it is better to have limited data in real time than lots of data at a low speed; *iii*) *variety* refers to the diversity and range of different data types and data sources; *iv*) *veracity* is related to the quality, accuracy, and trustworthiness of data and data sources and it is the most important factor of all the 5 V's for business success; and *v*) *value*, at the very top of the pyramid, refers to the ability to transform data into useful insight.

Several telemetry architectures have been defined in the literature [11]. In general, telemetry measurements are collected from observation points in network devices and sent to a central system running besides the Software Defined Networking (SDN) controller. This defines a telemetry pipeline with basically two elements: *data collectors* that gather measurements from observation points in devices and send them to a *centralized telemetry system* that stores and processes the received data. That design is based on the principle of collecting from the network and storing as much data as possible, in the hope that they can feed network automation systems, e.g., based on Machine Learning (ML) [12]. For instance, the authors in [13] present use cases showing the tight relation between network telemetry and automation. Their approach is collecting as many measurements as possible and store them in a centralized data lake, where ML algorithms can extract insight from data. The

motivation of such centralized architecture comes from the fact that the analysis of temporary events can be of great importance to anticipate degradations and outages [14].

Other telemetry pipelines have been proposed, like hierarchical ones for scalability purposes. See, e.g., the tutorial in [15] which details the challenges and requirements for optical telemetry and streaming. In addition, the authors in [16] demonstrated a modular telemetry broker for the remote collection and exchange of telemetry data.

In parallel, events generated by applications/platforms (e.g., SDN controllers and management systems) can be used to keep consistency among systems. In fact, an event streaming mechanism is made available as an alternative to traditional notifications. The streaming capability is distinct from Transport API (TAPI) notifications [17] and is designed to better deal with volume and to provide an improved operational approach. In this context, any component of the SDN control plane may act as a source of *event telemetry*, which should be transported and distributed unaltered to other systems in the control and management planes.

As a result of volume, velocity, and variety characteristics of telemetry data, efficient and flexible mechanisms need to be considered to convey measurements and events from network devices and other systems (*producers*) to consumers, e.g., in a central location. In this regard, some works (see, e.g., [18]) have proposed telemetry architectures using the generalized Remote Procedure Call (gRPC), a protocol specifically devised for telemetry, which can reduce data volume with efficient encodings that compress data. However, scalability is an issue that needs to be carefully considered. This is of special importance in disaggregated scenarios [19], where measurements from several devices can be collected although a single interface, e.g., gRPC, can be used to convey all of them to the central location. In this regard, the SONiC open source network operating system [20] defines an architecture to decouple hardware of packet switches from software through abstraction. The architecture includes a Redis database (DB), which is used as high-performance communication system for internal components. SONiC is currently being considered to control packet-optical nodes [21].

Even though gRPC is very efficient to deal with data volume, its flexibility is very limited since exchanged messages need to follow predefined schemas. In view of this, the authors in [22] proposed and experimentally validated Apache Kafka data streaming to exchange telemetry data as simple text messages, which provides the required flexibility to deal with data variety. In addition, Kafka facilitates the integration of different data sources.

In this paper, we extend our previous initial work in [23, 24]. Specifically, the contribution of this paper is two-fold: *i*) a telemetry architecture is proposed to support intelligent data aggregation nearby data collection, thus extending the telemetry pipeline. The main target is to provide a solution that is able to deal with the 5 V's of telemetry data, while providing scalability, efficiency, flexibility, easy integration of different data sources with a variety of measurements and events, and facility for turning data into useful insight that can be used for network automation; and *ii*) three techniques are proposed for reducing the dimensionality of telemetry measurements to deal with volume and velocity. The techniques are specifically designed for the measurements of larger size, i.e., optical spectrum and IQ constellations.

The rest of the paper is organized as follows. In Section 2, we first illustrate with examples the limitation of centralized telemetry architectures and highlight the requirements that need to be considered when designing an optical network telemetry system. Next, in Section 3, we propose our telemetry architecture supporting intelligent data aggregation. In Section 4, we propose three dimensionality reduction techniques: *i*) supervised feature extraction (FeX); *ii*) data compression; and *iii*) data summarization. Section 5 provides illustrative results from an experimental setup of the proposed telemetry architecture that extends [25]. The focus is on demonstrating how the techniques in Section 4 effectively reduce telemetry data dimensionality, while dealing with variety and veracity and maximizing value. Finally, Section 6 draws the main conclusions of this work.

2. MOTIVATION

Let us illustrate each of the 5 V's with an example of the optical core network for a national telecom operator. Let us assume a core mesh network with 50 optical nodes, with average nodal degree of 3. Imagine that each node is connected with any other node in the network through one single optical connection (*lightpath*). Then, the network supports 2,450 unidirectional lightpaths and needs the same number of transmitters (Tx) and receivers (Rx). Finally, let us assume that we can collect telemetry data from Tx, Rx, Optical Amplifiers (OA) in the nodes that compensate for filtering and fiber attenuation (i.e., 300 OAs in total), and from OSAs installed in every optical link in the network (i.e., 150 OSAs in total). For illustrative purposes, Table 1 summarizes the measurements that can be collected from every device and its estimated size. See [13] for a more exhaustive list of measurements that can be collected from real equipment and [26], where the authors analyze metrics for evaluating system performance and mechanisms of various optical impairments.

Volume: Let us assume that the measurements in Table 1 are collected every second. Then, the network described above generates 15.64 terabyte (TB) of data every day, i.e., 5.58 petabytes (PB) every year, that need to be collected, conveyed to the centralized telemetry system, stored in a data lake, and analyzed.

Velocity: Collecting measurements every second imposes additional requirements related to data collection from the devices and its transport to the centralized telemetry system. Starting with the optical devices, those generating measurements of large size (i.e., OSAs and optical receivers)

Table 1. Illustrative Optical Measurements

Device	Measurements	Size (bytes)
Tx	• Laser params, e.g., temperature. • Configuration, e.g., modulation format and symbol rate.	20 20
Rx	• Optical constellation (e.g., 10,000 IQ symbols). • Receiver parameters, e.g., BER, signal to noise ratio, etc.	80,000 40
OA	• Input power, gain, etc.	20
OSA	• Optical spectrum (4.8 THz C-band, resolution 1 GHz).	19,200

need high-speed data interfaces. E.g., OSAs would require 150 kb/s interfaces, while Rx would require 640 kb/s interfaces. These speeds are assuming that measurements are generated as a stream of floating-point numbers, although such measurements, once collected, are usually formatted, e.g., as a JSON object, which increases its size. In the example, every node agent collecting local telemetry data, formatting data and sending them to the centralized telemetry system, would generate around 40 Mb/s, so the centralized system would receive 1.9 Gb/s of telemetry data in total.

Variety: Six structured measurements are defined in Table 1, which consist of tuples of individual magnitudes to vectors of related values. Additionally, also events and other unstructured data from different systems are collected, which require totally different processing. All these different data types need to be processed, analyzed and correlated in real time. For instance, analysis of spectrum measurements from nodes in the route of a lightpath, together with analysis of IQ constellations in the Rx can be used to identify and localize the cause of a sudden increase of the BER measured in the Rx.

Veracity: Right decisions are made with thorough and correct information. Data can only help if it is clean, i.e., it is accurate, error-free, reliable, consistent, bias-free, and complete. Some factors that contaminate data are, among others: *i*) meaningless information that distorts the data; *ii*) outliers that make the dataset to deviate from the normal behavior; *iii*) software vulnerabilities that could enable data hijacking; and *iv*) statistical data that misrepresents a particular network resource.

