IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 15 December 2023, accepted 4 January 2024, date of publication 15 January 2024, date of current version 22 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354172

RESEARCH ARTICLE

Open Networking Engine (ONE): An Orchestration Tool for Open Optical Line System

HAFIZ MATI UR RAHMAN[1], RIDA HANIF[2], SALMAN GHAFOOR[1], AND ARSALAN AHMAD[1], (Senior Member, IEEE)

[1]Optical Networks and Technologies (ONT) Lab, School of Electrical Engineering and Computer Science (SEECS), National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[2]Military College of Signals (MCS), National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan

Corresponding author: Salman Ghafoor (salman.ghafoor@seecs.edu.pk)

This work was supported by the European Union (EU) through the Asi@Connect Project under Grant Asi@Connect-19-060.

ABSTRACT Software-Defined Optical Networking (SDON) signifies a revolutionary paradigm in optical network management and provisioning. It leverages the power of software-defined networking (SDN) principles to enhance the flexibility, efficiency, and intelligence of optical networks. In the realm of SDON, we introduce the Open Networking Engine (ONE), a groundbreaking application designed to optimize White-box Reconfigurable Optical Add-Drop Multiplexers (ROADMs). ONE serves as a vital bridge between cutting-edge technology and real-world network orchestration. By harnessing the device's YANG Models and the NETCONF protocol, it empowers network administrators with the ability to dynamically configure, provision, and monitor ROADMs. This study not only highlights ONE's competency in the efficient management of optical network resources but also underscores its effectiveness in demonstrating the potential of flex-grid technology, particularly in terms of spectral efficiency. The study also delves into the impact of channel bandwidth on critical network traffic parameters, including packet loss, round-trip time (RTT), throughput, and TCP window size. The result is a noteworthy achievement, with a 93.4% improvement in spectral efficiency realized through comprehensive signal analysis. This research highlights the promising path toward enhancing optical network performance, fostering a more agile and resource-efficient network infrastructure.

INDEX TERMS Software defined networking, software defined optical networks (SDON), open networking engine (ONE), optical line system (OLS), orchestration, configuration & monitoring, NETCONF, YANG.

I. INTRODUCTION

In an era in which connectivity is the backbone of the digital world, the relentless demand for faster, more efficient, and flexible networking solutions remains a constant force driving technological innovation. Within this landscape, Software-Defined Networking (SDN) [1] has emerged as a transformative paradigm, reshaping the way we think about network design and management. SDN's success in the realm of data centers and enterprise networks has triggered a natural extension of its principles into the optical domain, giving birth to Software-Defined Optical Networks (SDON) [2]. The fusion of SDN and optical networks introduces a new era of adaptability and intelligence to optical infrastructure, offering a solution that is poised to revolutionize how we orchestrate, provision, and monitor optical networks.

SDON's foundational premise lies in separating the network's control plane from the data plane. In doing so, it enables dynamic and centralized control over the optical network elements, granting network administrators a level of flexibility and responsiveness that was previously unimaginable. By abstracting and virtualizing the optical

The associate editor coordinating the review of this manuscript and approving it for publication was Tianhua Xu.

layer, SDON empowers administrators to efficiently allocate resources, and bandwidth, and respond to networks in real-time. These attributes are vital in an age where bandwidth-hungry applications, such as video streaming, cloud computing, and the Internet of Things (IoT) continue to drive data consumption to unprecedented levels [3]. While SDN's influence on traditional networks is well-documented, its entry into optical networking presents distinct challenges and opportunities. Optical networks, known for their vast and intricate infrastructure, possess unique characteristics that require specialized solutions. Unlike electrical signals, optical signals are carried via light which is highly sensitive to factors like wavelength, signal quality, interference, noise, and optical fiber types. SDON stands as the answer to these intricacies by offering dynamic resource utilization and management. It harmonizes the traditionally rigid optical networks with the agility of software-defined control [2].

The primary goal of this study is to present the design and architecture of Open Networking Engine (ONE) which is a pioneering application that leverages the SDON principles and provides a single pane of glass management for open optical line systems. ONE bridges the gap between state-of-the-art technology and real-world network orchestration.

The main contributions of this article are:

1) **Design and Implementation of ONE:** The design and implementation details of ONE, a pioneering application built on Software-Defined Optical Networking (SDON) principles.
2) **Monitoring Open Optical Line System (OOLS) using ONE:** Introducing ONE as a powerful tool for monitoring SDON, offering network administrators insights into the dynamic behavior and performance of the optical network.
3) **Configuration and Management of OOLS using ONE:** Showcasing the role of ONE in the configuration and centralized management of SDON, providing administrators with the capability to dynamically control and orchestrate network elements.
4) **Efficient Resource Utilization and Spectral Efficiency using ONE:** Demonstrating ONE's efficacy for efficient resource utilization, specifically emphasizing spectral efficiency as a use case. Highlighting how ONE optimizes the allocation of resources to achieve a more streamlined and effective network operation.

The article is structured as, Section II provides a background on relevant technologies, Section III delves into prior research in the field, Section IV outlines the proposed architecture, Section V offers insights into the experimental setup, Section VI presents results and demonstrations, and Section VII concludes the study.

II. BACKGROUND

Software Defined Networking has become increasingly popular and advanced in recent years. SDN defines a clear separation of control and data planes to ease network management and service roll-out, while also maintaining its roots in earlier stages of programmable network development [4]. The introduction of open interfaces, such as OpenFlow, had substantially increased the scope for creativity in designing controller platforms e.g ONOS, ODL, Floodlight, RYU and networking applications e.g L2 and L3 routing, firewall, IDS, IPS, and Load-Balancers, etc. This led to the easing of constraints imposed by earlier networks, which were tailored for a limited range of telecommunications services [5], [6].

A. SOFTWARE DEFINED OPTICAL NETWORKS

Software-Defined Optical Networks (SDON) is an extension of traditional optical networks that incorporate principles from Software-Defined Networking (SDN) to provide greater flexibility, agility, and control over optical resources. The Reconfigurable Optical Add/Drop Multiplexer (ROADM) or cross-connect is a fundamental component used in SDON to enable dynamic optical circuit switching [7].

Network design choices are reflected in the selection between black-box, gray-box, and white-box ROADMs. Black-box ROADMs are shipped with proprietary control and management software leading to vendor lock-in, restricting the flexibility and interoperability of network operators. On the contrary, gray and white-box ROADMs support disaggregation, provide open interfaces, and follow industry standards, giving network operators more flexible design choices [8].

Disaggregation is a key aspect of Software-Defined Optical Networking (SDON), opening up new possibilities and conceptual breakthroughs by providing crucial qualities including scalability, dependability, and efficiency [9], [10]. Disaggregation entails the composition and integration of open and readily available components, devices, and sub-systems to construct optical infrastructures and networks. This approach involves assembling a combination of "best-in-class" devices tailored to specific needs. Within the context of optical transport networks, various models are viable under this disaggregation trend, offering more flexibility, reconfigurability, and elasticity in network architectures [11], [12]. For instance, there is an option of partial disaggregation, where the control of transceivers and terminal devices is separated from the control of the open optical line system (OOLS). Alternatively, a full disaggregation model, relying on white-boxes, allows different optical network elements, such as ROADMs, transponders, line amplifiers, etc., to be sourced from different vendors [13]. This diverse spectrum of disaggregation models provides network architects with the flexibility to choose an approach that aligns with their specific requirements and optimization goals.

B. YANG DATA MODEL

Network Automation plays an essential role in managing and configuring network devices. YANG (YET Another Next Generation) data model language is a tool for automating tasks such as network configuration, maintenance, and troubleshooting. YANG provides rules for defining the

structure and behavior of network elements. It offers a way to represent data models for various network configuration protocols, like NETCONF (Network Configuration Protocol) RESTCONF (RESTful Network Configuration Protocol), and gNMI (gRPC Network Management Interface) [14]. YANG data models consist of modules and sub-modules, which define the structure, constraints, and behavior of the data. However, modules can reference definitions in other modules and sub-modules using the *import* and *include* statements [15]. A YANG module is the fundamental building block of a data model. It contains data definitions, hierarchical organization, and constraints on the data. Each module is identified by a unique namespace URI (Uniform Resource Identifier), which ensures the uniqueness of the module's name [16].

Concurrently, efforts are underway to establish a standardized abstraction for data plane devices. Notably, industry initiatives such as OpenConfig [17] and OpenROADM [18] are actively contributing to the community by defining multi-source agreements and YANG models [19]. These initiatives play a pivotal role in creating a foundation for SDN controllers to consistently control, configure, and monitor optical networks using standard YANG models and procedures. It's noteworthy that major network vendors have already embraced this paradigm, incorporating adopted (though not yet standardized) YANG models into their products and documentation.

