A Scalable Telemetry Framework for Zero Touch Optical Network Management

L. Valcarenghi, A. Pacini, A. Sgambelluri
Scuola Superiore Sant'Anna, Pisa, Italy

F. Paolucci
CNIT, Pisa, Italy

***Abstract*—The interest about Zero Touch Network and Service Management (ZSM) is rapidly emerging. As defined by ETSI, the ZSM architecture is based on a closed-loop/feedback control of the network and the services. Such closed-loop control can be based on the Boyd's Observe Orient Decide and Act (OODA) loop that matches some specific management functions such as Data Collection, Data Analytics, Intelligence, Orchestration and Control. An efficient implementation of such control loop allows the network to timely adapt to changes and maintain the required quality of service.**

Many solutions for collecting network parameters (i.e., implementing ZSM data collection) are proposed that fall under the broad umbrella of network telemetry. An example is the Google gRPC, that represented one of the first solutions to provide a framework for data collection. Since then, the number of available frameworks is proliferating. In this paper we propose the utilisation of Apache Kafka as a framework for collecting optical network parameters. Then, the paper goes beyond that by proposing and showing how Apache Kafka can be effective for supporting data exchange and management of whole ZSM closed-loop.

Experimental evaluation results show that, even when a large number of data are collected, the solution is scalable and the time to disseminate the parameter values is short. Indeed, the difference between the reception time and the generation time of data is, on average, 40-50ms when about four thousand messages are generated.

I. Introduction

Control systems are not novel in engineering. They are heavily employed in many fields, such as mechanical engineering, aerospace, electrical engineering. In general, a control system consists of *subsystems* and *processes* (or *plants*) assembled for the purpose of obtaining a desired *output* with desired *performance*, given a specified *input* [1].

Control systems can be used to provide convenience by changing the form of the input. For example, in a temperature control system, the input is a position on a thermostat while the output is heat. Thus, a convenient position input yields a desired thermal output. In a network this could be represented by the mapping between service Key Performance Indicators (KPIs) or Service Level Agreement (SLA) and network KPIs. Another advantage of a control system is the ability to compensate for disturbances. For example, in an antenna system that points in a commanded direction, the system itself might measure the amount that a disturbance (e.g., winds) has repositioned the antenna and then return the antenna to the position commanded by the input. In this case the system input does not change but its the autonomous system control that repositions the antenna.

Control systems present two major configurations: open-loop systems and closed-loop or feedback systems. Open-loop systems do not correct for disturbances and are simply commanded by the input. Closed-loop systems compensate for disturbances by measuring the output response, feeding that measurement back through a feedback path, and comparing that response to the input at the summing junction. If there is any difference between the two responses, the system drives the plant, via the actuating signal, to make a correction. Thus, closed-loop control systems make the systems more autonomous.

Recently, many research projects and Standard Developing Organisations (SDOs), such as ETSI, are aiming at applying closed-loop control to communication networks for network and service management, in particular. Among the motivations is the need for overcoming the traditional network management, done in a silo-oriented way with very limited automated interaction among those management silos. Another one is the need for transformation from the old Operational System Support (OSS) into a modern, autonomous network management environment not anymore based on open-loop but on closed-loop for providing Fault, Configuration, Account, Performance and Security (FCAPS) Management. The main objective is to provide a framework that enables zero-touch automated network and service management (ZSM) in a multivendor environment based on data-driven management algorithms based, for example, on machine learning and artificial intelligence. Finally, another motivation is the support of Intent-based Networking [2], [3] that is based on defining high level policies for the network, translated to configuration by underling systems. Morever, closed-loop control is proposed also for the management and control of the 5G Radio Access Network (RAN) in the O-RAN architecture [4].

To implement the closed-loop, ZSM can exploit the research conducted in another ETSI initiative: the Experiential Networked Intelligence (ENI) [5]. The ENI specifies an architecture to enable closed-loop network operations and management leveraging Artificial Intelligence and Machine Learning (AI/ML). ENI can be deployed as an external AI/ML entity, outside of an existing "Assisted System".