Value: Telemetry data can bring large benefits for network automation but only if they are converted into useful insight. Operators can capture value from telemetry data by: *i*) reducing network margins; *ii*) automating service provisioning; *iii*) improving resource utilization and reducing operational costs; *iv*) extending the working life of network equipment; *v*) detecting soft-failures before they become hard failures; *vi*) simplifying maintenance by finding root cause of failures and scheduling works; and many others.

Let us challenge some of the previous assumptions aiming at bringing requirements to the telemetry architecture: 1) Different measurements should have different collection periodicity, which can be variable, or even being collected asynchronously. For instance, Tx are configured at connection setup time or upon some event, and the temperature of the laser would not significantly change that fast; 2) In general, it is not useful to store all the measurements when no significant changes happen. However, to determine whether a significant variation in a measurement has happen, some analysis needs to be carried out, and that should be done earlier in the telemetry pipeline, e.g., at the node level, to reduce volume of data being conveyed to the centralized telemetry system; 3) Compression techniques, which can be either lossy or lossless, can be explored to reduce bandwidth requirements; 4) From the two previous issues, telemetry systems should be somehow decentralized. Some processing and data analysis might be needed at the node level. However, such analysis might be orchestrated by some entity running at a centralized level, which can have global network vision; 5) Data veracity should be checked along the telemetry pipeline and it should be discarded from the main pipeline whenever there is evidence that such data is somehow contaminated. For instance, a sample that does not follow statistically last measurements can be either an outlier or an anomaly. However, the detection point can be local, e.g., if it refers to the gain of an amplifier, or conversely, it needs to be in the centralized system, e.g., if it requires correlation with other measures, e.g., in the case of spectrum measurements in the route of a lightpath; 6) Value should be extracted from data as soon as possible in the telemetry pipeline. E.g., we should not wait to detect degradations from the data collected from a network node in the centralized telemetry system, if this can be done directly in the node. However, sometimes, it is necessary to perform correlation among data collected from different network nodes to extract value from data.

In conclusion, to reduce the impact of the 5 V's, intelligence can be applied along the telemetry pipeline, which needs to be extended with new elements where telemetry measurements can be processed.

3. PROPOSED TELEMETRY ARCHITECTURE

In this section, we introduce our proposed unified telemetry architecture supporting both telemetry events, as well as distributed intelligence along the telemetry pipeline for telemetry measurements.

Fig. 1 presents the reference network scenario, where an SDN architecture controls a number of optical nodes, specifically optical transponders (TP) and reconfigurable optical add-drop multiplexers (ROADM) in the data plane. Note that the SDN architecture might include a hierarchy of controllers, including optical line systems and parent SDN controllers [27]. A centralized *telemetry manager* is in charge of receiving, processing, and storing telemetry data in a telemetry DB, which includes two repositories: *i*) the measurements DB is a time-series (TS) DB that stores measurements; and *ii*) the events DB is a free-text search (FT) engine. In addition, telemetry data can be exported to other external systems, e.g., through Kafka. Some data exchange between the SDN control and the telemetry manager is needed, e.g., the telemetry manager needs to access the

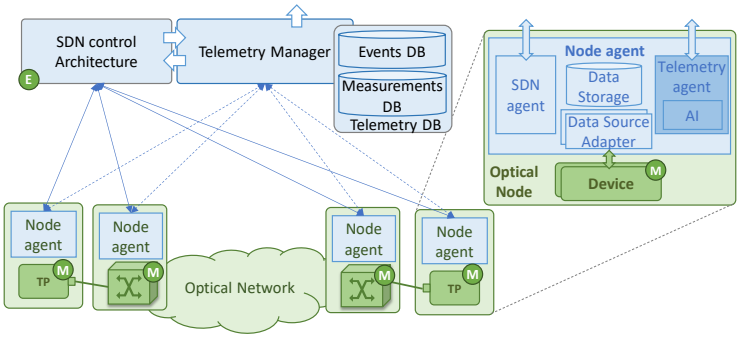

Fig. 1. Overall network and proposed telemetry architecture.

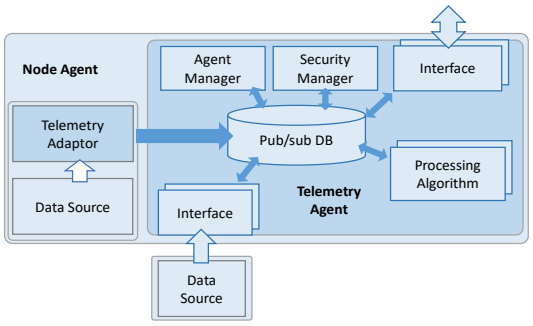

Fig. 2. Telemetry agent architecture

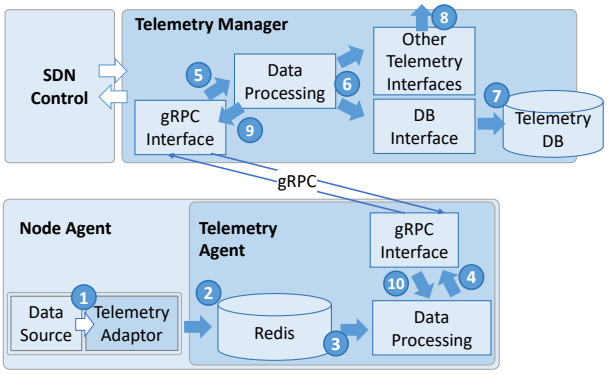

Fig. 3. Illustrative workflow.

topology DB describing the optical network topology, as well as the label-switched path (LSP) DB describing the optical connections (these DBs are not shown in Fig. 1 for the sake of simplicity).

Every node in the data plane is locally managed by a node agent (see some internal details in Fig. 1). The node agent translates the control messages received from the related SDN controller into operations in the local node. In addition, the node agent includes data source adaptors that collect measurements from observation points (labeled M) enabled in the optical nodes or in specific optical devices, like OSAs, as well as a telemetry agent that processes and exports telemetry data to the telemetry manager. In addition, events can be collected from applications and controllers (labeled E).

The internal architecture of telemetry agents inside node agents is presented in Fig. 2, which consists of five main components: *i*) a *manager* module configuring and supervising the operation of the rest of the modules; *ii*) a *security manager* in charge of security aspects, like key management; *iii*) a number of *algorithms* for data processing that include dimensionality reduction and data veracity checking; *iv*) a number of *interfaces*, e.g., gRPC, to communicate with other systems. Additionally, interfaces take care of the security of telemetry data, e.g., to ensure data privacy, authentication, and integrity; and *v*) a *Redis DB* that is used in *publish-subscribe* mode to communicate the different modules among them, i.e., no direct communication is allowed. This facilitates the definition of specific workflows for telemetry data and provides an agile, reliable, and secure environment that simplifies communication, as well as integration of new modules.

Data sources can be integrated in two different ways: *i*) internal data sources, i.e., those that are deployed inside the node agent, can access the Redis DB directly to publish new telemetry data (measurements or events); *ii*) external data sources are connected to the telemetry agent through a dedicated interface (e.g., based on gRPC). Only trusted peers are allowed to connect externally to the telemetry agent. A gRPC interface is used for the telemetry agents to export telemetry to the telemetry manager, as well for the telemetry manager to tune the behavior of algorithms in the agents.