C. SDN INTERFACES AND PROTOCOLS

SDN interfaces are critical components in SDN systems that provide communication, control, and programmability. These interfaces allow for interactions between various SDN elements, enabling dynamic network setup and administration using various network communication protocols [20], [21]. There are different protocols introduced as SDN interfaces e.g. OpenFlow, RESTCONF, NETCONF, gRPC, and gNMI etc.

OpenFlow specifies a standardized protocol for communicating with OpenFlow-enabled switches or routers via SDN controllers [6]. NETCONF is an IETF-standardized protocol for carrying configuration and management data to and from network devices. It uses XML as data format and SSH or TLS for secure communication. NETCONF provides a rich set of functionalities including configuration changes, rollbacks, and authentication [22]. RESTCONF is also an IETF protocol that implements RESTful principles over HTTP(S) for network device management. It provides a simple and web-friendly approach, using XML or JSON for data representation. RESTCONF provides CRUD (Create, Read, Update, Delete) operations for configuration making it an easy choice for developers [23]. gNMI is a protocol designed for efficient network management, using gRPC as a transport tool and protocol buffers for data encoding. It excels in collecting and streaming high-frequency telemetry data and provides an interface for real-time network monitoring [24]. gNMI supports a variety of functions including establishing and registering telemetry data. It is commonly used in large networks and modern network monitoring solutions for its performance benefits [25].

To address challenges related to the widespread commercial adoption of standard YANG models as discussed in the previous section, the research community has proactively launched initiatives [26] which focus on the development of NETCONF-based software agents, acting as translation points between SDN controllers and various optical devices, whether commercial or experimental. This collaboration is becoming integral to ensuring seamless multi-vendor support and advancing the interoperability of optical networking technologies.

III. RELATED WORK

Considering the evolving landscape of optical networking, it's essential to look into recent studies. In this section, we'll perform a concise review of the relevant literature, discussing how these works relate to and differentiate from this research.

The authors in [27] introduced a distributed telemetry architecture using gRPC. While their work closely aligns with this research in terms of monitoring concepts, there is a significant distinction. The contributions of this study lie in presenting an open networking engine, which not only facilitates monitoring but also offers capabilities for the orchestration and configuration of ROADMs. Furthermore, we have plans to expand ONE's functionalities to encompass telemetry. It's worth noting that their work utilized synthetic data sources to illustrate their concepts, whereas ONE is purpose-built for real optical elements, as elaborated in Section V.

In [28] the authors conducted optical x-Haul emulation by leveraging mininet-optical [29] and introduced an SDN controller for dynamic provisioning within the emulated network. When compared to SDN controllers their roles and responsibilities we can consider ONE as an alternative SDN controller capable of provisioning and monitoring lightpaths and different components of ROADMs as well. This research differs from [28] primarily concerning the application focus, while their study delved into investigating the processing limits of baseband units (BBU), this work focuses on ONE and its usefulness e.g exploring the impact of channel bandwidth on transmission quality through varying numbers of TCP sessions as a use-case. The study conducted by the authors in [30] introduced an SDN-based application that utilizes OpenFlow [5] to implement resource allocation functions within the wavelength division multiplexing (WDM) layer. To achieve this, they harnessed the power of the optical switch fabric's mirror [31] and created simulations of a CLOS data center topology within mininet. In [32] the authors made significant enhancements to the ONOS SDN controller [33] and provided drivers for lumentum ROADMs, enabling the setup of network connectivity using these drivers. Their research represents a fundamental step

in pioneering the concept of using an SDN controller for optical networks. ONE is inspired by this research and it is currently implemented as a standalone application. However, we have intentions to integrate ONE with the ONOS SDN controller in the future. The authors in [34] demonstrated the utilization of ONOS as a control plane for multi-domain disaggregated transport networks based on OpenROADM MSA. Similar to the previously mentioned works, this research represents an early effort to implement proof-of-concepts for SDN-based optical control. The work in [35] introduced a solution to establish compatibility between two well-known optical device models, OpenROADM and OpenConfig. This was achieved by transforming patterns and rules from one model to another, ensuring their seamless integration.

One of these notable studies [36] introduces a real-time life path network survivability using an Intent-Based Networking (IBN) framework. This framework encompasses three layers dedicated to data collection, analysis, and management. Leveraging mininet-optical and BMv2 as an emulator, it employs a single REST server to streamline the configuration of ROADMs and terminals. Furthermore, it adopts a NETCONF server for each ROADM to facilitate input and output power monitoring. The framework also incorporates a Translator and Engine to oversee the intents, ensuring real-time functionality and backup path throughput, especially during protection switching scenarios. This demonstration prominently features IBN, offering artificial intelligence (AI) based natural language processing (NLP) and translation capabilities. These capabilities hold great potential for exploring advanced customer-operator automated interactions. The use of AI is beyond the scope of this work as we have focused on the provisioning and monitoring of the commercial lumentum ROADMs so far. Similarly, another work [37] emphasizes the potential of software-configurable optical networks that leverage digital sub-carrier multiplexing (DSCM). This study investigates various network topologies, including point-to-point (P2P) and point-to-multipoint (P2MP) transceivers. The primary objective is to reduce capital expenditures (CAPEX) and enhance port efficiency. In [38], a novel approach is presented, offering a disaggregated framework for generating a digital twin of the physical layer, termed PHY-DT. Their methodology revolves around the configuration of a triangular optical network structure, where each node is loaded with off-the-shelf transceivers (TRXs) and Reconfigurable Optical Add-Drop Multiplexers (ROADMs) using an ONOS (Open Network Operating System) controller [34] alongside GNPy, facilitating the modeling of the physical layer within the PHY-DT [39], [40]. The configuration is managed through NETCONF for the OpenConfig YANG data models to abstract the network's topological characteristics.

In [41] authors explored the utilization of hybrid packet-optical nodes and conducted an assessment of two hierarchical strategies: Shar and Exec, for governing SDN (Software-Defined Networking) controllers. The outcomes of this study reveal that the Shar workflow surpasses the Exec approach in both speed and efficiency. This discovery stands as a pivotal cornerstone in the realm of managing packet-optical nodes that feature coherent pluggable transceivers, thus extending support for a diverse range of transmission applications. The work reported in [42] introduces a network operation platform tailored for OpenROADM environments, with a particular emphasis on delivering FCAPS (Fault, Configuration, Accounting, Performance, and Security) functionalities to equipment conforming to OpenROADM standards. This platform holds substantial value for both network operators and equipment vendors. The work in [37] presents an upgraded network operation platform with extended capabilities designed for diverse multi-vendor optical networks adhering to the OpenROADM standards. The demonstration highlights the challenges encountered during the implementation of automated path restoration mechanisms, alongside the continuous monitoring of crucial network elements at the receiver.

For a more in-depth understanding of this topic, we recommend the readers should refer to a comprehensive survey on Software-Defined Optical Networks (SDON) conducted by [7]. This survey delves into the role of SDN in optical networks, explores the orchestration of multi-layer and multi-domain SDONs, and thoroughly discusses the open challenges and opportunities within this domain.

IV. PROPOSED ARCHITECTURE

This section provides a comprehensive overview of the Open Networking Engine (ONE) and its architecture. The proposed architecture bridges the realms of Software-Defined Networking (SDN) and open optical networking, leveraging the synergy between these technologies. Open Networking Engine is a vendor-agnostic orchestration tool for white-box/gray-box ROADMs. Figure 1 shows the architectural framework of ONE. We have placed all the code for ONE on the GitHub repository [45] and is available publicly. The following subsections describe the components of the ONE architecture.

A. ABSTRACTION AND CONTROL OF ROADM

ONE is designed for the orchestration and monitoring of white-box/gray-box ROADMs from any vendor. In our testbed, we utilize the lumentum ROADM-20 gray-box available in the LAB (ONT) [46] as discussed in section V. Figure 6 represents a detailed view of the internal components of the ROADM and power monitoring points at different stages. There are four main components wavelength selective switch (WSS) containing multiplexer and de-multiplexer, booster (EDFA-1), preamplifier (EDFA-2), and optical channel monitoring (OCM) unit. This is a single-degree ROADM with a single input/output line. The OCM reports per channel power (red dots) and total power (green dots) information.

