European Conference on Optical Communication (ECOC) 2022 © Optica Publishing Group 2022

Demonstration of a Real-Time ML Pipeline for Traffic Forecasting in AI-Assisted F5G Optical Access Networks

Mihail Balanici*, Geronimo Bergk*, Pooyan Safari*, Behnam Shariati*, Johannes Karl Fischer, and Ronald Freund

Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, Einsteinufer 37, 10587 Berlin, Germany (Email: mihail.balanici@hhi.fraunhofer.de)

Abstract *We showcase a proof-of-concept demonstration of a ML pipeline for real-time traffic forecasting deployed on a passive optical access network using an XGS-PON compatible telemetry framework. The demonstration reveals the benefits of fine-granular telemetry streaming for QoS monitoring and adaptive capacity adjustment of end-customers. ©2022 The Author(s)*

Introduction

Current optical transport networks are becoming increasingly larger and more complex due to the constantly growing number of online users, traffic-generating user applications, and network devices interconnected to offer and support this dramatic growth. As a consequence of this evolution, the underlying front-, mid- and backhaul networks are expected to become even larger in the future, subsequently leading to a sharp increase in the number of network equipment units (often referred to as network elements (NE)) to be managed and monitored at all topology levels, including both, data- and optical planes [1]. As such, in order to ensure a high quality of transmission (QoT), locate and resolve network failures [2-4], detect anomalies [5] or equipment faults [6], as well as prevent traffic congestion and bottlenecks by applying throughput capacity adjustment [7] at a large scale and high network complexity, the future optical networks must support fully automated network management system (NMS) solutions. Apart from that, the current tendency towards network automation and development of next generation intelligent networking systems are only feasible when having full access to the monitoring data characterizing the network behaviour under different operational conditions and traffic intensities [8]. These objectives can only be achieved by collecting data from a variety of NE sources at various time granularities, while also extracting knowledge and deriving insight for performance monitoring, troubleshooting, and maintenance of network service continuity [9].

In this work, we perform a live demonstration of a *Telemetry Framework and Machine Learning (ML) Pipeline* running on an XGS-PON testbed located at Fraunhofer HHI premises in Berlin. The developed solution complies with the specifications laid down in the telemetry work item of the European Telecommunication Standards Institute (ETSI) Industry Specifications Group (ISG) Fifth Generation Fixed Network (F5G) [10].

The novel contributions of our demonstration are fourfold: 1) definition and development of a full telemetry streaming procedure for PONs, which includes the mapping between a PON structure and the functional blocks of the framework (e.g., broker, database, inference host) and their interfaces; 2) development of a ML-assisted traffic forecasting model that follows traffic variations in the order of 10-15 seconds with high accuracy; 3) incorporation of a ML inference host to the telemetry workflow for real-time prediction and analysis; and 4) implementation of a cloud-native version of the whole pipeline in which every individual block runs in a dedicated Docker container that allows easier deployment and automation. We will perform live demonstration of the whole solution running on the F5G testbed using XGS-PON.

Concept of End-to-End Network Telemetry

End-to-end (E2E) network telemetry [11,12] is a technology of remotely collecting device- and network-related data from the physical or virtual network components, using a set of automated processes aiming to guide network policy updates for planning, dynamic allocation and optimization of network resources, identification

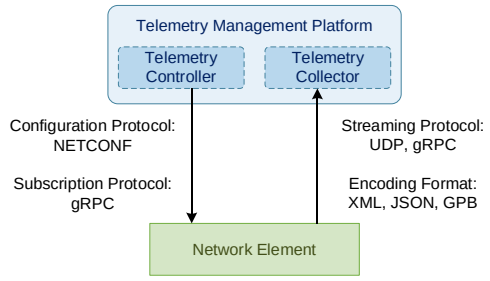

Fig. 1: Subscription and streaming of telemetry data.

* These authors have equally contributed to this work.

of service degradation, and quick localization of equipment failures and link downtimes [9]. In traditional monitoring techniques, which are preponderantly based on protocols such as SNMP, data is constantly queried (pulled) by the NMS from the NEs, incurring low data rates and high processing overhead – drawbacks incompatible with current automatic network applications. As a result, telemetry has emerged as a solution to automate network monitoring, in which the NMS subscribes to the NEs, which in turn stream their state, operation and/or configuration data proactively in a push mode.

