Experimental Demonstration and Results of Cross-layer Monitoring Using OpenNOP: an Open Source Network Observability Platform

Nathan Ellsworth[1], **Sebastian Troia**[2], **Tianliang Zhang**[1], **Marco Tacca**[1],
Guido Maier[2], **Andrea Fumagalli**[1]
[1]*Open Networking Advanced Research (OpNeAR) Lab, UT Dallas, TX, USA*
[2]*Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy*
E-mail: nathan.ellsworth@utdallas.edu

ABSTRACT

Ensuring the smooth operation and optimal performance of communication networks requires continuous monitoring of key network elements. Network operators can detect and prevent potential issues by monitoring various real-time network parameters. This paper proposes and presents results from the implementation of a cross-layer monitoring system for OpenROADM-compliant optical transport networks using an open source network observability platform called OpenNOP, and for the first time includes simultaneous optical layer and transport layer metrics. It leverages open source tools as a cost-effective and efficient solution for network monitoring and management. OpenNOP collects and analyzes data from various network layers, including physical, data link, network, and transport layers. OpenNOP can also ingest status and log information. This data is all stored in a common time-series database. The results show that OpenNOP can provide comprehensive network visibility and effective cross-layer monitoring of OpenROADM-based networks.

Keywords: Optical transport networks, OpenROADM, network observability, eBPF, NETCONF

1. INTRODUCTION

The modern era is marked by a rapid emergence of new services and applications that demand unprecedented levels of network data, low latency, high reliability, increased connection density, and improved energy efficiency. Applications such as augmented reality, remote vehicle control, and disaggregated mobile phone networks are paving the way for the next generation of services that require high data rates and ultra-low latency networks. Cloud computing has enabled real-time, on-demand access to a wide range of applications, each with unique network requirements. As such, all layers in the network need to be agile and dynamic, working in a coordinated cross-layer fashion, including the optical transport network, to dynamically and efficiently allocate resources to support the zero-touch provisioning architecture for distributed data centers [1].

However, providing consistent real-time information for effective and efficient quality of service (QoS) monitoring of cloud applications remains a significant challenge, particularly in the context of open optical transport networks, and more specifically in the OpenROADM ecosystem [2]. In this paper, we propose an enhanced cross-layer monitoring system called OpenNOP (or NOP), which is based on our prior Network Operation Platform (NOP) [3]. OpenNOP is designed to collect cross-layer real-time network data from an OpenROADM-compliant network consisting of ROADMs (Reconfigurable Add-Drop Multiplexer), optical transponders, and other network elements from multiple vendors. This system is based entirely on open source software and is being made available for public use [4].

We will then demonstrate the efficacy of the OpenNOP system in collecting these performance metrics on a physical OpenROADM-based optical network testbed, illustrating the value of OpenNOP as a reliable and cost-effective solution for cross-layer monitoring of optical transport networks.

2. OPEN SOURCE NETWORK OPERATIONS PLATFORM (OPENNOP)

It is crucial for network operators to have a clear view of fault, configuration, availability, performance, and security (FCAPS) in order to meet the high service standards expected by customers and regulators. However, most commercial Operations/Business Support Systems (OSS/BSS) rely on proprietary and closed-source tools. To foster academic research and encourage innovation in this field, we have been progressively integrating various monitoring capabilities into our entirely open-source network observability platform (OpenNOP), as shown in Fig. 1.

Previously, we have referred to OpenNOP as the Network *Operations* Platform, where *operations* reflects the usage of such tools in a typical business operations center[1]. However, now that OpenNOP has progressed into the area of collecting various data for long-term storage and analysis, we feel it is also appropriate to refer to OpenNOP as the Network *Observability* Platform. In recent years, observability has come to represent more than just simple monitoring, but the collection and aggregation of enough data to fully characterize the behavior of a system from both an internal and external perspective [5].

Acknowledgments This material is based in part upon work supported by the National Science Foundation under Grant No. CNS-1956357 and CNS-2211989.

[1] eNOP is a name we used earlier as well. OpenNOP is now the preferred name.