Every component of ROADM shown in Figure 6 can be configured and monitored via its respective YANG data

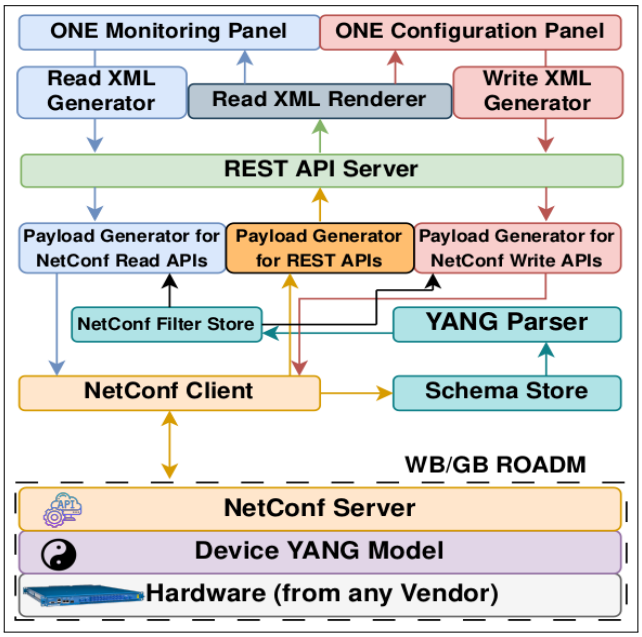

FIGURE 1. ONE architectural block diagram.

model. The YANG data model provides an abstraction of the device through its modules. The lumentum ROADM-20 YANG model is a mix of standard and lumentum-specific modules as listed in Table 1. While space constraints limit an exhaustive explanation of each module, interested readers can find detailed information in the lumentum NETCONF guide. Although the discussion made in this section is very specific to devices from a particular vendor, however, the general principle applies universally. ONE exploits ROADM's YANG data model and NETCONF server available on the device for orchestration and monitoring of the ROADMs. The YANG parser, as explained in subsection IV-E, plays a pivotal role in ensuring ONE's vendor-agnostic capability, enabling its seamless integration with ROADMs from different vendors.

TABLE 1. Lumentum YANG modules.

Module	Description
IETF Modules	These are IETF standard modules that define standard interfaces and NETCONF-based configuration and monitoring modules e.g ietf-netconf, ietf-netconf-monitoring, ietf-netconf-notifications, and ietf-netconf-server, etc.
Lumentum System Modules	These are the lumentum-specific system-related modules e.g lumentum-alarms, lumentum-perfmon, lumentum-rpc, and lumentum-sw-upgrade, etc.
Optical Transport Entity (ote) modules	The OTE modules are also lumentum-specific modules that define interfaces for the transport part of the ROADM e.g lumentum-ote-connection, lumentum-ote-edfa, lumentum-ote-fru and lumentum-ote-ocm, etc.

B. ONE MONITORING PANEL

The main monitoring panel displays real-time graphs for 6 most recent samples of power in dBm at MUX input ports, DeMUX output ports, Booster input & output, and Pre-Amplifier input & output as shown in Figure 2. The user can select input port(s) for MUX and output port(s) for DeMUX. There are options for visualizing the data component-wise as can be seen in the left sidebar under the ''Monitoring'' section. The data samples are gathered through periodic REST-POST requests, initiated at user-defined intervals with a default setting of 3 seconds. The POST request is received by the REST API server which calls the NETCONF read API payload generator as shown in Figure 1. The choice to use REST-POST with a default polling interval of 3 seconds in this implementation was made considering several factors which include device sweep time, processing, and data transfer as well as the allowance for the optical selector switch to settle as mentioned in the lumentum product guide, the refresh interval is 2 seconds for MUX and DeMUX.

The payload generator in the proposed system is a vital component responsible for creating payloads for NETCONF clients based on the read or write operation specified in the REST-POST call. The REST API server then invokes the corresponding payload generator module to generate a payload for the NETCONF client. The parameters and the operation to perform on the device are part of the REST-POST call. The payload generator makes use of this information and includes the corresponding NETCONF filter from the NETCONF filter store generated by YANG parser as described in the YANG parser section IV-E below. The generated payload is then forwarded to the NETCONF client. The working of the NETCONF client is described in section IV-D below.

C. ONE CONFIGURATION PANEL

The configuration panel contains dedicated forms for configuring four key components of ROADMs i.e. Booster, Pre-Amplifier, WSS MUX Channel, and WSS DeMUX Channel. ONE provides a configuration option for each writeable parameter of these components as per the YANG specification of the device. Figure 3 illustrates the configuration form for the EDFA (Booster), where users can set Name, Maintenance State, Control Mode, Gain Switch Mode, Target Gain (dB), Target Power (dBm), Tilt Gain (dB) and Alarm Thresholds etc. EDFAs support two control modes i.e. constant power mode (default mode) and constant gain mode. In constant power mode, the EDFA maintains a constant total output power by automatically adjusting the pump laser bias current while in the constant gain mode, it maintains a constant gain by adjusting the pump bias current based on the optical input power. The WSS channel configuration form for MUX input ports and DeMUX output ports can be used to set up an optical channel/connection by providing connection ID and start & end frequencies with a granularity of 6.25 GHz and per channel attenuation in dB. When a user submits a request to update/create a configuration from the ONE configuration panel a REST-POST method is called which carries user-defined configurations. The REST

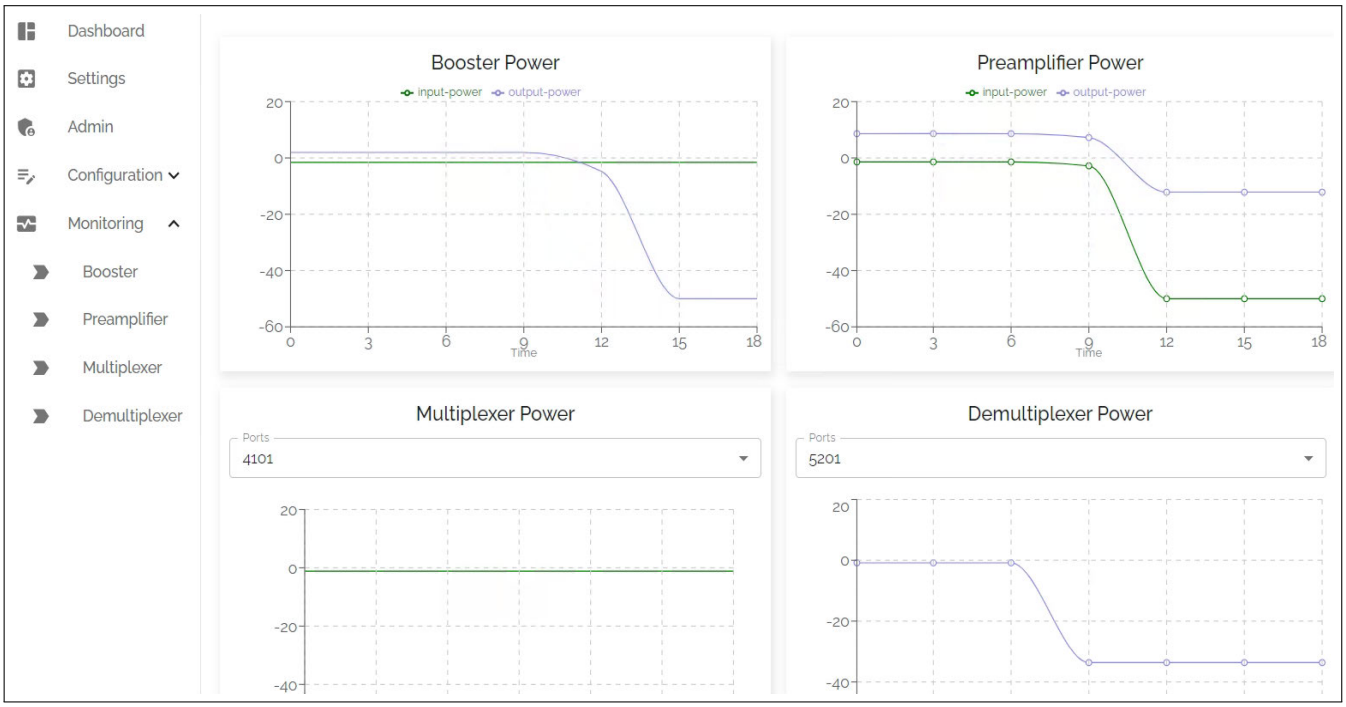

FIGURE 2. ONE monitoring panel. (Displaying input/output power for EDFAs and MUX/DeMUX.)

API server receives the request and calls NETCONF to write the API payload generator as shown in Figure 1. The NETCONF write API payload generator identifies the requesting component based on information received from the REST-POST call includes the corresponding NETCONF filter from the NETCONF filter store and forwards this payload to the NETCONF client for further processing.

We have shown the configuration panel for booster only due to scarcity of space however, users can easily access the configuration forms for each component by navigating to the left side-bar under the "Configuration" section as shown in Figure 3.