The telemetry system/management platform consists of two main components [10]: the telemetry controller, responsible for telemetry setup and configuration on the NE side, and the telemetry collector, acting as a sink and storage of the streamed telemetry data. Worth noting is that the telemetry management system acts as a telemetry client subscribed to the telemetry server represented by the NE (Fig. 1). In this data acquisition scheme, one subscription packet typically carries multiple pieces of sampled data produced by different sources of the same network, and aggregated within one telemetry package, describing device and network operation parameters. The data is typically normalized and encoded efficiently using both, human readable formats (e.g., XML, JSON), but most commonly binary formats such as Google Protocol Buffers (Protobuf) for an efficient and automatic mechanism of data serialization for limited bandwidth utilization, reduced storage and fast transmission rates [13]. As such, both the telemetry and configuration data have model-based formats, allowing applications to configure and consume data easily. In this respect, the configuration data transmitted from the telemetry controller to the NE is often modelled using open-source and proprietary YANG data models, and is carried over the wire using the SSH-based NETCONF protocol with its XML-encoded data format [14]. On the other hand, the streamed telemetry data is commonly carried using the UDP or gRPC transport/streaming protocols, while encoded using predefined JSON or Protobuf model-based encoding formats [14].

F5G OpenLab Testbed Architecture

The optical access network testbed is a XGS-PON consisting of an optical line terminal (OLT), to which a group of optical network terminals/units (ONTs/ONUs) are linked optically through an optical splitter, sharing the same XGS-PON interface (Fig. 2). Different end-user devices are then connected to the ONTs through traditional 10G Ethernet interfaces, where the E/O conversion is carried out. These devices include fixed, thermal and pan-tilt-zoom (PTZ) cameras, laptops for simulation of video streaming services, and a traffic generator for generation of traffic flows of different nature and intensity. All end-points are configured to be part of the same VLAN, thus forming the data plane (the blue links in Fig. 2).

The OLT is also connected through a management interface to a LAN, constituting the management plane (red connections in Fig. 2). To the same LAN are also interconnected a management terminal and an Ubuntu telemetry server, on which the entire ML pipeline, discussed in more detail in the next section, runs.

When it comes to the protocol stack and data models, it is worth noting that the entire telemetry functionality is configured on the OLT through the NETCONF protocol using different YANG modules. Among these can be found such open-source models as openconfig-telemetry.yang [15], whereas the configuration of specific

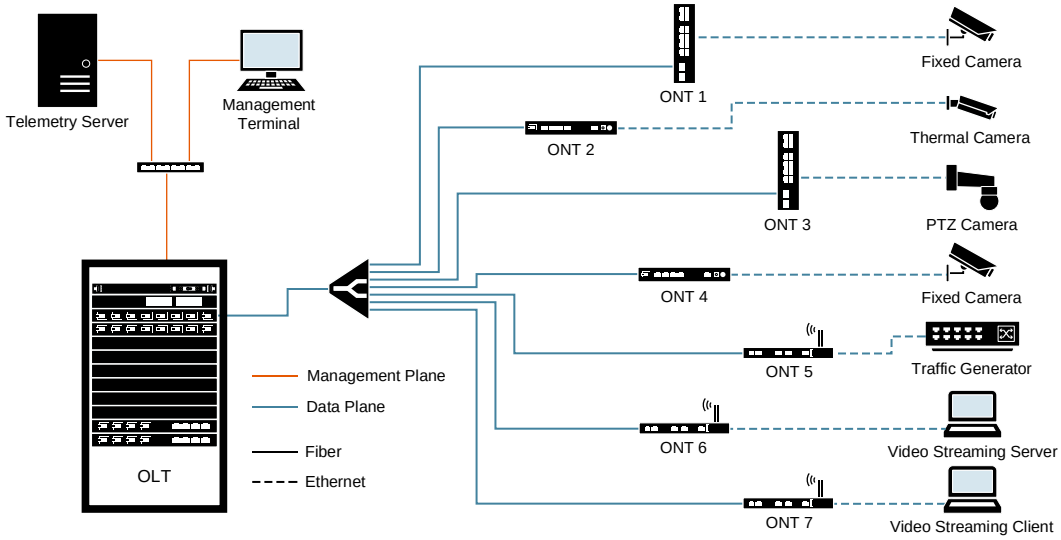

Fig. 2: The XGS-PON Optical Access Network testbed in the OpenLab facility.

telemetry functions, such as the type of streamed parameters (e.g., traffic volume or data rate), the source of telemetry data, i.e., whether the data is produced by one of the OLT's or ONT's sensors, generally require the proprietary YANG models of the hardware vendor. Finally, the configurations are carried out and visualized on the terminal PC.