The internal architecture of the telemetry manager is the same that the one for the agents. The difference between them is on the algorithms and the interfaces that they run. E.g., the telemetry manager includes interfaces to the telemetry DBs and to export data to external systems.

Let us describe now a typical telemetry workflow that fits for a wide range of use cases by means of the illustrative example presented in Fig. 3. Data sources gather measurements from the optical devices; the collected raw data is received by the telemetry adaptor (labeled 1 in Fig. 3) that generates a structured JSON object, which is then published in the local Redis DB (2). The periodicity for data collection can be configured within a defined range of values. A number of algorithms can be subscribed to the collected measurements (3). Note that in the case of events, workflows do not include data processing algorithms. In this example, let us assume that only one algorithm is subscribed, which processes the measurements locally. Such processing might include doing: *i*) some sort of data aggregation, FeX or data compression; or *ii*) some inference (e.g., for degradation detection) with no data transformation. The output data (transformed or not) are sent to a gRPC interface module through the Redis DB (not shown in the figure) (4), which applies some sort of data security, like encryption or digital signature using keys provided by the security manager, and then, the data are sent to the telemetry manager. Because gRPC requires a previous definition of the data to be conveyed, our implementation defines a unique message of type *bytes*, which allows generalization of the telemetry data to be conveyed. Note that, although such encoding could largely increase the volume of data to be transported, intelligent data aggregation performed by telemetry agents

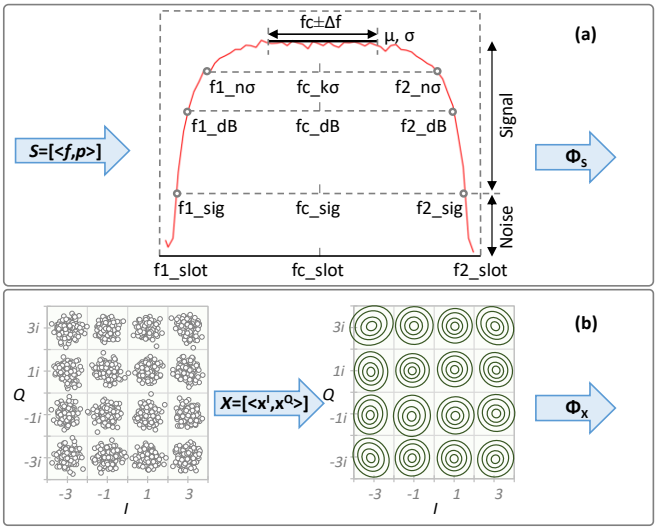

Fig. 4. Optical spectrum (a) and IQ constellation (b) samples and FeX.

could reduce such volume to a minimum.

In the telemetry manager, the data are received by a gRPC interface module. Any message received by an interface module in the telemetry agent/manager is first checked to validate that the source is a trusted peer and then, the data are decrypted or the sender is authenticated and verified data integrity. Next, a set of rules are applied to decide the workflow(s) that that message will follow. Rules play an important role for dynamically modifying workflows while affecting the related modules to that workflow only. Received messages are published to a specific topic in the Redis DB depending on the rule they matched, so subscribed modules can receive them. The data is afterward received by a data processing algorithm (5). Such algorithms in the telemetry manager can implement functions related to data aggregation, inference, etc. Once processed, the output data (6) can be stored in the telemetry DB (7) (measurements are stored in the Measurements DB and events in the Events DB) and/or be exported to external systems (8). Interestingly, algorithms in the telemetry manager can communicate with those in the telemetry agents using the gRPC interface (9, 10), e.g., for parameter tuning.

4. INTELLIGENT DATA AGGREGATION FOR DIMENSIONALITY REDUCTION

In this section, we introduce techniques to greatly reduce the impact of both volume and velocity of telemetry data. In particular, we analyze: *i*) supervised *FeX*; *ii*) data compression using *autoencoders* (AE); and *iii*) *data summarization* using the arithmetic mean of a number of observations obtained when variation is stable. We focus on two examples of telemetry measurements, optical spectrum and IQ constellations from a *m*-QAM signal, which are by far the cases where collected samples are larger.

A. Supervised feature extraction

A simple but effective dimensionality reduction technique

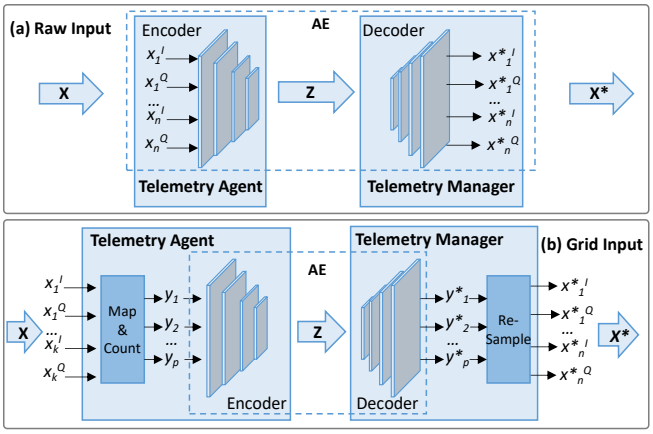

Fig. 5. IQ constellation sample compression using autoencoders.

is supervised FeX. This technique is intended to generate the set of features $\Phi(M)$ that characterize a measurement sample M. As an example of Φ, in our previous work in [4], we proposed a module to pre-process the optical spectrum of a signal, i.e., an ordered list S of frequency-power pairs, i.e., S=[<f, p>] (see Fig. 4a). After equalizing power, the module characterizes the mean (μ) and the standard deviation (σ) of the power around the central frequency ($fc\pm\Delta f$), as well as a set of primary features computed as cut-off points of the signal with the following power levels: *i*) equalized noise level, denoted *sig* (e.g., -60dB + equalization level); *ii*) a family of power levels computed with respect to μ minus $n\sigma$, denoted $n\sigma$ (e.g., 3 and 5σ); and *iii*) a family of power levels computed with respect to μ minus a number of dB (e.g., -3 and -6 dB), denoted *dB*. Each of these power levels generates a couple of cut-off points denoted $f1_{(\cdot)}$ and $f2_{(\cdot)}$. In addition, the assigned frequency slot is denoted $f1_{\text{slot}}$, $f2_{\text{slot}}$. Then, the input list with 75 <f,p> pairs representing the spectrum of a 75GHz channel is processed to generate a set Φ_S with 13 features that can be easily transformed into value, e.g., for failure detection and identification, in the telemetry agent or the manager.

Another example is for IQ constellations, where we assume that the observation point is in a TP that gathers the received optical symbols of a *m*-QAM signal. The related data source periodically retrieves a constellation sample X (a sequence of k IQ symbols as represented in Fig. 4b for a 16-QAM signal) and publish it in the local Redis DB. In our previous work in [28], we applied Gaussian Mixture Models (GMM) [29] to characterize each constellation point of an optical constellation sample as a bivariate Gaussian distribution. Therefore, each constellation point i is characterized by 5 features, the mean position in I and Q axes [μ^I,μ^Q], as well as the I and Q variance and symmetric covariance terms that the symbols belonging to the constellation point i experience around the mean [σ^I,σ^Q,σ^{IQ}]. Therefore, for a *m*-QAM signal, a set Φ_X with m*5 features need to be propagated from the telemetry agent to the manager.