D. NETCONF CLIENT

The NETCONF client serves as a key component within ONE, facilitating the connection with the device to read and write the configuration and operational state of various ROADM components. In the implementation, we leverage the NETCONF manager from the Python *ncclient* library to establish a secure NETCONF over SSH communication channel between the ROADM and ONE. This channel ensures security through encryption of all the traffic using the AES-128 algorithm. The NETCONF client encapsulates the user request in an XML object and sends it to the NETCONF server [43]. The NETCONF client uses the *get-config* method to fetch state data from the device and the *edit-config* method to set the parameter values of the configurational data. The monitoring steps are outlined in Figure 4(a), presented as a flow chart, and Figure 4(b) illustrates the sequence of steps for the configuration process.

While we acknowledge the existence of alternative methods such as gRPC and the extensive literature on telemetry in SDN-controlled optical networks, our choice was deliberate to ensure a straightforward and universally applicable solution. The REST/POST approach aligns with widely adopted practices, providing compatibility with a diverse set of devices, including those that may not support gRPC like the one we have in our testbed which only supports NETCONF APIs. We believe that the selected approach strikes a balance between simplicity and efficiency, making it accessible and applicable to a broader range of open optical networking scenarios. However, we acknowledge the ongoing advancements in the field and are considering the merits of different telemetry methods in future work.

E. YANG PARSER

The NETCONF client fetches the YANG model from the device using the *get-schema* method and places it in the schema store. A user can manually submit the YANG model of the device from the GUI as well. The YANG parser module applies its logic, detailed in Algorithm 1, to process the YANG modules from the schema store. This process results in the creation of XML filters for the NETCONF client which are subsequently placed in the NETCONF Filter Store. Following is the line-by-line description of Algorithm 1:

Line 1: Define the `ParseYANGFile` function. This function uses the `pyang` library to parse the YANG file and returns the parsed module.

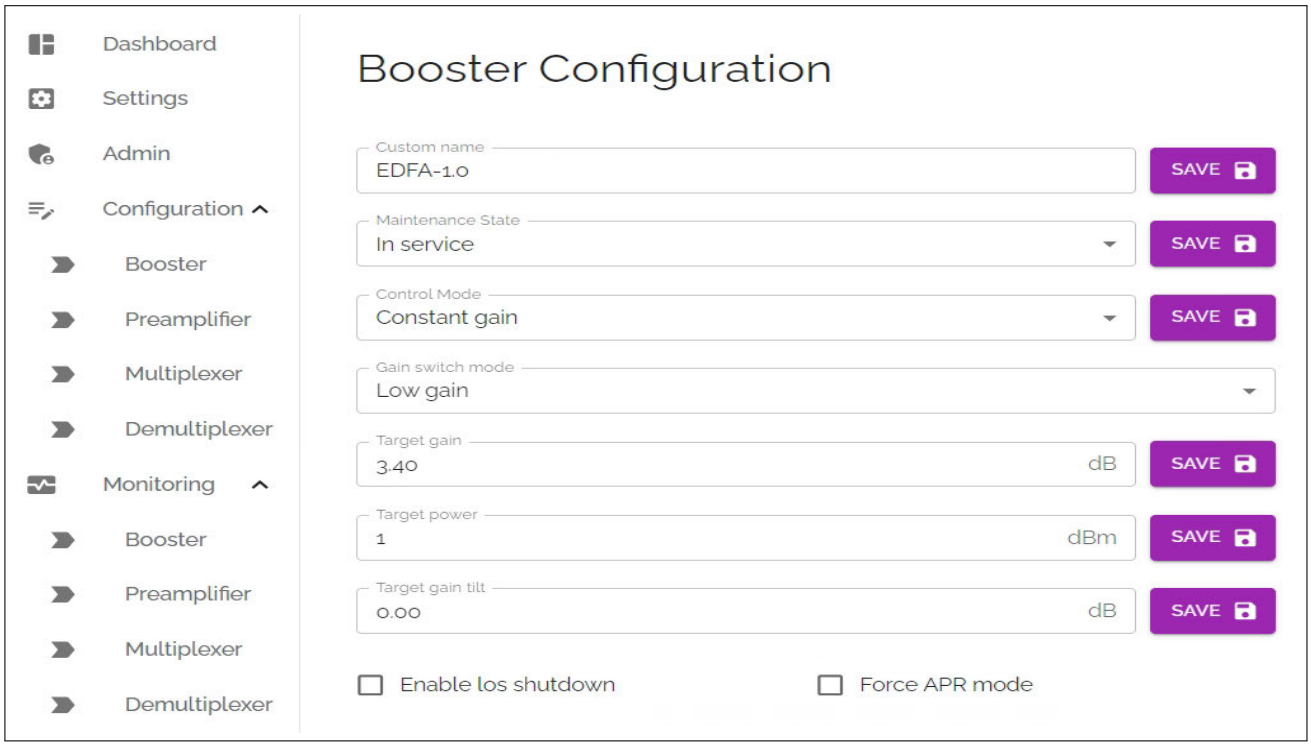

FIGURE 3. Booster configuration panel.

Line 2: A new pyang context (ctx) is created using the Pyang.Context() constructor. This context is a fundamental component for handling YANG modules with the pyang library.
Line 3: A new module is added to the context (ctx) using the ctx.add_module method. The module is added based on the YANG file specified by the yang_file_path parameter.
Line 4-5: The parse method is called on the module. This step is crucial for parsing the YANG module and preparing it for further processing. The parsed YANG module is then returned from the function.
Line 6: Define the `ExtractDataNodes` function. This function extracts relevant data nodes (e.g., containers or lists) from the parsed YANG module.
Line 7: The function initializes an empty list named data_nodes to store information about relevant data nodes (e.g., containers or lists) within the YANG module.
Line 8-10: Iterate over each child node (m) in the module's children (module.i_children). Each child node represents a part of the YANG module's structure. Each child node is checked, if it is either 'container' or 'list'. If the condition is met, the node_name (the argument of the node) and node_type (the keyword of the node, which is either 'container' or 'list') are extracted from the module.
Line 11-12: The extracted information (node_name and node_type) is then encapsulated in a dictionary and appended to the data_nodes list. The final list of data nodes is then returned by the function.
Line 13: Define the `GeneratePayloadTemplate` function. This function generates a NETCONF payload template based on the extracted data nodes.
Line 14: The function initializes an empty string variable named payload_template to store the generated NETCONF payload template.
Line 15-19: Iterate over each data_node in the data_nodes list. For each data_node, the node_name and node_type are extracted from the dictionary. Using the extracted information, a string to the payload_template variable is appended. The string represents a NETCONF element in the form of <node_name>node_type</node_name>. The function utilizes a line break (\n) at the end of each appended string for better readability. The final payload_template string, representing the entire NETCONF payload template, is returned by the function.
Line 20-24: These lines of code demonstrates the step-by-step process of processing a YANG file. It starts by specifying the file path, parsing the YANG module, extracting relevant data nodes, generating a NETCONF payload template based on the extracted information, and finally, printing or using the resulting template. The sequence of function calls showcases how the different functions in the algorithm work together to convert a YANG file into a usable NETCONF payload template, facilitating interaction with network devices.

While the monitoring and configuration panels presented in this section illustrate a single-degree ROADM with a single line-in and line-out, it's essential to note that the back-end

Algorithm 1 YANG Parser Logic

```
Data: YANG File Path
Result: Generated NETCONF Payload Template
Function ParseYANGFile(yang_file_path)
    ctx ← pyang.Context();
    module ← ctx.add_module(None, yang_file_path);
    module.parse();
    return module;
Function ExtractDataNodes(module)
    data_nodes ← [] foreach m in module.i_children do
        if m.keyword == 'container' or m.keyword == 'list' then
            node_name ← m.arg;
            node_type ← m.keyword;
            data_nodes.append({'name' : node_name, 'type' : node_type});
    return data_nodes;
Function GeneratePayloadTemplate(data_nodes)
    payload_template ← "";
    foreach data_node in data_nodes do
        node_name ← data_node['name'];
        node_type ← data_node['type'];
        payload_template += "<" + node_name + ">"+node_type+"</"+node_name+">\n";
    return payload_template;
yang_file_path ← "path/to/your/module.yang";
parsed_module ← ParseYANGFile(yang_file_path);
data_nodes ← ExtractDataNodes(parsed_module);
payload_template ← GeneratePayloadTemplate(data_nodes);
Print or Use: payload_template;
```

of ONE is not restricted to a specific ROADM type, such as single-degree or 2-degree, etc. ONE operations are based on the device's YANG model, which makes it agnostic to the type and vendor of the device. In essence, the back end of ONE autonomously adjusts itself according to the provided YANG model. Notably, we avoided hard-coding specific details related to single-degree ROADMs as well, ONE has adjusted its back-end as per the lumentum ROADM-20 YANG model. The front-end GUI is initially designed for a single-degree ROADM. Acknowledging the absence of a one-size-fits-all solution for effectively representing diverse device types, ONE possesses the flexibility to undergo enhancements, allowing seamless adaptation to 2-degree, 3-degree, and other ROADMs. This adaptability is facilitated by the REST API server positioned between the front-end and back-end.