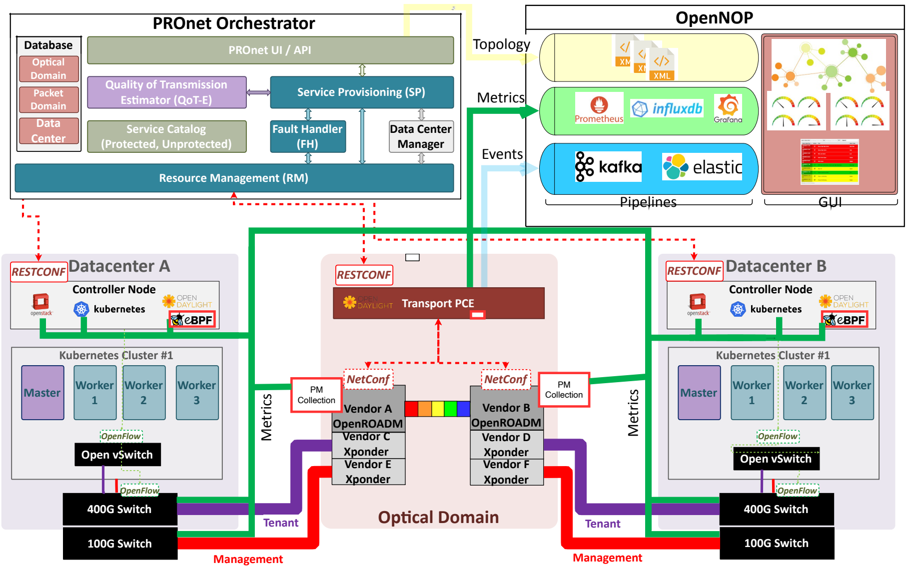

Figure 1: OpenNOP Architecture Diagram

OpenNOP is comprised of several standard open source software packages such as Grafana, InfluxDB, Kafka, Elastisearch, Prometheus, and various Prometheus *exporters*, all organized into a Docker Compose container stack. This provides a simple but robust platform on which to build collection mechanisms and retain observability data for many current and future aspects of multi-layer networking systems. Source code, installation instructions, and typical use-case examples for OpenNOP available for public use at `https://github.com/utdal/OpenNOP` [4].

The OpenNOP system can be regarded as a *data lake* constructed to contain monitoring data gathered from three distinct layers: optical, network packet, and transport. A recent definition of a data lake is "a scalable storage and analysis system for data of any type, retained in their native format and used mainly by data specialists (statisticians, data scientists or analysts) for knowledge extraction." [6] This data is readily accessible for the purpose of conducting correlation analyses.

2.1 Optical Network Layer Performance Data

The OpenROADM standard requires that compliant optical network devices publish certain PMs using the RFC 6241 NETCONF standard. OpenNOP uses a Python NETCONF library to query these PMs, including optical input and output power for ROADM and transponder links as well as transponder pre-FEC corrected error counts. The latter is important as it is proportional to the bit error rate (BER) when signal rate and forward error correction (FEC) overhead are taken into account. These PMs are stored in a Prometheus time-series database for future analysis and can also be graphed live in the OpenNOP Grafana-based UI console.

Our previous work [3] reported some problems with NETCONF session locking when using the open-source Transport PCE (TPCE) SDN controller [7] as a north/south RESTCONF-to-NETCONF translator to perform network management functions. To circumvent this locking conflict, the OpenNOP system now can directly query the OpenROADM devices using a new Prometheus exporter software package developed in our lab to query NETCONF devices for PMs.

The NETCONF PM exporter has two input configurations: an XML sub-tree filter that specifies which PMs should be gathered and list of IP addresses that should be queried. The exporter then gathers the PM XML data, parses out the data elements, and posts the data OpenNOP using the standard Prometheus line exposition format. This module will be shared with the broader community using the Github.com code repository for Prometheus [8].

2.2 Network Layer Bandwidth Data

OpenNOP uses the Prometheus SNMP exporter module to efficiently perform SNMP queries against the network layer 2 devices in the testbed. OpenNOP uses the *ifHCInOctets* and *ifHCOutOctets* 64-bit counters available on standard enterprise-grade switches to calculate the input and output data rate of each port by subtracting the previous counter value from the current sample and dividing by the sample interval. We have found that the Juniper QFX5220 switches used in our testbed do not update their internal counters more frequently than every 7 seconds, so our sample interval is 7 seconds.

OpenNOP monitors the ports used by each of the network clients and most importantly, the 400G uplink port connecting to the optical transport network via the optical transponders, enabling it to characterize the amount of network packets traversing the optical network.