This paper first overviews ZSM and current efforts in network telemetry, with specific focus on optical networks. Indeed, telemetry is an essential element for providing ZSM

with information about the network status. Then, the paper focuses on how the Apache Kafka framework can be utilised not only as an efficient data streaming service but also as the overall framework underpinning the ZSM functional elements. Finally, results on the Kafka scalability are provided.

II. ZSM OVERVIEW

The Zero-touch Network and Service Management (ZSM), proposed by ETSI [6], is among the emerging architectures for autonomously operating and managing communications networks. Research papers and projects already reported studies related to network management automation [7]. For example, Deep Learning (DL) applications for various traffic control aspects, such as network traffic classification, network flow prediction, mobility prediction, Cognitive Radio Networks (CRNs), and Self-Organized Networks (SONs) have been proposed in [8]. A closed-loop solution for network slicing, where traffic forecasting information is ingested by an admission control engine to maximize the number of granted network slice requests while meeting the slice Service Level Agreements (SLAs) guarantees, is presented in [9]. Forecasting/predictive analytics has been proposed in [10] for scaling 5G core network resources by anticipating traffic load changes. The approach is based on two Artificial Intelligence/Machine Learning (AI/ML) techniques: Recursive Neural Networks (RNN), more specifically Long Short-Term Memory (LSTM), and Deep Neural Network (DNN). The results show that forecast-based scalability mechanism outperforms threshold-based solutions in terms of latency to react to traffic change. In [11] a first demonstration of ROADM White Box augmented with machine learning capabilities is demonstrated. The white box includes various levels of disaggregation, NETCONF/YANG control, telemetry and spectrum-based advanced monitoring functionalities.

However, in all the aforementioned papers all the control loop functions are co-located or the delay in exchanging data among the control loop functional elements is not taken into account. One of the first studies taking into account the delay introduced by the data collection is presented in [12]. There, the time response of a Hybrid Forecast Framework (HFF) running at the network edge or in the cloud is compared. When running at the edge, the HFF exploits traditional time series analysis prediction algorithms to minimize the utilized resources and energy while it exploits AI/ML tools to make predictions in the cloud. Results show that if the look ahead time is long, cloud-based prediction is preferable because the long look ahead time compensates for the higher time for prediction, mainly due to data transfer.

Other architectures than the ZSM are proposed by standard developing organisations (SDOs): the TM Forum's Open Digital Framework (ODF) [13], MEF 3.0 Lifecycle Services Orchestration (LSO) [14], Linux Foundation Platform for Network Data Analytics (PNDA) [15], TM Forum Smart (BPM) [16]. Even research projects are exploring solutions that provide management automation. SLICENET [17] is a 5G-PPP phase II project that aims at designing and implementing an E2E cognitive vertical-oriented 5G network slicing framework. 5G-enabled Growth in Vertical Industries (5Growth) [18] aims at empowering vertical industries such as Industry 4.0, Transportation, and Energy with Artificial Intelligence-driven automated 5G end-to-end solutions that allow them to simultaneously achieve their respective key performance targets.

In general, the ZSM shall provide means for the ordered invocation of the phases of a closed-loop control. An example is the closed-loop devised by J.R. Boyd, also known as the OODA loop: the Observe, Orient, Decide, and Act. Thus, it possible to find a mapping between management functions and closed-loop steps: Data Collection contributes to Observe, Analytics contributes to Orient, Intelligence contributes to Decide, and Orchestration and Control contribute to the Act steps as shown in Fig. 1.