B. Data compression

Let us now focus on intelligent telemetry data compression performed at telemetry agents before data are sent to the

telemetry manager through the gRPC interface. In this subsection, we target the compression of IQ constellations, since every sample X might include a large number of symbols, i.e., complex numbers. Note that the proposed compression method is compatible with any data serialization and compression engine [30] built on the gRPC interface, which applies additional compression to any data being sent, including raw samples and features.

In our previous work [28], we proposed using AEs for intelligent IQ constellations compression. An AE is a type of neural network with two components: the *encoder*, which maps input data into a lower-dimensional *latent* space, and the *decoder*, which gets data from the latent space and reconstructs the original data back. Once trained, the AE proposed in [28] takes as input k IQ symbols from the received constellation sample X, i.e., $[x_1^I, x_1^Q, \ldots x_k^I, x_k^Q]$ and generates latent space $Z=[z_1, \ldots, z_L]$, where the size of Z is significantly lower than that of X (see Fig. 5a). Although such approach (hereafter, referred to as *raw input*) shows remarkable performance in terms of compression rate and average reconstruction error, it requires IQ constellations to contain a fixed number of symbols and the same proportion of symbols per constellation point. In addition, the size of both input and output layers and consequently, the complexity of the AE, depend on the number of symbols. This lack of flexibility reduces noticeably the applicability of this approach.

In order to overcome the aforementioned issues, in this paper we propose an alternative AE-based IQ constellation compression method (hereafter referred to as *grid input*), which is sketched in Fig. 5b. Firstly, the whole IQ constellation is split into p regular grid cells, where each cell covers a small quadrant of the IQ constellation. Then, the input sample X is processed to generate vector $Y=[y_1, y_2, \ldots, y_p]$ containing the count of symbols that fall into each grid cell. Note that the length of Y only depends on p, which represents the *resolution* of IQ constellation pre-processing. Therefore, once p is fixed, the AE is trained to compress and reconstruct Y, which enables the AE to compress samples with different number of symbols and variable proportion of symbols per constellation point.

In consequence, the algorithm module in the telemetry agent runs both the map and count and the encoder, and exchanges Z for every input sample X with the decoder running in the telemetry manager through the gRPC interface. The algorithm in the telemetry manager uses the decoder to reconstruct the count of symbols in each grid cell (Y^*) and then, generates X^* by re-sampling Y^*, i.e., the number of symbols in each grid cell is generated by randomly choosing I and Q components within the range of the grid cell. Once generated, the sample X^* is stored in the telemetry DB to be subsequently analyzed. Note that reconstruction can be performed also in the telemetry agent, e.g., for veracity checking purposes, like detecting outliers and/or anomalies.

Algorithm 1. Data Summarization

INPUT: Φ
OUTPUT: *send*, Φ'

```
out ← False
for each φ ∈ Φ do
    if φ.value < R[φ.id].low OR φ.value > R[φ.id].high then
        out ← True
    H[φ.id].update(φ.value)
    Φ^avg[φ.id] ← avg(H[φ.id])
    R[φ.id].low ← Φ^avg [φ.id] - α· std(H[φ.id])
    R[φ.id].high ← Φ^avg [φ.id] + α· std(H[φ.id])
if out = True then
    count ← 0
    return True, Φ
count ← count + 1
if count = maxcount then
    count ← 0
    return True, Φ^avg
return False, -
```

Algorithm 2. Main Procedure

INPUT: *sample*
OUTPUT: *res*

```
Φ ← FeX(sample)
send, Φ' ← summarization(Φ) (Algorithm 1)
if send = False then return ∅
if isIQ(sample) = True then
    Z ← compression(sample)
    return {Φ', Z}
return {Φ', sample}
```

C. Data Summarization

In the two previous techniques, telemetry data are propagated from the observation point to the telemetry manager with the same frequency, i.e., every time a new sample M is collected from the observation point, a subset of data representing it is generated and conveyed to the telemetry manager. Assuming a high collection frequency, this policy entails large volume of data being conveyed. However, this is not needed in general in normal conditions. Hence, we could measure variations in the computed features to decide whether a new sample M or a representation of it needs to be sent to the telemetry manager. In case of no significant variations are found, the telemetry agent can send averaged values of the features with a much lower frequency, thus reducing the volume of telemetry data being conveyed.

Algorithm 1 presents the proposed data summarization procedure. The algorithm receives the set of computed features Φ and returns whether features need to be sent (Boolean variable *send*) and if needed, the set of features Φ' that can be either those received as input or averaged ones. To that end, the algorithm maintains and updates the following internal data, which are assumed to be initialized beforehand: *i*) H is a time series database containing the last w values of each feature; *ii*) Φ^{avg}, with the average value of the features stored in H; *iii*) R, with the range of variation of each feature, computed as Φ' (+/-) α times the standard deviation of values in H; and *iv*) *count*, with the number of consecutive telemetry periods where all features remain within the range R. Besides, *maxcount* defines the interval to convey averaged features.

Before starting with feature analysis, we assume that all features will stay within the range defined in R, by setting auxiliary variable *out* equal to False (line 1 in Algorithm 1). Then, input set Φ is firstly processed to find any feature that is out of the range R. If so, *out* is set to True (lines 2-4). After this, H, average features Φ^{avg}, and range R are updated accordingly (lines 5-8). Once all features have been processed, the output of the algorithm is prepared, which leads to three different cases. In case that at least one feature is out of range, *count* is reset and the input features Φ are returned (lines 9-11). Otherwise, *count* is increased and, if *maxcount* is reached, it means that a period of low frequency collection has been achieved, so *count* needs to be reset to 0 and averaged features Φ^{avg} are returned (lines 13-15). Note that in both previous cases, *send* is True in order to indicate that features must be conveyed. However, if all features are within the range and *maxcount* is not reached, then there is no need to convey any feature from agent to manager (line 16).

Algorithm 2 shows the main process that needs to be performed every time a new measurement becomes available at the telemetry agent. The output of this algorithm is the data that needs to be conveyed through the gRPC interface to the telemetry manager. Note that the result can be empty, i.e., no data need to be conveyed. The first step is to compute features $\Phi_{(\cdot)}$ from the input sample (see Section 4.A) (line 1 in Algorithm 2). Then, the data summarization procedure (Algorithm 1) is executed and, in case that there is no need to send data, empty set is returned (lines 2-3). Otherwise, if the sample is an IQ constellation, the AE-based compression detailed in Section 4.B is applied, and both the features after data summarization (Φ') and latent space (Z) are returned (lines 4-6). On the contrary, i.e., if no compression is needed, e.g., the sample is an optical spectrum, features Φ' and original sample are sent (line 7).

5. ILLUSTRATIVE RESULTS

In this section, we first present the telemetry scenario used to obtain the results and the data sources that generate telemetry measurements. Next, we focus on FeX and data compression, determine the size of the telemetry measurements at the different stages and find the compression ratios obtained with the different techniques. The performance of data summarization is then analyzed and illustrative examples are eventually presented.

A. Scenario

The telemetry system runs in 4 virtual machines (VM) deployed in an infrastructure using OpenStack as virtual infrastructure manager and Ubuntu Server 22.04 LTS as operating system (see Fig. 6). All the software, including the manager, algorithms, interfaces and the telemetry adaptor, have been implemented in Python and are executed using Python 3.10.4. Every telemetry agent and the manager with their respective Redis DB instances run inside Docker containers and are deployed using Docker Compose.