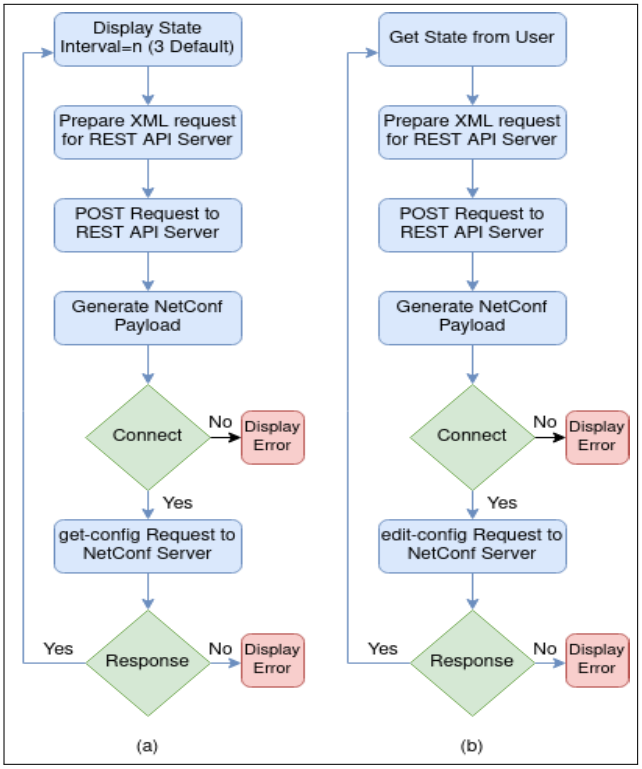

FIGURE 4. ONE flow chart: (a) Monitoring flow (b) Configuration flow.

TABLE 2. Equipment details.

Item	Description
SDN Control Plane	**Server:** Dell R620, 128GB RAM, 1.7TB DAS **OS:** Hypervisor *VMware ESXi7.02*, Guest-OS *Ubuntu 22.04* **Application:** *Open Networking Engine (ONE)*
ROADM	Lumentum ROADM-20
End-Nodes	**System:** Core-i7 9th Gen, 32GB RAM, 512 SSD **OS:***Ubuntu 22.04* **Ethernet Controller:***Intel x710 Dual Port 10G SFP+*
SFP	Huawei Optical Transceiver 1550nm, 10Gb/s, LC, SM, 80 Km
Fiber Spool	Corning China 25 Km, SMF G.652D, SC/PC

V. EXPERIMENTAL TESTBED

The experimental testbed is a software-defined optical transport network consisting of SDN-controlled optical elements which may be called an infrastructure plane and an SDN control plane. The infrastructure plane comprised of two ROADMs, connected via a 25 Km fiber spool and two end nodes (Node-A & Node-B) equipped with SFPs as shown in Figure 6. The details of the equipment are summarised in Table 2. To provide a comprehensive understanding of the experimental setup, a real picture of the testbed deployed in the laboratory is presented in Figure 5. The testbed comprises a top-of-rack switch, two Edgecore-CASSINI transponders,

two lumentum ROADM-20, two End-Nodes, and two 25 Km fiber spools. The image offers a tangible glimpse into the physical infrastructure supporting our research efforts. The ROADMs have 20 input lines connected to a 20 × 1 optical multiplexer (MUX), the output of the MUX is fed to a booster (EDFA) which amplifies each channel in the multiplexed signal by the same amount. The multiplexed optical signal is fed to a variable optical attenuator to control per-channel optical power in the output signal. The optical signal received at the Line-In port is fed to Pre-Amp (EDFA) which amplifies each channel in the received signal by the same amount. The output of the Pre-Amp is fed to a 1 × 20 optical de-multiplexer (DEMUX) which separates out channels to drop and connect them to one of 20 output lines. The wavelength selective switching (WSS) component connects the remaining channels to the MUX so that they can pass through the ROADM. All the ports on the ROADMs are LC ports so an LC-SC connector is used to connect the fiber spools with the ROADMs.

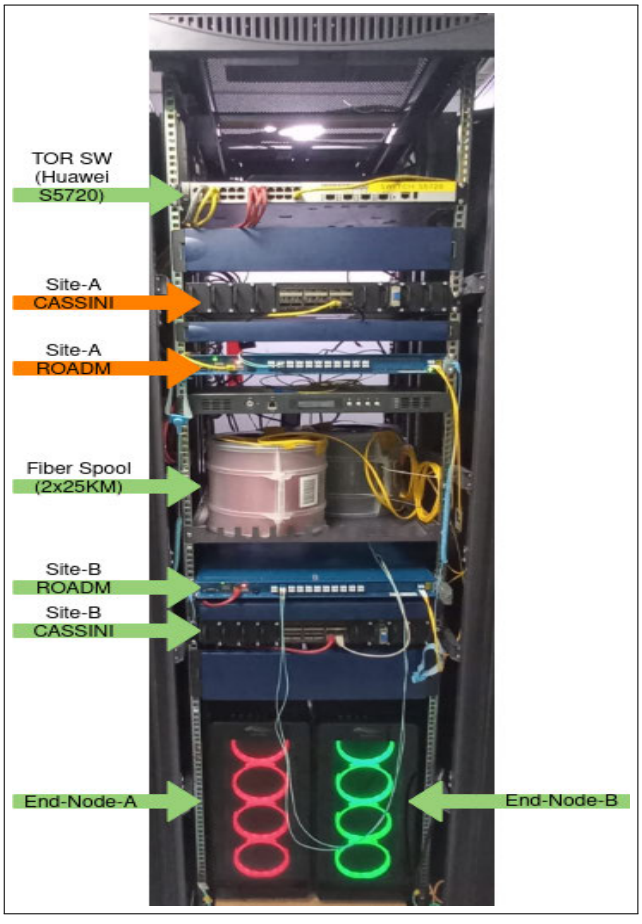

FIGURE 5. ONT [46] open optical network testbed.

The End Nodes are equipped with an Intel Core-i7 9th Gen processor, 32GB RAM, 512GB SSD, and running Ubuntu 22.04 operating system. The End Nodes are also equipped with an Intel x710 Ethernet controller consisting of dual 10G sfp+ ports. The TX side of SFP on Node A/B is connected to ROADM-A/B input line 1 and the RX side of SFP on Node A/B is connected to ROADM-A/B output line 1 via 5 meter LC-LC fiber patch cables. The specifications of the SFPs are summarised in Table 3.

TABLE 3. SFP specifications.

Item	Description
Transmission Rate	10Gb/s
Connection Type	LC
Transmission Distance	40 Km
Optical Fiber Type	SMF
Working Wavelength Range of Optical Transmitter (nm)	1530 - 1565
Working Wavelength Range of Optical Receiver (nm)	1260 - 1600
Transmit Optical Power Range (AVG) (dBm)	(-4.7) - (4)
Receiving Sensitivity (AVG) (dBm)	-14.1
Saturated Optical Power (AVG) (dBm)	0.5

The SDN control plane is running in a virtual machine (VM) on the Dell R620 Server. We have installed VMware ESXi 7.02 hypervisor on the server and the Ubuntu 22.04 guest OS on the Virtual Machine. The proposed Open Networking Engine (ONE) application as described in section IV is running on the Virtual Machine. ONE application acts as an SDN control plane that configures, provisions, manages, and monitors the SDN-controlled optical elements in the infrastructure plane via NETCONF APIs.