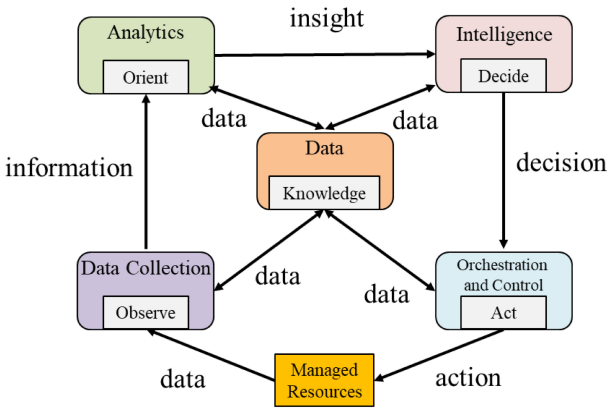

Fig. 1. Mapping between ZSM architecture building blocks and closed loop functions as in [6]

In the classical control engineering the performance of a system that is controlled by a closed-loop depends on many factors, such as the feedback that is applied. Typical key performance indicators are the system time response and the stability. In ZSM, because of the complexity of the utilized functional elements, the performance depend not only on the time it takes to execute each functional element of the control loop but also on the time it takes to exchange data among the functional elements. Such times impact the reaction time of the system, that can be assimilated to the system response time of the classical control engineering closed-loop.

III. TELEMETRY OVERVIEW

In general, network telemetry is "...a technology for gaining network insight and facilitating efficient and automated network management..." [19]. As stated in [19], the elaboration of network big data provides network operators with an opportunity to gain network insights and move towards network autonomy. This evolution would probably lead to the reduction of human labor, to a more efficient use of network resources, and to provide better services more aligned with customer requirements. However, with respect to the fast evolution of

data elaboration techniques, for example based on AI/ML, the data collection (i.e., telemetry) is lagging behind in developing efficient ways for extracting and translating network data into useful information.

For what concerns optical networks, in the last five years, there have been significant research activities related to telemetry and monitoring architectures and solutions. The main research directions are based on telemetry protocols, platforms, architectures, and novel disaggregated optical devices offering open interfaces and hardware drivers for physical parameter monitoring.

The work in [20] is one of the first proposals of YANG extensions for on-demand optical equipment data acquisition, in the context of the IETF Abstraction and Control of TE Networks (ACTN) virtual network models. The work in [21] presents two architectural implementations of telemetry based on partial and full disaggregation degrees, using the gRPC Remote Procedure Calls (gRPC) framework, showing good scalability performance and highlighting the effectiveness of the gRPC framework, able to support efficient encoding and native push streaming mode, more efficient with respect to NETCONF-based notification mechanisms.

The study reported in [22] proposed the use of data analytics platforms based on continuous monitoring of IPFIX protocol observation points related to sliced optical network based on the Spark platform. The work was extended in [23] to support disaggregation and perform telemetry of data sourced by open Optical Spectrum Analyzer (OSA) aiming at inferring optical filter malfunctioning. Multi-layer extensions were covered in [24].

Focusing on the optical device agent, the work in [25] adopts an extended agent performing threshold-based telemetry streaming for network verification and data analytics. The work in [26] proposes an online gRPC telemetry of physical fiber parameters (i.e., bending) affecting coherent receiver OSNR, combined with AI-based convolutional network.

A number of recent works report the implementation of telemetry agents co-located with optical devices having high frequency rates and resolutions [27]. For example, in [28] it is proposed a power monitor blade with 400μs telemetry streaming period and control plane using direct memory access to support AI engines. In addition, the work reported in [29] details an open disaggregated ROADM including filterless add/drop module equipped with photodetector tap arrays with gRPC telemetry servers including NETCONF notifications, capable of 20Hz sample update frequency.

In the framework of multi-layer networks a combined per-layer telemetry approach has been proposed in [30]. Authors propose to combine optical telemetry related to lightpath health with in-band telemetry (INT) of tributary flows handled by a P4 switch, analyzing the benefits of a combined approach. In the context of packet-switched layer in-band telemetry, a preliminary evaluation of a Kafka-based monitoring system handling INT values has been carried out in [31], proposing an event-triggered mechanism to produce telemetry reports to the Kakfa system from aggregate statistics related to specific data plane flows.