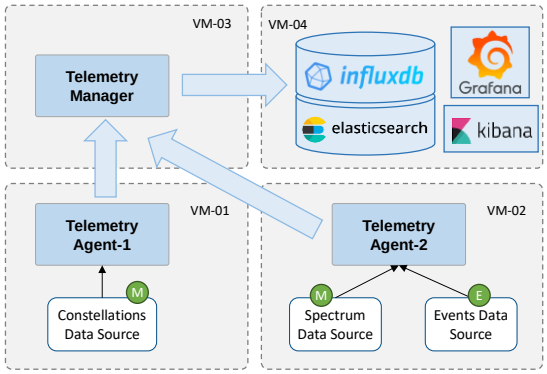

Fig. 6. Experimental deployment.

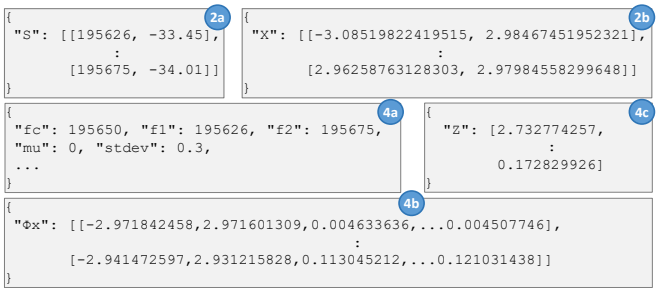

Fig. 7. JSON representations of received (2) and processed (4) samples.

Containerized versions of Influx DB 2.4.0 as measurements DB and of Elasticsearch 8.3.3 as Events DB, are deployed. To visualize data, Grafana 9.1.1 and Kibana 7.14 both running in Docker containers are used for telemetry measurements and events, respectively. In addition, a Web UI that offers a general view of telemetry system has been implemented in Python using Django 4.1.

Three data sources have been developed. Representation of the JSON objects are shown in Fig. 7 identified with the same label as the related message in Fig. 3 for the sake of clarity. The spectrum data source emulates spectrum samples collection from an OSA. The OSA measures the whole C band (4.8 THz) and an algorithm processes the measurement and selects the spectrum for each channel separately. Therefore, each measurement S for the spectrum of a 75GHz channel consists of a list of 75 $<f,p>$ pairs, i.e., 600 bytes assuming 32-bit scalars. The telemetry adaptor in the data source publishes samples S encoded as JSON objects of size 1,207 bytes (2a in Fig. 7).

The constellations data source emulates IQ constellation samples collection from an observation point in a TP. The openly available dataset in [31] have been used to get constellation samples of lighpaths with length ranging from 80km to 2000 km. Specifically, two sizes of constellation samples X are considered, containing k=2,048 and k=10,000 symbols, respectively from a 16-QAM optical signal. The size of each raw sample X in a scalar representation is $2\times k\times 4$, i.e., 16,384 bytes and 80,000 bytes, assuming that every symbol is represented with two scalars (I and Q). The telemetry adaptor publishes raw samples X encoded as JSON objects (2b in Fig. 7) of size 75,783 and 370,007 bytes, respectively.

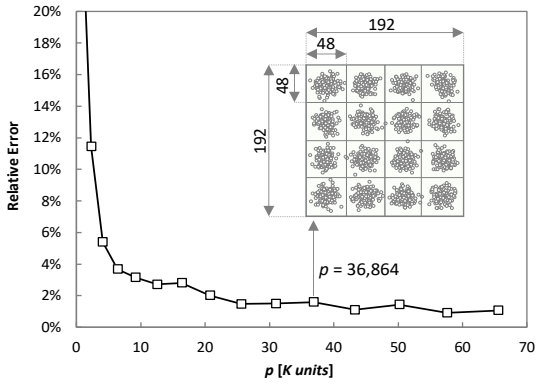

Fig. 8. Resolution Evaluation

Table 2. AE parameters and performance

AE Approach	Input Neurons	Hidden Neurons		Z	MSE	Accuracy
		#	Type			
Raw Input	20,000	500	ReLu	32	2e-2	86%
Grid Input	36,864	500	ReLu	32	4e-5	95%

Finally, the events data source reproduces TAPI entities, i.e., YANG sub-trees, generated by an SDN controller reporting asynchronous events that happen in the network. Each event has different size, being 40,000 bytes long on average.

The implemented gRPC interface removes spaces and serializes JSON objects into byte streams using the Python's *pickle* module, which can be deserialized to obtain the same object. Additionally, the gRPC interface implements security using Transport Layer Security (TLS), which encrypts the end-to-end communication between telemetry agents and the manager. On top of that, the gRPC interface applies the Gzip compression algorithm to each received message aiming at reducing even more the amount of data being transmitted.

B. Feature extraction and data compression

Intelligent data aggregation algorithms have been implemented in Python and deployed in the telemetry agent and manager for the techniques detailed in Section 4.

In the case of supervised FeX from the optical spectrum of a lightpath, the algorithm in the telemetry agent generates features Φ_S in a JSON object with 184 characters (4a in Fig. 7), which is then serialized before being conveyed through the gRPC interface. As for IQ constellations, the algorithm in the telemetry agent applies GMM fitting to every constellation sample X received and generates features Φ_X encoded as a JSON object with 1,000 characters (4b in Fig. 7).

Regarding data compression using AEs, the *raw input* approach in Fig. 5a was numerically evaluated in [28] using IQ constellation samples of reduced size (2,048 symbols). It was shown that the maximum compression that produces negligible reconstruction error results in vectors Z of size 32. Then, we now focus on comparing *raw input* and *grid input* approaches for IQ constellation samples of larger size (10,000 symbols), which provide much accurate information of the optical signal.

Table 3. Size of telemetry measurements (bytes)

Measurement	Scalar	JSON (Fig. 7)	Serialized & Gzip	Gzip compr. ratio
S (75 GHz)	600	1,207	403	3.0
S (4.8 THz)	38,400	76,807	19,773	3.9
X (2,048 symb.)	16,384	75,783	35,485	2.1
X (10k symb.)	80,000	370,007	168,677	2.2
Φs (13 featu.)	52	184	105	1.8
Φx (5x16 featu.)	320	1000	519	1.9
Z (32 values)	128	391	248	1.6

Table 4. Compression ratios from collection to gRPC

Measurement	w/o process	FeX	AE
S (75 GHz)	1.5	5.7	-
X (2,048 symb.)	0.5	31.6	66.1
X (10,000 symb.)	0.5	154.1	322.6

In the case of *grid input* approach, we need to firstly determine the resolution p that allows an accurate representation of the original constellation (see Section 4.B). To this aim, we executed the *map and count* and *resample* blocks in Fig. 5b (without AE) and compared the supervised features described in Section 4.A of both original and reconstructed constellations as fair measurement of fidelity. Fig. 8 shows the relative error of the extracted features as a function of parameter p. We observe negligible error (~1%) for $p > 35{,}000$. In view of this, we selected $p = 36{,}864$, which entails splitting the IQ constellation in a 192x192 grid i.e., each of the 16 constellation points is mapped on a 48x48 grid (see the inner graph in Fig. 8).