Real-Time ML/Telemetry Pipeline POC

Our demonstration is carried out on the XGS-PON network testbed discussed in the previous section (Fig. 2). For demonstration purposes, the telemetry data characterizing the traffic flows on the shared XGS-PON interface is fetched from the corresponding OLT sensor. These flows represent an aggregate of different smaller traffic streams generated by the individual end devices connected to their corresponding ONTs. As such, the OLT streams its telemetry data including such traffic-related parameters as the number of TX and RX bytes, number of TX and RX packets, reception and transmission data rates, etc. With its minimal sampling period/telemetry data granularity of 5s, the OLT transmits the telemetry data encoded as Protobuf binary messages. These events enter the ML/telemetry pipeline through a customized Kafka producer written in Python (Fig. 3), whose main purpose is the Protobuf – JSON format conversion for a human readable data formatting, followed by topic initialization within the Kafka broker. Apache Kafka is a popular open-source distributed event streaming platform for high-performance data pipelines, streaming analytics and data integration [16], which has found its successful application relatively recently in telemetry of optical transport networks [11,12]. Its main component, the Kafka broker, is a distributed publish-subscribe event storage system and a robust queuing mechanism, capable to handle high volumes of data by organizing streaming events into topics, partitions and offsets. In our demonstration, a single topic is dedicated to a single source of telemetry data (i.e., sensor), and for simplicity, the topic consists of one partition only stored on a single broker (Fig. 3). In this setting, the events originating from the same source/sensor are written in the same Kafka topic. As the next step in data propagation through the telemetry pipeline, events are consumed by the Telegraf module, which is an agent for metrics, events and logs collection and reporting [17], and acts in our setting as a Kafka consumer. Its main purpose resumes to format conversion of the incoming events (JSON) into Line Protocol (LP) entries for their subsequent publication into the TSDB. Finally, the converted telemetry events are stored within the InfluxDB – a TSDB specifically designed and optimized for storage, retrieval and serving of time-stamped data, i.e., associated pairs of values and times [18] – the de-facto format of telemetry data. In this context, the InfluxDB serves as the telemetry data collector from which the ML inference host retrieves its data for the subsequent data analytics and traffic prediction tasks.

The ML inference host includes a ML forecasting model, a ML model repository and an input/output module. The model repository stores the model artifacts/hyperparameters which are used by the forecasting model for making predictions. The input/output module clients connect to the TSDB of the telemetry pipeline to query data for the forecasting model, and after the prediction procedure, to insert the predictions back into the TSDB for storage (Fig. 3). Finally, we use a Grafana module interfaced to the InfluxDB for fetching and display of telemetry data and the forecasting of the ML inference host.

Conclusions

The proposed telemetry workflow and ML pipeline offers great advantages for full automation of F5G network fabric. Our future activities focus on exploring new use-cases and their performance evaluation.

Acknowledgements

This work was partly funded by the German Ministry of Education and Research (BMBF) in the framework of the project AI-NET PROTECT (KIS8CEL010, FKZ 16KIS1282).

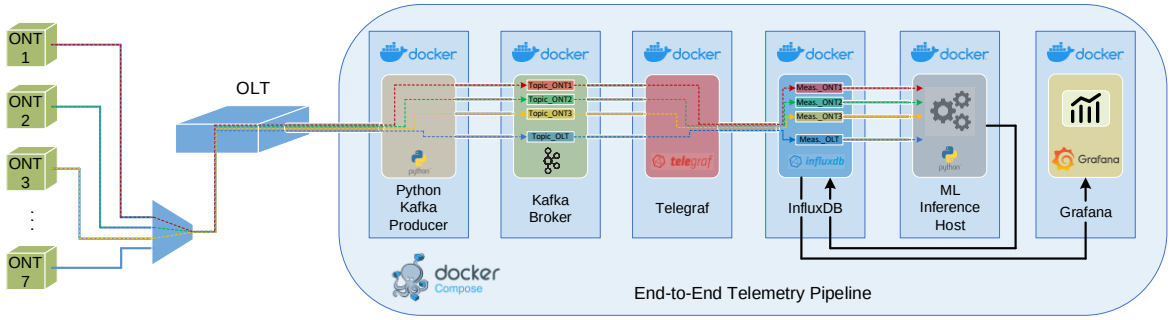

Fig. 3: The ML pipeline architecture with its containerized components.