The work in [32] exploits a telemetry-based workflow to assign the proper transmission operational modes to transponders (including proprietary modes not disclosed explicitly to the SDN controller) during connection provisioning in partial disaggregation. The work evaluates also the impact of the proposed telemetry workflow in a multi-layer network upon soft failure recovery, affecting both the optical and the packet switched layer.

Optical telemetry may be extended not only to support isolated online network monitoring, but also to enrich data correlation related to optical network security, such as online video analytics [33]. Agile telemetry systems were also proposed in the critical use case of disaster recovery, addressing high reconfigurability. In particular the system is robust to unstable control plane connectivity and failure recovery to auxiliary low bandwidth network segments [34].

Although many studies are dealing with telemetry and some of its issues, such as scalability for disaggregated optical networks, other telemetry framework issues, such as reliability and joint scalability/reliability, have not been fully addressed yet. In addition, in the context of ZSM, how to provide a common communication framework for data exchange among the ZSM functional elements and their management has not been fully investigated yet.

IV. Kafka-based ZSM

This section presents the proposed Apache Kafka-based communication infrastructure to support ZSM. First, it provides an overview of the Kafka features that make it suitable for efficiently supporting communications among the ZSM functional elements. Then, it presents how Kafka is utilized for the Data Collection functional element. Finally, it presents how Kafka can be the communication infrastructure to support the whole ZSM.

A. Apache Kafka Overview

Apache Kafka [35] is an open-source distributed event streaming platform. An event (also called record or message) corresponds to a situation that happens and for which a record shall be kept, such as a change in the amount of traffic flowing through a card. The Apache Kafka architecture offers a publish subscribe communication based on a persistent storage. It provides a low latency, reliable communication, hence it is employed in many fields. The core of this platform is represented by the Kafka *cluster*. The cluster receives data from the *producers*, that are used to send data from the events' sources. Data are then stored into the cluster, which consists of one or more entities, called *brokers*. Each broker has its own storage, and it could be a physical machine, a virtual machine or a container. *Consumers* are in charge of reading those data by continuously polling them from the cluster. Every consumer is independent from each other: they can read the same event stream but at different points, and implement their own application, that could be related to processing, displaying dashboards, insights and so on. A key feature of Kafka is that

producers and consumers are decoupled. A producer does not send data to a particular consumer, it just sends data to the cluster within an abstraction called *topic*.

Kafka uses the concept of topic as a flow of related events. It acts as a logical log file, that is handled by the cluster. New messages of a topic are appended to this log, and one or more consumers can independently read from it at different positions. More specifically, a topic is split into one or more *partitions*, which are distributed among the brokers in the cluster. This is the main concept that allows Kafka to handle heavy data loads, since messages are distributed across the available partitions, and data can be consumed in parallel from them. Those partitions are practically logs, thus messages sent to the same partition belonging to the same topic follows a strict ordering. Conversely, a topic split in multiple partitions cannot preserve the global ordering, thus messages requiring to be ordered have to be sent to the same partition. In addition, Kafka offers mechanisms to replicate partitions across the other brokers. This natively provides a way of overcoming brokers' failures, since partitions that were handled by the failed brokers are automatically replaced by the others that have a copy of the unreachable partitions. It offers also processing capabilities through Kafka Streams, which applies both stateless and stateful operations on the cluster's topics. Streams applications acts as a producer and as a consumer at the same time, thus they feature the same producer/consumer horizontal scalability by processing in parallel from multiple topics' partitions. Building upon the aforementioned features, Kafka can offer low latency, resiliency, scalability and flexibility, that make it suitable for implementing the ZSM Data Collection function.