Table 2 shows the configuration and performance of *raw input* and *grid input* approaches after training AEs with 2,000 samples from lightpaths ranging from 80km to 2,000km during 1,000 epochs. The two selected evaluation metrics are: *i*) the loss in terms of mean squared error (MSE) computed with a validation dataset containing 500 samples not used during training; and *ii*) the reconstruction accuracy. For the sake of a fair comparison, the latter has been computed by applying the *map and count* block to both original and reconstructed constellations (regardless of the AE approach) and computing the accuracy on reconstructing the count in each grid quadrant. In light of the noticeable results, we conclude that the proposed *grid input* approach allows not only a better reconstruction than the *raw input* one, but also it achieves negligible reconstruction error when compressing 10,000 symbols into 32 latent space features. Such vectors Z of size 32 are output as JSON objects (4c in Fig. 7), resulting in 391 characters in total for the JSON object.

C. Size and data rate analysis

Let us now study the size and data rate of the different telemetry measurements. Table 3 shows the size of every telemetry measurement at their different stages: *i*) scalar representation, i.e., using float, integer and string data types; *ii*) JSON object using text; and *iii*) JSON object serialization and gzip compression. The compression ratio achieved by gzip is also presented as reference, where we observe that gzip

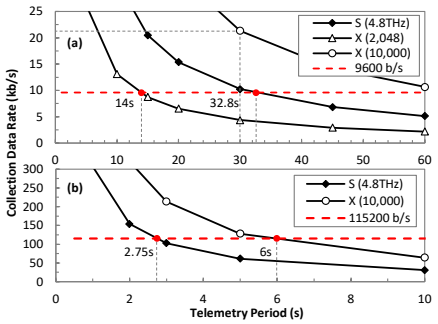

Fig. 9. Collection rate vs telemetry period.

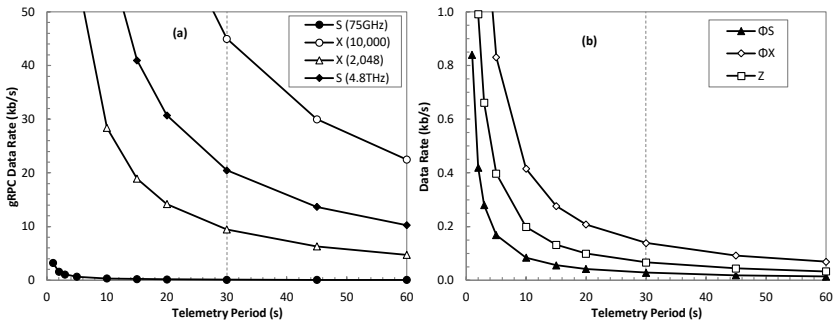

Fig. 10. gRPC data rate after serialization and gzip w/o processing (a) and using FeX/AEs (b).

Table 5. Processing times in telemetry agents (ms)

Measurement	w/o process	FeX	AE
S (75 GHz)	3	4	-
X (2,048 symb.)	4	24	12 (raw)
X (10,000 symb.)	16	243	162 (grid)

reduces the size of the JSON objects in the order of 2-3 times. Table 4 shows the achieved compression ratio from the size of the collected measurement (scalar) to the size of the serialized and compressed byte stream being conveyed through the gRPC interface, when: *i*) the measurement is sent unprocessed; *ii*) when features are extracted and sent; and *iii*) when the AE is used to generate the latent space to be sent (only in the case of optical constellations). In general, the reduction of the size when processing is carried out is higher when the size of the collected measurement is high, ranging between 5.7 and 322 times. In addition, we observe larger size reduction when AEs are used as compared to FeX. Finally, we observe that if no processing is performed, using compressed JSON objects for the gRPC interface results in increased size of telemetry measurements. In this case, binary serialization of the scalar measurement would be a much better option.

Let us analyze the requirements of the data interface of the optical devices and their relation to the telemetry period. Fig. 9 presents the telemetry collection data rate when the telemetry period ranges from 1s to 1min for samples using scalar values. Assuming a maximum data rate for telemetry collection of 9600 b/s (e.g., for a typical serial interface), the minimum telemetry period for optical constellations with 2,048 symbols would be around 14 sec (Fig. 9a). That period increases to over 32 sec in the case of the spectrum for the whole C band, and over 1 minute in the case of optical constellations with 10,000 symbols. A reduced collection period increases the speed of the interface, e.g., 21.3 Kb/s are needed to collect 10,000 symbols every 30 sec. To reduce the telemetry collection period, higher speed interfaces are needed. E.g., with a 115200 b/s serial interface (Fig. 9b), the telemetry period reduces to 2.75 sec in the case of the spectrum for the whole C band, and to 6 sec for optical constellations with 10,000 symbols.

Once the samples are collected, let us analyze the data rates generated through the gRPC interface when the serialized and gzip compressed measurements are conveyed (Fig. 10). In Fig. 10a, we observe that sending unprocessed samples results in large data rates. E.g., assuming a telemetry period of 30 sec

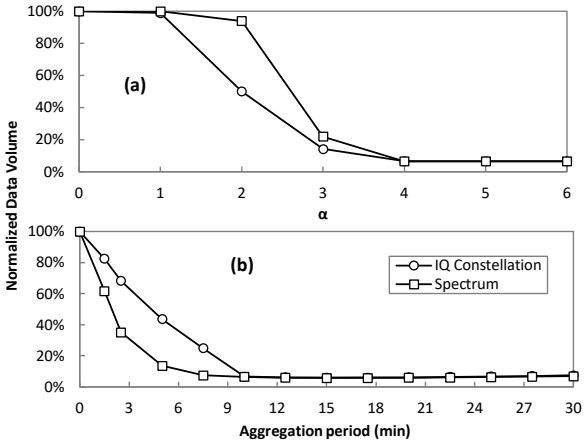

Fig. 11. Data summarization performance

results in data rates as high as 45 kb/s in the case of optical constellations with 10,000 symbols, which reduces to 9.5 kb/s when the constellations have 2,048 symbols only. In the case of the optical spectrum of a single channel, the generated data rate is low because of the coarse resolution of the OSA. In Fig. 10b, the generated data rate when FeX or AEs are used to reduce the dimensionality of the telemetry measurements is shown. In this case, data rates as low as 138 / 66 b/s are generated when FeX / AEs are used to process the received constellation samples.

In the example in Section 2 for a network with 50 nodes and considering constellation and spectrum measurements only, the network would generate 541 GB of data every day when samples are sent to the centralized repository every 30 sec in their scalar format, i.e., 192.89 TB per year. This reduces to 3.74 GB per day, 1.33 TB per year, when telemetry measurements are processed by the algorithms in the telemetry agents.