2.3 Transport Layer Performance Data

A new addition to OpenNOP is use of the extended Berkeley Packet Filter (eBPF) technology to collect metrics pertaining to the TCP network transport layer. To gather this data, we developed a custom eBPF-based program capture the congestion window, receive window, round trip time (RTT), and number of re-transmitted segments. Normally, when user space software makes a kernel system call to make a network request, the OS context switch required to move from user space to kernel space is very "expensive" operation in CPU cycles. eBPF is a technology that resides in the kernel, allowing for very fast access to certain internal data structures, while at the same time providing an API to user space programs to access that data, without incurring the context switch overhead. A user space program can issue very simple eBPF scripts that the kernel eBPF components will execute. The data collected by the eBPF program allows us to analyze the behavior of TCP traffic flows and identify potential issues, such as an increase in RTT or re-transmissions caused by transmission impairment factors at the physical layer.

3. EXPERIMENTAL RESULTS

This section describes the experimental setup and collected results used to showcase the capabilities of the proposed OpenNOP as a cross-layer monitoring framework. The structure of the testbed is much like that shown in Fig. 1 with hardware compute nodes communicating across a multi-vendor OpenROADM-based optical transport network.

3.1 Testbed Setup

The multi-layer and multi-vendor testbed is shown in Fig. 2. The switch (Juniper QFX5220) aggregates the iperf network traffic of 7 compute nodes (Dell PowerEdge C8220 with dual 40G NICs) in each DC (data center) and directs it to the OpenROADM optical network. The optical transponder (400G TPDR) modulates the data onto a coherent lightwave at central frequency 196.08125 THz (DP-16QAM 63.1 Gbaud 400G signal) and sends it to the ROADM for multiplexing and routing (with channel spacing 87.5 GHz), ultimately reaching the other TPDR. This bidirectional transmission, known as a *lightpath*, is established using the open source TPCE controller.

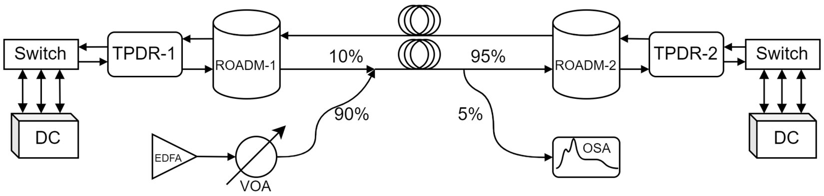

Figure 2: Multi-layer Multi-vendor Testbed Setup with Noise Injection Mechanism

To replicate typical interference affecting the optical transport network transmission, a noise injection mechanism is applied to one transmission direction of the fiber span between ROADMs. A C-band Erbium-Doped Fiber Amplifier (EDFA) is turned to its maximum drive current to generate sufficient Amplified Spontaneous Emission (ASE) noise to carry out a meaningful experiment. The ASE noise is attenuated by a Variable Optical Attenuator (VOA) setting at the cut-off point (∞ attenuation) as the initial state. As the attenuation decreases gradually, more noise is injected into the transmission line using a 90-10 coupler. 5% of the signal power in the transmission line is sent to an OSA for monitoring, while the rest goes to the destination.

The OpenROADM-compliant transponder detects the incoming coherent signal and corrects errors using oFEC. The lower attenuation is, the higher noise and BER is. As long as the received Optical Signal to Noise Ratio (OSNR) is above its critical threshold (and BER below its critical threshold), end-to-end zero-error communication is maintained. If the OSNR falls below its threshold, uncorrectable errors occur at the optical layer, leading to packet loss and re-transmission at the higher network layers.

3.2 Cross-layer Network Observations

To exercise OpenNOP in conjunction with this testbed, we designed an initial experiment to monitor the impact to a continuous 400G iperf packet stream over the ROADM link while ramping the noise up and down by decreasing and increasing the VOA attenuation. The attenuation began at 10 dB, ramped down 0.1 dB every 10s to 7.2 dB, held for 6 minutes, then ramped back up to 10 dB. The purpose is not to perform any in-depth analysis, but to highlight the current OpenNOP capabilities.

Fig. 3 (a) is the attenuation curve and the time scale for the other data elements in (b) through (e). Fig. 3 (b) shows the layer 2 switch utilization across the network, which drops off sharply once the injected noise is at its