B. Kafka-based Optical Network Data Collection

This section focuses on a Kafka-based data collection/telemetry for optical networks. This framework was originally proposed in [36]. The proposed framework exploits the built-in scalability and reliability of Kafka to go beyond the traditional monitoring systems. As per Kakfa implementation, monitoring messages are exchanged in a publish subscribe fashion by simple text messages that can be formatted according to desired models without requiring specific protocol for message exchange. Specifically, the framework is mainly based on Apache Kafka, Kafka Streams, and Telegraf. Telegraf is a low-memory footprint plugin driven agent capable of collecting data from multiple sources and sending them to multiple destinations. Telegraf output plugins allow both a flexible deployment and a size configurable local buffer for retaining metrics in case the cluster is not reachable. Kafka producer is one of the available Telegraf output plugins. In the proposed framework the Telegraf output plugin collects data from a local testbed composed of several optical devices. Those metrics are sent to a unique topic, called OpticalNetwork, and distributed in a three broker cluster. The cluster is configured to fully manage a single broker failure. On top of that, a management topic is used to distribute commands to the Kafka Streams applications, that filter metrics from OpticalNetwork per lighpath. More specifically, every time a new lightpath is set up in the network, a new topic is created and all the metrics coming from all the devices involved go within.

C. Kafka as a Communication Infrastructure for ZSM

Because of the ZSM functional element complexity and their distributed nature, ZSM time response depends not only on the actions taken to control the system but also on the execution time of each functional element and their interaction (mainly communication) time. Thus, Apache Kafka appears as an effective solution to provide ZSM with a communication framework for data exchange among the ZSM functional elements and their management.

Moreover, because ZSM has potentially unlimited applications (e.g. optical networks, SDN networks, mobile networks), it is not possible to consider any possible condition and consecutive reactions a priori. Thus, a ZSM framework should be capable of growing and scaling out considering always more complex scenarios, along with its responsibility of managing the network. For this reason, a modular approach is fundamental: new components and functionalities have to be tested without affecting the working configuration. Apache Kafka perfectly fits this scenario. Thanks to its publish-subscribe pattern, the same data that an active module in the ZSM is ingesting can be consumed by another one for testing purpose.

V. Performance Evaluation

This section reports some initial evaluation on the convergence time of the proposed Kafka-based solution of the ZSM Data Collection functional element. The convergence time is defined as the time required to deliver a message from a producer to a consumer (i.e.,end-to-end latency experienced by the messages).

A. Evaluation Scenario

The evaluation is based on the cloud deployment depicted in Figure 2.

The three Kafka Streams application (e.g., Stream App1, 2, 3) run in three Ubuntu 18.04 physical machines, with 4 Intel CPUs @1.5GHz, 4GB of RAM and 1GB of JVM committed memory.

Then, two Ubuntu 18.04 servers (e.g., Producer Server1 and Producer Server2), equipped with 16 AMD Epyc 7262 CPUs @3.2GHz and 64GB of RAM, have been used to run the Python-based Kafka producers.

Another Ubuntu 18.04 server (e.g., Consumer Server), equipped with 32 Intel Xeon Gold 6244 CPUs @3.60GHz and 64GB of RAM, has been used to run the Kafka consumers to consume, in parallel, the messages received over the different topics.

A Windows physical machine (e.g., GUI), equipped with 2 Intel Core i5 @3.1GHz and 8GB of RAM, has been used (i) to run the Telegraf producer, using the python-based driver as input plugin to retrieve the data from real optical devices, (ii)

to execute an instance of the influxDB time-series database and to run the Grafana-based GUI.

Inside a cloud EXS environment, represented with grey background in the figure, the Kafka cluster, consisting of three Ubuntu 18.04 virtual machines (e.g., Broker1, Broker2 and Broker3), with 4 Intel Xeon CPUs @2.3GHz and 8GB of RAM, each running the Kafka broker along with a Zookeeper instance, have been deployed. All of them have 1GB of JVM committed memory. Finally, an other Ubuntu 18.04 virtual machine (e.g., KtoDB), with 4 Intel Xeon CPUs @2.3GHz and 8GB of RAM, has been adopted to receive all the metrics and to inject this data to the time-series influxDB database.