A. CONFIGURATION OF END TO END LIGHTPATH VIA ONE

The lumentum ROADM-20 operates in the C-Band, therefore we used SFPs which transmit in the C-Band but the exact frequency transmitted by the SPFs is unknown. Optical Spectrum Analyser (OSA) is normally used to observe an optical signal. As our testing environment lacks OSA, so we opted to utilize ONE as an OSA to analyze the optical signal transmitted by SFPs in both End Nodes. We configured 48 possible channels on port 1 (4101) of the ROADM with a channel bandwidth of 100 GHz using ONE configuration panel for multiplexer, we did not include images for all configuration panels due to scarcity of space, however, all the options under the configuration section can be seen in Figure 3. We monitored the received signal power for each channel using the ONE monitoring panel for multiplexer which provides information about received signal power for each channel. The SFP in Node-A transmits a 100 GHz channel from 193425.00 GHz to 193525.00 GHz having a total signal power of −0.3dBm and the SFP in Node-B transmits a 100 GHz channel from 193525.00 GHz to 193625.00 GHz having a total signal power of −1.8dBm as shown in Table 5. To configure an end-to-end lightpath on ROADMs we need to know what signal power is received, what should be the channel bandwidth, what is the saturation power (maximum signal power a receiver can uphold), and what is the receiving sensitivity (minimum receivable signal power) of the receiver, etc. The saturation power of the SFPs

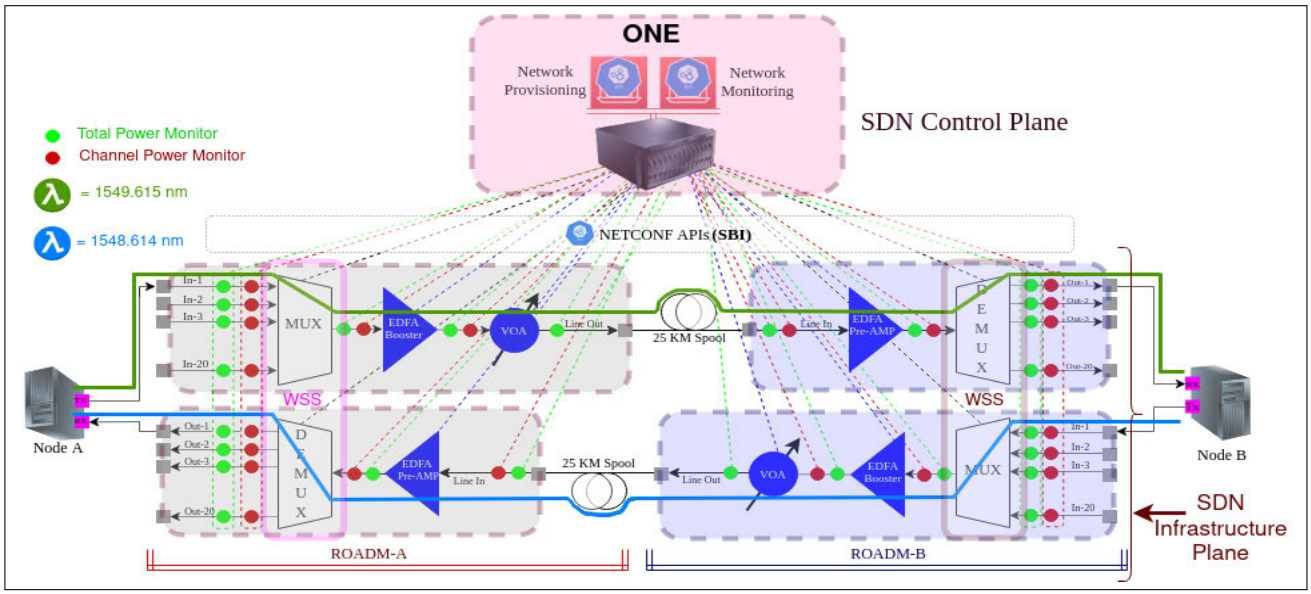

FIGURE 6. Experimental testbed.

TABLE 4. Signal power profile for lightpath from Node-A to Node-B.

ROADM-A								
	MUX Profile				Booster Profile		VOA Profile	
Channel Center-Fr	Input Port	Channel Input Power (dBm)	Channel Output Power (dBm)	MUX Loss (dB)	Gain (dB)	Booster Output Power (dBm)	Attenuation (dB)	Output Signal Power (dBm)
193475 GHz	4101	-0.3	-4.3	-4	3.4	-1.08	0	-1.09
ROADM-B								
	Pre-AMP Profile			DEMUX Profile			VOA Profile	
Channel Center-Fr	Line Input Power (dBm)	Gain (dB)	Pre-AMP Output Power (dBm)	DEMUX Loss (dB)	Output Port	Channel Input Power (dBm)	Channel Attenuation (dB)	Channel Output Power (dBm)
193475 GHz	-6.11	9.8	3.09	-3.4	5201	3.01	0	-0.39

is 0.5dBm and the receive sensitivity is −14.1dBm as shown in Table 3.

The signal power at various components in the ROADMs is given in Table 4. The power received at ROADM-A MUX port 1 is −0.3 dBm and the power of the signal at the output of the MUX is −4.3 dBm showing a loss of about −4dB at the MUX. The gain of the booster is set to the lowest possible value i.e. 3.4 dB which amplifies the signal to −1.08 dBm. We configured an attenuation level of 0dB as we did not want to reduce the signal power. The power of the output signal coming out of the ROADM-A line port is −1.09dBm. We configured ROADM-B such that the same 100 GHz channel can be dropped at DEMUX port 1. The signal received at ROADM-B Line input is about −6.11dBm showing about 5dB loss in the signal power due to the 25 Km fiber spool between ROADM-A and ROADM-B. This signal gets amplified by the Pre-AMP to 3.09dBm. We have configured Pre-AMP to the Lowest possible gain i.e. 9.8dB. The output channel power observed at the DEMUX output port 1 is −0.39dB. With this configuration, the lightpath is setup from Node-A to Node-B. The detailed process described in this section can be repeated to establish a lightpath from Node-B to Node-A as well.

VI. RESULTS AND DEMONSTRATION

A. ANALYSIS OF SIGNALS TRANSMITTED BY SFPS

The lumentum ROADM-20 allows a user to configure a total of 130 channels system-wide starting from 191325.00 GHz to 196125.00 GHz with a granularity of 6.25 GHz. We configured all possible channels on port 1 at the MUX side of the ROADM, effectively converting this port to a fine-tuned OSA. We chose three channel sizes 100 GHz, 50 GHz, and 6.25 GHz for observing the power of the optical signal transmitted by the SFPs. There is no guard band placed between channels because we want to observe the power of each and every received frequency in the C-Band. The 100 GHz bandwidth channel gives an abstract level view of the transmitted signal power and the 6.25 GHz

TABLE 5. Per channel received signal power.

	NODE-B-SFP				NODE-A-SFP			
Channel Bandwidth	**Start-Fr (GHz)**	**End-Fr (GHz)**	**Signal Power(dBm)**	**Power %**	**Start-Fr (GHz)**	**End-Fr (GHz)**	**Signal Power(dBm)**	**Power %**
100 GHz	193525.00	193625.00	-1.8	100	193425.00	193525.00	-0.3	100
50 GHz	193525.00	193575.00	-23.5	0.67	193425.00	193475.00	-1.1	96.7
	193575.00	193625.00	-1.8	99.33	193475.00	193525.00	-15.8	3.3
6.25 GHz	193575.00	193581.25	-15.1 (B1)	6.91	193443.75	193450.00	-17.7 (A1)	2.98
	193581.25	193587.50	-10.5 (B2)	21.36	193450.00	193456.25	-11.9 (A2)	11.31
	193587.50	193593.75	-9.1 (B3)	29.49	193456.25	193462.50	-8.2 (A3)	26.52
	193593.75	193600.00	-9.7 (B4)	25.68	193462.50	193468.75	-7.4 (A4)	31.88
	193600.00	193606.25	-12.6 (B5)	13.17	193468.75	193475.00	-9.5 (A5)	19.66
	193606.25	193612.50	-18.5 (B6)	3.39	193475.00	193481.25	-13.6 (A6)	7.65

bandwidth channel gives a very fine-grained view. We can now observe any optical signal in the C-Band by just connecting the transmitter to port 1 of the MUX. The input ports of MUX are labeled as 41xx where xx is port number i.e. 4101 corresponds to port1 and 4120 corresponds to port 20. There is only one output port of MUX and is labeled as 4201.