References

[1] L. Gifre, and F. Boitier, “Role of monitoring and analytics in next generation optical networks (Invited),” *Proceedings of European Conference on Optical Communication*, 2021. DOI: 10.1109/ECOC52684.2021.9605997.

[2] 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," *Journal of Optical Communications and Networking*, vol. 10, no. 1, pp. A27-A36, 2018. DOI: 10.1364/JOCN.10.000A27.

[3] K. S. Mayer, J. A. Soares, R. P. Pinto, C. E. Rothenberg, D. S. Arantes, and D. A. A. Mello, “Machine-learning-based soft-failure localization with partial software-defined networking telemetry,” *Journal of Optical Communications and Networking*, vol. 13, no. 10, pp. E122-E131, 2021. DOI: 10.1364/JOCN.424654.

[4] F. Paolucci, A. Sgambelluri, M. Dallaglio, F. Cugini, and P. Castoldi, “Demonstration of gRPC telemetry for soft failure detection in elastic optical networks,” *Proceedings of European Conference on Optical Communication*, 2017. DOI: 10.1109/ECOC.2017.8346066.

[5] S. Nam, J. Lim, J. H. Yoo, and J. W. K. Hong, “Network anomaly detection based on in-band network telemetry with RNN,” *Proceedings of IEEE International Conference on Consumer Electronics-Asia*, 2020. DOI: 10.1109/ICCE-Asia49877.2020.9276768.

[6] J. Kundrat, M. Vasko, R. Krejci, V. Kubernat, T. Pecka, O. Havlis, M. Slapak, J. Jedlinsky, and J. Vojtech, “Opening up ROADMs: streaming telemetry [Invited],” *Journal of Optical Communications and Networking*, vol. 13, no. 10, pp. E81-E93, 2021. DOI: 10.1364/JOCN.425167.

[7] L. Velasco, S. Barzegar, F. Tabatabaeimehr, and M. Ruiz, “Intent-based networking and its application to optical networks [Invited Tutorial],” *Journal of Optical Communications and Networking*, vol. 14, no. 1, pp. A11-A22, 2022. DOI: 10.1364/JOCN.438255.

[8] D. Rafique and L. Velasco, “Machine learning for network automation: overview, architecture, and applications [Invited Tutorial],” *Journal of Optical Communications and Networking*, vol. 10, no. 10, pp. D126-D143, 2018. DOI: 10.1364/JOCN.10.00D126.

[9] C. Delezoide, P. Ramantanis, L. Gifre, F. Boitier, and P. Layec, “Field trial of failure localization in a backbone optical network,” *Proceedings of European Conference on Optical Communication*, 2021. DOI: 10.1109/ECOC52684.2021.9606152.

[10] ETSI ISG F5G, “Telemetry Framework and Requirements for Access Network,” Draft 006, Feb 2022 [Work in Progress].

[11] A. Sgambelluri, A. Pacini, F. Paolucci, P. Castoldi, and L. Valcarenghi, “Reliable and scalable Kafka-based framework for optical network telemetry,” *Journal of Optical Communications and Networking*, vol. 13, no. 10, pp. E42-E52, 2021. DOI: 10.1364/JOCN.424639.

[12] R. Vilalta, R. Casellas, R. Martinez, R. Munoz, A. Gonzalez-Muniz, and J. P. Fernandez-Palacios, “Optical Network Telemetry with Streaming Mechanisms using Transport API and Kafka,” *Proceedings of European Conference on Optical Communication*, 2021. DOI: 10.1109/ECOC52684.2021.9606002.

[13] Google Protocol Buffers (GPB, Protobuf): https://developers.google.com/protocol-buffers.

[14] 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. DOI: 10.1109/JLT.2018.2795345.

[15] OpenConfig Telemetry: https://github.com/openconfig/public/tree/master/release/models/telemetry/.

[16] Apache Kafka: https://kafka.apache.org/.

[17] Telegraf: https://www.influxdata.com/time-series-platform/telegraf/.

[18] InfluxDB: https://www.influxdata.com/products/influxdb-overview/.