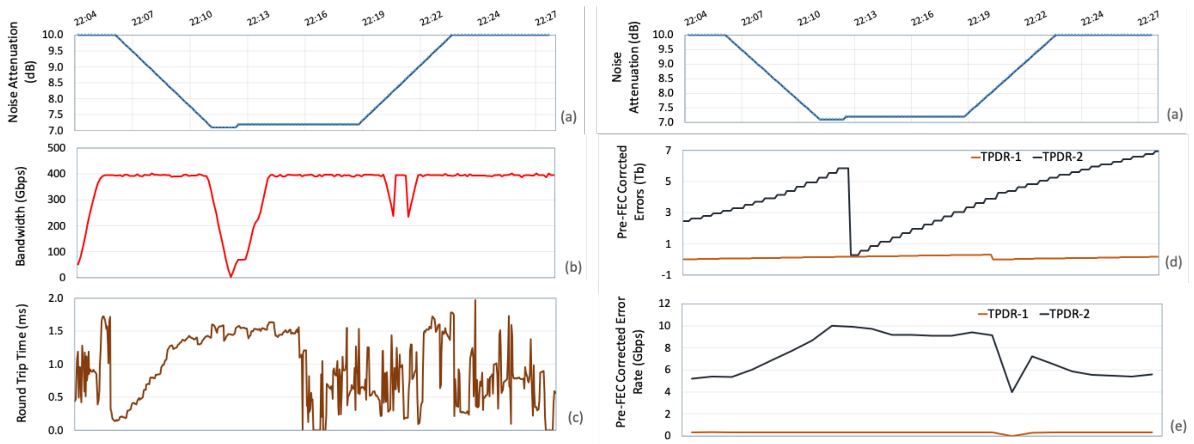

Figure 3: Cross-layer Data Capture showing (a) the gating of EDFA noise, (b) layer 2 utilization, (c) TCP RTT values, (d) pre-FEC corrected errors - accumulated and reset every 15 mins and (e) error rate

peak. Fig. 3 (c) shows the extreme variability of RTT while the noise was present. Fig. 3 (d) shows the cumulative number of pre-FEC corrected errors. This is normally a straight saw tooth wave since the OpenROADM PMs counters continually increase with a reset every 15 minutes. However, as the noise was being increased, the normally straight slope of the saw tooth is curved, meaning the rate of errors is increasing. Indeed, Fig. 3 (e), which is the delta increase (or first derivative) of the error counter, shows the error rate is increasing.

Despite the high level of noise presence, the 400G end-to-end data transfer resumes after a brief drop. During this time interval the charts report high error counts and significant fluctuations in the TCP RTT graph. This is an area requiring further investigation to determine what parts of the multi-layer network testbed allowed the data rate to recover despite the aggressive noise interference. We anticipate performing additional experiments with other types of signal perturbation over the optical transport network.

4. Final Remarks

This work described the latest enhanced features of the cross-layer monitoring system called OpenNOP. The OpenNOP system is designed to collect cross-layer real-time network data from an OpenROADM-compliant transport network consisting of network elements from multiple vendors. It will allow researchers to capture and perform data analysis on the resulting datasets to guide further exploration and lead to the development of enhanced algorithms for controlling multi-layer and multi-vendor end-to-end networks in the future. It leverages open source tools and the source code, configuration examples, and documentation will be shared with the community. We demonstrated the efficacy of the OpenNOP system in collecting various real-time performance metric datasets in on a non-emulated OpenROADM-based optical network testbed. We will continue to expand the OpenNOP capabilities and to organize and share datasets collected for various other test scenarios like that demonstrated here. We believe that the OpenNOP framework is a useful and cost-effective solution for cross-layer monitoring of optical transport networks and has the potential further insights into the operation of multi-layer networks.

REFERENCES

[1] IOWN Global Forum, “IOWN Global Forum,” https://iowngf.org/white-papers/, 2022, last retrieved 4/28/2023.
[2] OpenROADM MSA, “OpenROADM MSA,” http://OpenROADM.org, 2021, last retrieved 11/14/2021.
[3] N. Ellsworth, T. Zhang, S. Troia, G. Maier, and A. Fumagalli, “Enhancing cross layer monitoring on open optical transport networks,” in *Optical Fiber Communication Conference 2023*, 2023, pp. 1–3.
[4] UT Dallas, “OpenNOP,” https://github.com/utdal/OpenNOP, 2023.
[5] S. Almog, *Observability and Monitoring*. Berkeley, CA: Apress, 2023, pp. 175–187. [Online]. Available: https://doi.org/10.1007/978-1-4842-9042-2_9
[6] P. N. Sawadogo and J. Darmont, “On data lake architectures and metadata management,” *Journal of Intelligent Information Systems*, vol. 56, pp. 97–120, 2020.
[7] OpenDaylight Project, “Transport PCE,” https://wiki.opendaylight.org/display/ODL/TransportPCE, 2022, last retrieved 4/28/2023.
[8] “Prometheus Community,” https://github.com/prometheus-community, 2022, last retrieved 4/28/2023.