The ROADM input port 1 is connected with both SFPs one by one. We noted the start and end frequencies of these channels as shown in Table 5 and configured 12 channels on port 1 again with a channel bandwidth of 50 GHz such that they cover the transmission range of both SFPs. Two channels are reporting received signal power above receiving sensitivity for both SFPs. We noted the start and end frequencies of these channels and configured 64 channels on port 1 again with a channel bandwidth of 6.25 GHz such that they cover the transmission range of both SFPs. There are now six channels reporting received signal power above receiving sensitivity for both SFPs. We have summarized the details of the received signal power and corresponding channels in Table 5. It is evident from the table that the power is not evenly distributed among the received signals. ONE configuration panel as described in section IV-C is used to configure an optimal lightpath from Node-A to Node-B and vice versa using the same process as discussed in section V-A. The lightpath was established with different channel sizes in multiple of 6.25 GHz and studied their impact on the quality of transmission (QoT). We have considered five scenarios as described below:

- 6 Channels: *All six received signal channels are forwarded for both SFPs i.e 100% of the received signal power is forwarded*
- 4 Channels: *Only four channels B2-B5 for Node-B-SFP and A2-A5 for Node-A-SFP are forwarded. The selected four channels encompass around 90% of the total received signal power*
- 3 Channels: *Only three channels B2-B4 for Node-B-SFP and A3-A5 for Node-A-SFP are forwarded. The selected three channels encompass around 80% of the total received signal power*
- 2 Channels: *Only two channels B3-B4 for Node-B-SFP and A3-A4 for Node-A-SFP are forwarded. The selected two channels encompass around 55% of the total received signal power*
- 1 Channel: *Only one channel B3 for Node-B-SFP and A4 for Node-A-SFP are forwarded. The selected channel encompasses around 30% of the total received signal power*

Modulation techniques, Forward Error Correction (FEC), Bit Error Rate (BER), and Optical Signal to Noise Ratio (OSNR) are usually considered Quality of Transmission (QoT) measures. Unfortunately, the SFPs used in the testbed do not report any of these parameters. It was quite challenging to measure the effect of channel selection on the QoT. Therefore, we opted to measure TCP packet loss, TCP windows size, TCP throughput, and Round Trip Time (RTT) as an indicator of channel conditions which can be considered as a novelty in itself. We observed the effect of varying channel conditions on the TCP performance to access the QoT impacted by variations in the channel bandwidth of the optical signals. In the following section, we explained the selection process for the number of TCP sessions that can saturate the link.

B. TCP SESSIONS SELECTION

The testbed used SFPs which support a data rate of 10 Gbps, the specifications of SFPs are given in Table 3. We have used an Iperf TCP session with Node-A acting as the server and Node-B acting as the client. It is observed that a single TCP session is unable to saturate the underlying link. Table 6 shows the average aggregate TCP throughput in Gbps for different numbers of channels and TCP sessions. TCP throughput details are not considered at the moment, for now, we focused on the number of TCP sessions that can saturate the underlying link. It is evident from Table 6 that the aggregate TCP throughput is increasing with the number of TCP sessions irrespective of the number of channels configured. The TCP throughput does not increase further after 4 parallel TCP sessions. Therefore, we choose 4 parallel TCP sessions to measure TCP packet loss, TCP throughput, and Round Trip Time (RTT) for different combinations of channels as described in the previous section. We ran Iperf sessions for 30 seconds and used Wireshark to capture the traffic, for analysis and measurement of these parameters.

TABLE 6. TCP Avg. throughput (Gbps) vs TCP sessions vs channels.

	1 Ch	2 Ch	3 Ch	4 Ch	6 Ch
1 Session	3.34	3.40	3.36	3.22	3.27
2 Sessions	5.50	6.44	6.34	6.34	6.36
3 Sessions	8.46	8.45	8.49	8.49	8.42
4 Sessions	9.43	9.42	9.41	9.41	9.40
5 Sessions	9.45	9.44	9.43	9.44	9.43

C. TCP PACKET LOSS

Wireshark provides *tcp.analysis.flags* filter to report information of various TCP flags. TCP uses these flags to manage connections and data transmission [44]. We used these flags as an indicator of the issues in TCP traffic. Following are the TCP packet loss indicators that these flags typically represent:

- TCP Dup ACK: This flag indicates that the packet is a duplicate acknowledgment (ACK). Duplicate ACKs can be triggered when a segment is received out of order, and the receiver sends an ACK for a segment it has already acknowledged. This is often related to packet loss.
- TCP Out-Of-Order: This flag indicates that the packet arrived out of order. TCP is designed to deliver data in the correct order, so an out-of-order packet suggests network congestion or packet loss.
- TCP Retransmission: This flag indicates that the packet is a retransmission of a previously sent packet. Retransmissions occur when a packet is lost or not acknowledged by the receiver.
- TCP Fast Retransmission: This flag suggests that the packet is part of a fast retransmission event, which occurs when a sender retransmits data before the standard retransmission timeout.

TCP packet loss is plotted against the number of channels as shown in Figure 7. There are two series in the bar graph the blue colored series *"4 TCP Sessions"* is plotted against the left y-axis and the orange colored series *"1 TCP Session"* is plotted against the right y-axis. Different y-axis are used to visualize the huge difference in the values of the two series.

An obvious increase in the packet loss with the decreasing number of channels for 4 TCP sessions has been observed. There is an increase of about 5% in the packet loss between 6 channels and a single channel. There are three reasons for packet loss (a) The channel conditions have been worsened because of the reduced channel bandwidth i.e from 37.5 GHz (6 channels) to just 6.25 GHz (single channel) (b) the signal power has also been reduced from 100% (6 channels) to just about 30% (single channel) although we can overcome this issue by increasing the booster gain (c) End Hosts are unable to deal with the high data rate (9.4 Gbps) because of the 4 parallel TCP sessions. The fluctuating TCP window as discussed in the next section is evident that TCP is facing congestion.

Under normal traffic load i.e. single TCP session (3.3 Gbps), we see that the packet loss is almost 0 just 0.003% as shown in Figure 7. This shows that the reduction in the number of channels does not impact the packet loss. In fact, the packet loss has further dropped to 0.001% for a single channel. The reason for the drop in the packet loss is improved OSNR. Although we could not measure the OSNR, we can infer from the results that OSNR has improved because the channel bandwidth has reduced to just 6.25 GHz (single channel) where the signal strength is very good as given in Table 5 and there will be a very small effect of AWG Noise on the signal.

Based on the preceding discussion, it is concluded that a reduction in the number of channels appears to have no effect on packet loss. The observed increase in packet loss, as depicted in Figure 7, is primarily attributed to the end hosts' inability to sustain traffic at the line rate (10 Gbps).

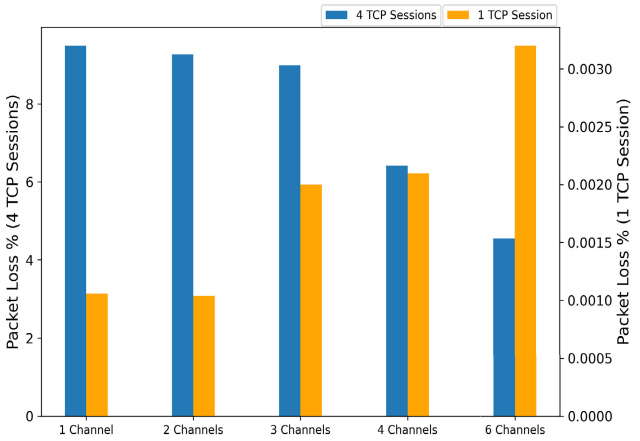

FIGURE 7. Packet loss percentage vs no of channels.

D. TCP WINDOW SIZE

The TCP window size represents the amount of data that a receiver is willing to accept and buffer at any given time without requiring the sender to wait for acknowledgments. TCP Window size is plotted against the time for single and 4 TCP sessions. A single TCP session represents a low load scenario while 4 parallel TCP sessions represent a heavy load scenario. The window size shows very stable behavior for 1, 2, and 3 channels and slight variations for 4 and 6 channels as shown in Figure 8, this behavior is consistent with the packet loss shown in Figure 7 for a single TCP session. A stable window size typically indicates that the network is operating under relatively consistent conditions the sender and receiver are efficiently managing the flow of data, minimizing the potential for congestion or delays. With a stable window size, users can often predict the network's throughput and round-trip times more accurately, which is crucial for real-time applications and interactive communication.

The window size fluctuates under heavy load i.e. for 4 parallel TCP sessions as shown in Figure 9. TCP is designed to adapt to changing network conditions and optimize performance. One of the common reasons for fluctuations is network congestion. When the network experiences

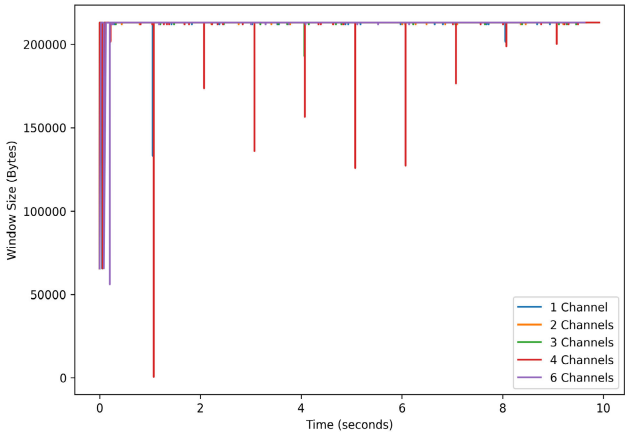

FIGURE 8. TCP window size for single TCP session and different channels.