All the considered servers are interconnected using a dedicated L2 Ethernet network at 1 Gbps.

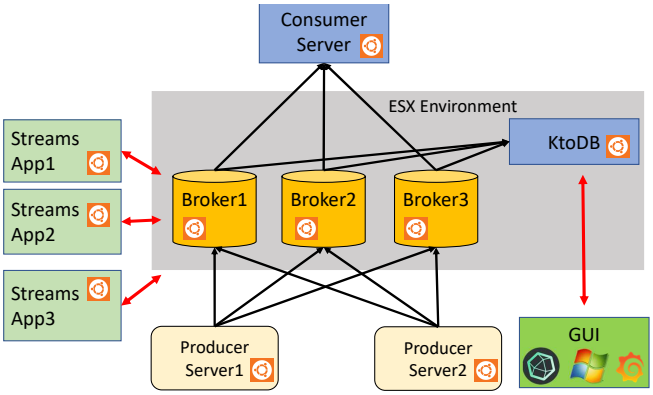

Fig. 2. Experimental testbed considered for the performance assessment.

messages per second. The consumer group, consists of three consumers, connected to the three partitions, receiving all the generated messages in parallel. The test has been repeated ten times, collecting the performance per partition (i.e., per broker) considering six hundred messages (i.e., test duration of sixty seconds). No message loss has been detected in all the executed tests. Figure 3 shows the distribution of the end-to-end latency respectively collected for partition0, partition1 and partition2.

The three distributions are quite similar, with around the 90% of the samples in the range 20-140ms, average end-to-end latency 53.88ms, 53.48ms, 53.92ms respectively for partition0, partition1 and partition2 and confidence interval at 95% in the range 4.85ms and 5.32ms.

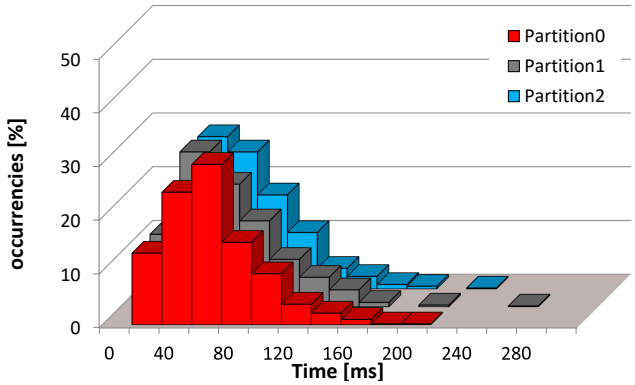

Fig. 3. End-to-End latency distribution for the messages handled by the three configured partitions.

B. Results

To validate the performance of the system in terms of convergence time, a scenario consisting of ten consumer groups active and thirty-two producers generating 3920 messages/s is considered. The consumer groups are subscribed to all the active Kafka topics. An additional consumer group and three additional producers, connected only to the Management topic, are utilised for our performance evaluation. Both the consumers and the producers are running in the same machine to preserve the time synchronization, adopting the same clock reference. The producers send messages over the Management Kafka topic, including the generation timestamp, related to the following elements of a partially disaggregated optical netywork: xPonder nodes that control the optical cards of an optical white box and report the metrics related to the optical ports of all the controlled cards; Optical Line System (OLS) agents reporting metrics related to all the controlled amplifiers spread over the different optical links. At the consumer side, subscribed to the Management topic, the elapsed time is computed considering the difference between the reception time and the generation time, contained within the received message.

To evaluate the system performance each producer generates ten messages per second (i.e., one message each 100ms) to one of three partitions, resulting to an aggregate rate of thirty

Obtained results show that the transmission of the messages through the Kafka cluster will produce an additional average delay of around 50ms on the communication among producer (i.e., network device) and consumer (i.e., the collector) with no difference regarding the traversed broker.