Table 5 presents the processing times in the telemetry agent for each sample as a function of the type of processing, including when only the format of the sample is changed, when features are extracted and when the encoder is used for data compression. Note that in the case of AE, additional time is needed for the decoder to reconstruct the samples, which takes the same time as the encoder. We observe relatively short processing times, which increases with the size of the input sample. This indicates the low complexity of the proposed algorithms. In fact, in our tests, two cores of an Intel

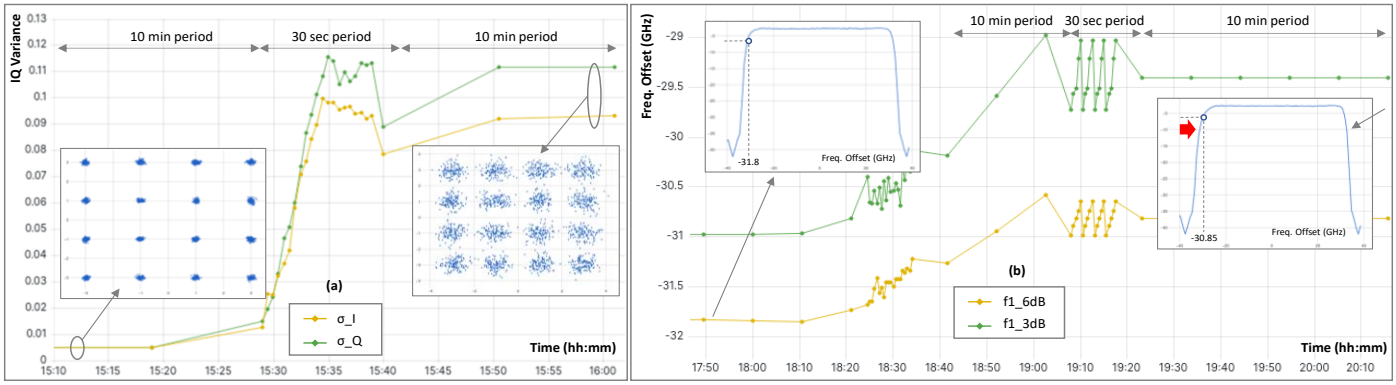

Fig. 12. Example of data summarization for constellation samples. Visualization in Grafana.

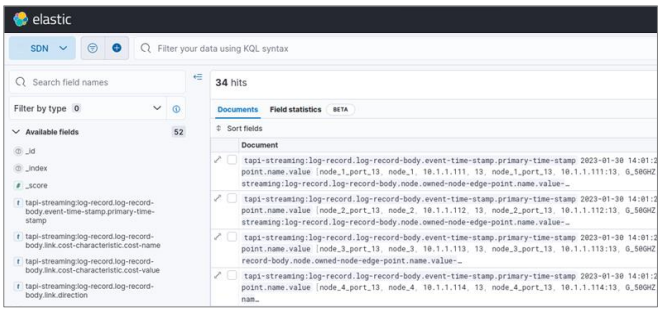

Fig. 13. Event visualization in Kibana

i7 processor were able to extract features of 5 constellations samples with 10,000 symbols per second. Note that although such computing resources need to be available in every network location, their cost is more than compensated by the reduction in the volume of data that would otherwise require to be sent to the central processing system.

Finally, when messages arrive at the telemetry manager through the gRPC interface, they follow specific processing. In the specific case of the latent space *Z*, it is used as input to the decoder that generates samples *X**. To compare the results of FeX to those from the AE, the algorithm in the telemetry manager, samples each distribution to obtain IQ constellations samples with *k* symbols and stores them in the DB.

D. Data summarization

Telemetry data rate can be further reduced by implementing data summarization on the extracted features (Algorithm 1), which can be sent with a larger period in case no significant changes occur. Algorithm 1 has been evaluated for both IQ constellation using samples with 10,000 symbols and 75 GHz optical spectrum samples from a lightpath of 1,000 km operating under normal conditions. Fig. 11a shows the data volume sent by the telemetry agent to the manager as a function of parameter α. Data volume is normalized to the no summarization case, i.e., data is sent every collection period (30 sec). We observe that configuring α=4 allows achieving remarkable summarization (<10% of total telemetry measurements) for both IQ constellations and optical spectrum samples. In addition, Fig. 11b shows the impact of the aggregation interval after fixing α=4. As a result, we conclude that an aggregation interval of 10 min, i.e., w=20

Table 6. Processing time and telemetry measurements volume

	Scalar (30 sec)	FeX + AE (30 sec)	Summarization (30 sec / 10 min)
E2e processing time (S)	6 ms	6 ms	1 ms (+ FeX + AE)
E2e processing time (X)	32 ms	583 ms	7 ms (+ FeX + AE)
Measurements (day)	541 GB	3.74 GB	0.21 GB
Measurements (year)	192.89 TB	1.33 TB	78 GB

telemetry measurements collected every 30 seconds, achieves the largest data summarization observed in Fig. 11a. Note that 10 min is an enough short period to adapt to smooth variations without the need of sending finer telemetry data.

Fig. 12 shows the measurements that are finally stored in the Measurements DB for a lightpath of 80 km. The plots are extracted from visualization panels in Grafana. For illustrative purposes, we synthetically induced a gradual degradation of the lightpath (length increase) that causes variation (large dispersion) on the received IQ constellations. Fig. 12a plots the evolution of σ^I and σ^Q features for constellation point (-3+3i); before and after constellations stored in the Telemetry DB are also shown. Data summarization reduces the telemetry period to 10 min until the value of the features increases significantly due to the induced degradation, when it increases the telemetry period to equal the collection period. This behavior facilitates algorithms in the telemetry manager to detect the degradation as fast as possible. Once the value of the features stabilizes again, the telemetry period is automatically increased back. Next, the reference lightpath suffers a filter shift of 1 GHz in some ROADM along its route, which causes variation on the measured spectrum. Fig. 12b plots the evolution of $f1_{3dB}$ and $f1_{6dB}$ features. Data summarization adapts the telemetry period as needed to follow the variations in the features, while reducing the volume of measurements that are sent.

Finally, to assess the support of variety of telemetry data on the proposed telemetry architecture, Fig. 13 shows a capture of Kibana with some TAPI events generated by an SDN controller and injected through telemetry agent 2 (see Fig. 6).

6. CONCLUDING REMARKS

The 5 V's of telemetry data have been examined and illustrated and the *variety* of measurements, e.g., from optical devices, as well as events, e.g., from control systems, that are part of telemetry data in the context of optical networking, were reviewed. We concluded that having fine grain (*velocity*) and true (*veracity*) telemetry for network data analysis (*value*) at a centralized location only results in a large amount of data to be conveyed from the devices (*volume*). A way to deal with such characteristics is by extending the telemetry pipeline and adding intelligence as close as possible to the observation points, where measurements are collected. In view of that, a distributed architecture has been proposed, where intelligence is not only located in the centralized system, but also in the telemetry agents, which receive and process telemetry measurements before sending them to the central location.

Intelligent data aggregation for dimensionality reduction (volume and velocity) has been proposed for those measurements with larger size, i.e., optical spectrum and IQ constellations. In particular, three techniques have been proposed: *i*) supervised FeX; *ii*) data compression using AE; and *iii*) data summarization.

Synthetically-generated events and measurements of optical spectrum and IQ constellations are injected into the telemetry system to experimentally obtain illustrative results. It was shown that the proposed intelligent data aggregation techniques introduce negligible error and relatively low processing delay, while reducing the dimensionality of the telemetry measurements several orders of magnitude. Table 6 summarizes processing times and amount of data from measurements collected for a network with 50 nodes (described in Section 2) in one single day and along one year. The benefits of the proposed telemetry system architecture and the proposed intelligence techniques are absolutely clear in the view of the results.

Funding. The research leading to these results has received funding from the Smart Networks and Services Joint Undertaking under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096120 (SEASON), from the MICINN IBON (PID2020-114135RB-I00) project and from the ICREA Institution.