VI. Conclusion

Zero Touch Network and Service Management (ZSM) is based on a closed-loop control of the network and of the services. ZSM closed-loop control consists of Data Collection, Data Analytics, Intelligence, Orchestration and Control functions.

This paper proposed the utilisation of Apache Kafka as a framework for implementing the Data Collection function of a ZSM applied to an optical network. Moreover, the paper proposed how Apache Kafka can be effective for supporting data exchange and management of the whole ZSM closed-loop.

Experimental evaluation results show that, even when a large number of data are collected, the solution is scalable and the time to disseminate the parameter values is short. Indeed, the difference between the reception time and the generation time of data is, on average, 40-50ms when about four thousand messages are generated.

ACKNOWLEDGMENT

This work received funding from the ECSEL JU project BRAINE (grant agreement No 876967). The JU receives support from the EU Horizon 2020 research and innovation programme and the Italian Ministry of Education, University, and Research (MIUR).

REFERENCES

[1] N. S. Nise, *Control Systems Engineering, 8th Edition*. Wiley, 2019.

[2] A. Campanella, "Intent based network operations," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, 2019, pp. 1–3.

[3] C. Jacquenet, "Optimized, automated and protective: An operator's view on future networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[4] O-RAN Alliance, "O-ran architecture description," O-RAN.WG1.O-RAN-Architecture-Description-v03.00, Technical Specification, 2021.

[5] ETSI, "Experiential networked intelligence (eni); system architecture," ETSI GS ENI 005 V1.1.1 (2019-09), 2019.

[6] ——, "Zero-touch network and Service Management (ZSM); Reference Architecture," ETSI GS ZSM 002, V1.1.1, 2019.

[7] C. Benzaid and T. Taleb, "Ai-driven zero touch network and service management in 5g and beyond: Challenges and research directions," *IEEE Network*, vol. 34, no. 2, pp. 186–194, 2020.

[8] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[9] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5g network slicing resource utilization," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[10] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving traffic forecasting for 5g core network scalability: A machine learning approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018.

[11] A. Sgambelluri, J.-L. Izquierdo-Zaragoza, A. Giorgetti, L. Gifre, L. Velasco, F. Paolucci, N. Sambo, F. Fresi, P. Castoldi, A. C. Piat, R. Morro, E. Riccardi, A. D'Errico, and F. Cugini, "Fully disaggregated roadm white box with netconf/yang control, telemetry, and machine learning-based monitoring," in *Optical Fiber Communication Conference*, 2018. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2018-Tu3D.12

[12] V. Reddy Chintapalli, K. Kondepu, A. Sgambelluri, A. Franklin A, B. Reddy Tamma, P. Castoldi, and L. Valcarenghi, "Orchestrating edge- and cloud-based predictive analytics services," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 214–218.

[13] MEF. [Online]. Available: https://www.tmforum.org/opendigitalframework/

[14] ——. [Online]. Available: https://www.mef.net/mef-apis/

[15] PNDA. [Online]. Available: http://pnda.io

[16] TMF. [Online]. Available: https://www.tmforum.org/catalysts/smart-bpm/

[17] [Online]. Available: https://slicenet.eu

[18] [Online]. Available: http://5growth.eu/

[19] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," Internet Engineering Task Force, Internet-Draft draft-ietf-opsawg-ntf-07, Feb. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-ntf-07

[20] Y. Lee, R. Vilalta, R. Casellas, R. Martínez, and R. Muñoz, "Scalable telemetry and network autonomics in actn sdn controller hierarchy," in *2017 19th International Conference on Transparent Optical Networks (ICTON)*, July 2017, pp. 1–4.

[21] F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," *Journal of Lightwave Technology*, vol. 36, no. 15, pp. 3142–3149, 2018.