REFERENCES

1. L. Velasco, P. Layec, F. Paolucci, and N. Yoshikane, "Introduction to the JOCN Special Issue on Advanced Monitoring and Telemetry in Optical Networks," IEEE/OSA J. Opt. Commun. Netw., vol. 13, pp. AMTON1-AMTON2, 2021.
2. L. Velasco, A. Chiadò Piat, O. González, A. Lord, A. Napoli, P. Layec, D. Rafique, A. D'Errico, D. King, M. Ruiz, F. Cugini, and R. Casellas, "Monitoring and Data Analytics for Optical Networking: Benefits, Architectures, and Use Cases," IEEE Network Magazine, vol. 33, pp. 100-108, 2019.
3. L. Velasco, S. Barzegar, D. Sequeira, A. Ferrari, N. Costa, V. Curri, J. Pedro, A. Napoli, and M. Ruiz, "Autonomous and Energy Efficient Lightpath Operation based on Digital Subcarrier Multiplexing," IEEE J. on Selected Areas in Comm., vol. 39, pp. 2864-2877, 2021.
4. A. P. Vela, B. Shariati, M. Ruiz, F. Cugini, A. Castro, H. Lu, R. Proietti, J. Comellas, P. Castoldi, S. J. B. Yoo, and L. Velasco, "Soft Failure Localization during Commissioning Testing and Lightpath Operation [Invited]," IEEE/OSA J. Opt. Commun. Netw., vol. 10, pp. A27-A36, 2018.
5. S. Barzegar, M. Ruiz, A. Sgambelluri, F. Cugini, A. Napoli, and L. Velasco, "Soft-Failure Detection, Localization, Identification, and Severity Prediction by Estimating QoT Model Input Parameters," IEEE Transactions on Network and Service Mngt., vol. 18, pp. 2627-2640, 2021.
6. B. Shariati, M. Ruiz, J. Comellas, and L. Velasco, "Learning from the Optical Spectrum: Failure Detection and Identification [Invited]," IEEE/OSA J. of Lightwave Technol., vol. 37, pp. 433-440, 2019.
7. C. Natalino, A. Udalcovs, L. Wosinska, O. Ozolins and M. Furdek, "Spectrum Anomaly Detection for Optical Network Monitoring Using Deep Unsupervised Learning," IEEE Communications Letters, vol. 25, pp. 1583-1586, 2021.
8. D. Sequeira, M. Ruiz, N. Costa, A. Napoli, J. Pedro, and L. Velasco, "OCATA: A Deep Learning-based Digital Twin for the Optical Time Domain," IEEE/OPTICA J. Opt. Commun. Netw., vol. 15, pp. 87-97, 2023.
9. L. Velasco, M. Devigili, and M. Ruiz, "Applications of Digital Twin for Autonomous Zero-Touch Optical Networking [Invited]," invited in International Conference on Optical Network Design and Modeling (ONDM), 2023.
10. D. Laney, "3D Data Management: Controlling Data Volume, Velocity, and Variety," META Group Technical Report, 2001.
11. H. Lun; X. Liu; M. Cai; Y. Zhang; R. Gao; W. Hu; L. Yi; Q. Zhuge, "Machine-learning-based telemetry for monitoring long-haul optical transmission impairments: methodologies and challenges [Invited]," IEEE/OSA J. Opt. Commun. Netw., vol. 13, pp. E94-E108, 2021.
12. D. Rafique and L. Velasco, "Machine Learning for Optical Network Automation: Overview, Architecture and Applications," IEEE/OSA J. Opt. Commun. Netw., vol. 10, pp. D126-D143, 2018.
13. J. Pesic, M. Curtol, L. Abnaou, A. E. Imadi and S. Morganti, "SDN Automation for Optical Networks Based on Open APIs and Streaming Telemetry," in Proc. Int. Conference on Optical Network Design and Modelling (ONDM), 2022.
14. M. Ghobadi, and M. Ratul, "Optical Layer Failures in a Large Backbone." In Proc. Internet Measurement Conference (IMC), 2016.
15. R. Casellas, R. Martínez, R. Vilalta, R. Muñoz, A. González-Muñiz, O. González de Dios, and J.-P. Fernández-Palacios, "Advances in SDN control and telemetry for beyond 100G disaggregated optical networks [Invited]," IEEE/OSA J. Opt. Commun. Netw., vol. 14, pp. C23-C37, 2022.
16. H. Qarawlus, S. Biehs, B. Shariati, J. J. P. Manresa, A. Bouchedoub, H. Haße, P. Safari, A. Autenrieth, and J. Fischer, "Demonstration of Data-Sovereign Telemetry Broker for Open and Disaggregated Optical Networks," in Proc. Optical Fiber Communication Conference (OFC), 2023.
17. OIF: SDN Transport API, [On-line] https://www.oiforum.com/technical-work/hot-topics/sdn-transport-api-2/.
18. F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," IEEE/OSA J. Lightwave Technol., vol. 36, pp. 3142-3149, 2018.
19. Ll. Gifre, J.-L. Izquierdo-Zaragoza, M. Ruiz, and L. Velasco, "Autonomic Disaggregated Multilayer Networking," IEEE/OSA J. Opt. Commun. Netw., vol. 10, pp. 482-492, 2018.
20. SONiC, [On-line] http://sonic-net.github.io/SONiC/
21. A. Giorgetti, D. Scano, A. Sgambelluri, F. Paolucci, E. Riccardi, R. Morro, P. Castoldi, and F. Cugini, "Enabling hierarchical control of coherent pluggable transceivers in SONiC packet–optical nodes," IEEE/OPTICA J. Opt. Commun. Netw., vol. 15, pp. 163-173, 2023.
22. A. Sgambelluri, A. Pacini, F. Paolucci, P. Castoldi, and L. Valcarenghi "Reliable and scalable Kafka-based framework for optical network telemetry," IEEE/OPTICA J. Opt. Commun. Netw., vol. 13, pp. E42-E52, 2021.
23. L. Velasco, P. González, and M. Ruiz, "An Intelligent Optical Telemetry Architecture," in Proc. OFC, 2023.
24. L. Velasco, S. Barzegar, and M. Ruiz, "Is Intelligence the Answer to Deal with the 5 V's of Telemetry Data?" in Proc. OFC, 2023.
25. P. Gonzalez, R. Casellas, J-J Pedreno-Manresa, A. Autenrieth, F. Boitier, B. Shariati, J. Fischer, M. Ruiz, J. Comellas, and L. Velasco, "Distributed Architecture Supporting Intelligent Optical Measurement Aggregation and Streaming Event Telemetry," in Proc. OFC, 2023.
26. H. Lun, X. Liu, M. Cai, Y. Zhang, R. Gao, W. Hu, L. Yi, and Q. Zhuge, "Machine-learning-based telemetry for monitoring long-haul optical transmission impairments: methodologies and challenges [Invited]," IEEE/OSA J. Opt. Commun. Netw., vol. 13, pp. E94-E108, 2021.
27. R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Control, management, and orchestration of optical networks: Evolution, trends, and challenges," IEEE J. Lightwave Technol., vol. 36, pp. 1390-1402. 2018.

28. M. Ruiz, D. Sequeira, and L. Velasco, "Deep Learning -based Real-Time Analysis of Lightpath Optical Constellations [Invited]," IEEE/OPTICA J. Opt. Commun. Netw., vol. 14, pp. C70-C81, 2022.
29. N. Bouguila and W. Fao, *Mixture Models and Applications*, Springer, 2020.
30. C. McAnlis and A. Haecky, *Understanding Compression: Data Compression for Modern Developers*, O'Reilly, 1st Edition, 2016.
31. M. Ruiz, L. Velasco, and D. Sequeira, "Optical Constellation Analysis (OCATA)," [On-line] https://doi.org/10.34810/data146, 2022.