[22] L. Velasco, L. Gifre, J.-L. Izquierdo-Zaragoza, F. Paolucci, A. P. Vela, A. Sgambelluri, M. Ruiz, and F. Cugini, "An architecture to support autonomic slice networking," *Journal of Lightwave Technology*, vol. 36, no. 1, pp. 135–141, 2018.

[23] L. Velasco, A. Sgambelluri, R. Casellas, L. Gifre, J. Izquierdo-Zaragoza, F. Fresi, F. Paolucci, R. Martínez, and E. Riccardi, "Building autonomic optical whitebox-based networks," *Journal of Lightwave Technology*, vol. 36, no. 15, pp. 3097–3104, Aug 2018.

[24] L. Gifre, J. Izquierdo-Zaragoza, M. Ruiz, and L. Velasco, "Autonomic disaggregated multilayer networking," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 5, pp. 482–492, 2018.

[25] A. Sadasivarao, S. Syed, D. Panda, P. Gomes, R. Rao, J. Buset, L. Paraschis, J. Brar, and K. Raj, "Demonstration of extensible threshold-based streaming telemetry for open dwdm analytics and verification," in *Optical Fiber Communication Conference*. Optical Society of America, 2020, pp. M3Z–5.

[26] T. Tanaka, S. Kuwabara, H. Nishizawa, T. Inui, S. Kobayashi, and A. Hirano, "Field demonstration of real-time optical network diagnosis using deep neural network and telemetry," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2019.

[27] A. Sadasivarao, S. Jain, S. Syed, K. Pithewan, P. Kantak, B. Lu, and L. Paraschis, "High performance streaming telemetry in optical transport networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, 2018, pp. 1–3.

[28] K. Ishii, S. Yanagimachi, A. Tajima, and S. Namiki, "Submilisecond control/monitoring of disaggregated optical node through a direct memory access based architecture," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, March 2019, pp. 1–3.

[29] J. Kundrat, O. Havlis, J. Radil, J. Jedlinsky, and J. Vojtech, "Opening up roadms: a filterless add/drop module for coherent-detection signals," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 12, no. 6, pp. C41–C49, June 2020.

[30] A. Sgambelluri, F. Paolucci, A. Giorgetti, D. Scano, and F. Cugini, "Exploiting telemetry in multi-layer networks," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–4.

[31] J. Vestin, A. Kassler, D. Bhamare, K. Grinnemo, J. Andersson, and G. Pongracz, "Programmable event detection for in-band network telemetry," in *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, 2019, pp. 1–6.

[32] A. Sgambelluri, A. Giorgetti, D. Scano, F. Cugini, and F. Paolucci, "OpenConfig and OpenROADM automation of operational modes in disaggregated optical networks," *IEEE Access*, vol. 8, pp. 190 094–190 107, 2020.

[33] J. E. Simsarian, M. N. Hall, G. Hosangadi, J. Gripp, W. van Raemdonck, J. Yu, and T. Sizer, "Stream processing for optical network monitoring with streaming telemetry and video analytics," in *2020 European Conference on Optical Communications (ECOC)*, 2020, pp. 1–4.

[34] S. Xu, Y. Hirota, M. Shiraiwa, M. Tornatore, S. Ferdousi, Y. Awaji, N. Wada, and B. Mukherjee, "Emergency opm recreation and telemetry for disaster recovery in optical networks," *Journal of Lightwave Technology*, vol. 38, no. 9, pp. 2656–2668, 2020.

[35] Apache, "Apache Kafka and kafkaStreams," https://kafka.apache.org/.

[36] A. Sgambelluri, A. Pacini, F. Paolucci, P. Castoldi, and L. Valcarenghi, "Reliable and scalable kafka-based framework for optical network telemetry," *IEEE/OSA Journal of Optical Communications and Networking*, accepted May 10, 2021.