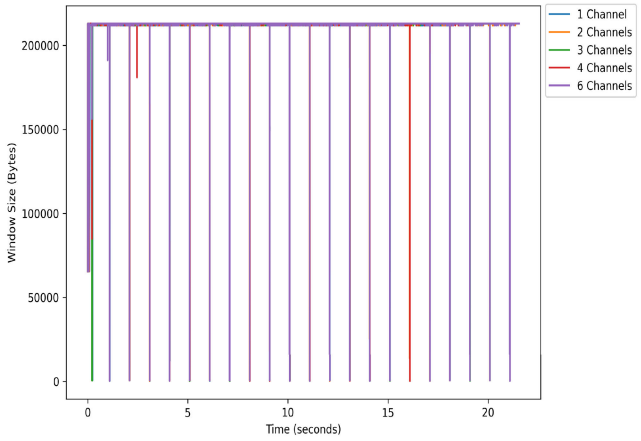

FIGURE 9. TCP window size for single 4 TCP sessions and different channels.

congestion, TCP's congestion control mechanisms kick in, leading to adjustments in the window size. A reduced window size may indicate congestion mitigation efforts. Changes in network latency can also influence the window size. Higher latency may result in a larger window size to accommodate more unacknowledged packets in flight. Lower latency might lead to a smaller window size. In the experimental setup, the network latency is not susceptible to change as there are no routing and switching devices placed along the path between end nodes. Variations in the receiver's buffer capacity can also cause fluctuations. If the receiver's buffer size changes due to factors like available memory or application behavior, it can result in window size adjustments. It is observed that memory did not overload when TCP sessions were active, so the receiver's buffer capacity also doesn't seem to be the reason for the fluctuations shown in Figure 9. The packet loss in the network can trigger fluctuations in the window size. Packet loss is often interpreted by TCP as a sign of congestion, leading to reductions in the window size. In the previous section we have observed packet loss under heavy traffic load as shown in Figure 7. It is evident from the above discussion and results shown in Figure 7 & Figure 9, that these fluctuations are due to the packet loss faced by TCP. Even, if fluctuation is considered due to the receiver's buffer capacity limitation, the number of channels doesn't seem to be the reason.

E. TCP THROUGHPUT

Throughput represents the amount of data that can be transmitted over the network in a given amount of time. It accounts for both the data payload and any overhead (e.g., headers and control information) introduced by the communication protocol. Iperf3 calculates throughput by measuring the number of bits or bytes transmitted during a specified time interval [47]. It reports this rate as the average or instantaneous throughput over the test duration. We ran different (1,2,3, and 4) TCP sessions for 30 seconds as shown in Figure 10 and plotted aggregate throughput in

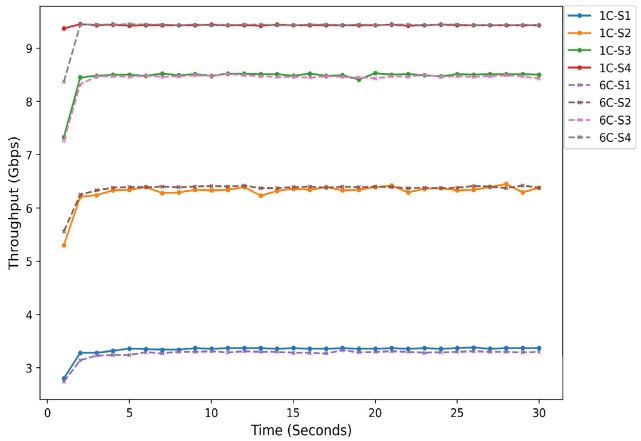

FIGURE 10. Throughput for single channel & 6 channels for different TCP sessions.

Gbps for only two scenarios i.e single channel (6.25 GHz) and 6 channels (37.5 GHz). The aggregate throughput means the sum of the throughput reported by each TCP session. We used the word throughput instead of aggregate throughput for the sake of brevity. The throughput is not affected by the number of channels, despite variations in the number of channels (1C to 6C), the throughput doesn't exhibit any increase or decrease. This suggests that the number of channels may not significantly impact network performance in this context. The aggregate throughput increases as it should be with the number of TCP sessions, it is also evident from Table 6 that a single TCP session can achieve a throughput of about 3.3 Gbps irrespective of the number of channels. The throughput almost doubled for 2 TCP sessions, it reached about 8.5 Gbps for 3 TCP sessions and it reached about 9.47 Gbps for 4 TCP sessions. We can also observe the almost overlapping curves for single and 6-channel scenarios at each step. This confirms that the transmission quality is not impacted by the number of channels. Thus, we can achieve the same throughput for a channel size as low as 6.25 GHz showing 93.75% improvement in the spectrum utilization

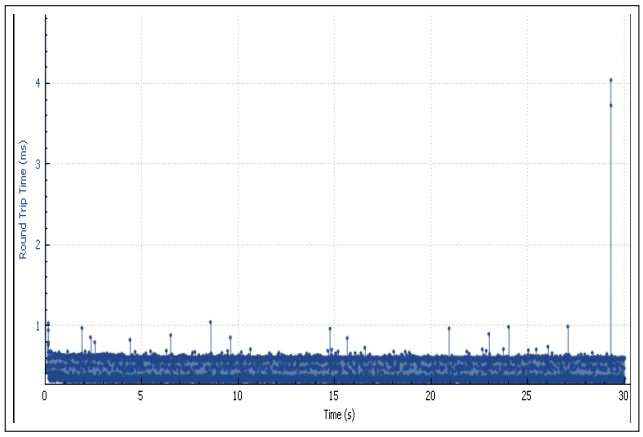

FIGURE 11. RTT for single channel & single TCP session.

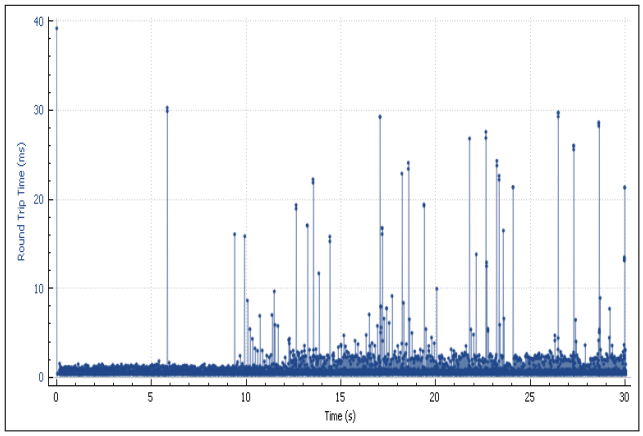

FIGURE 12. RTT for single channel & 4 TCP sessions.

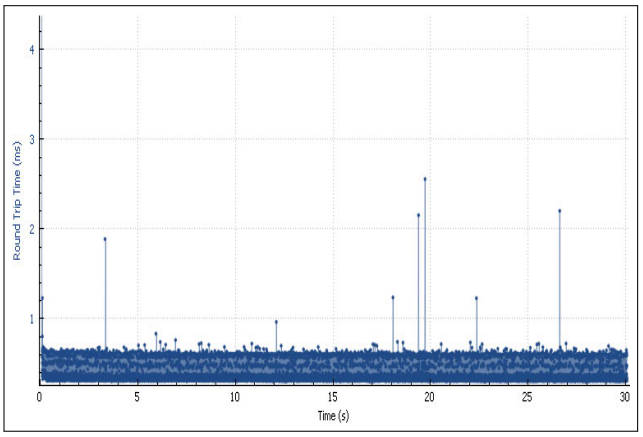

FIGURE 13. RTT for 6 channels & single TCP session.

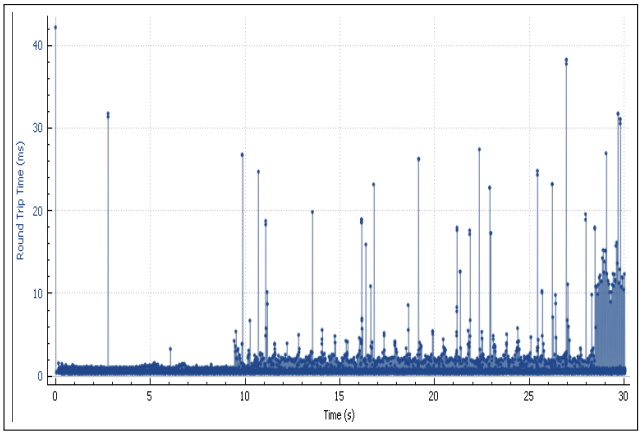

FIGURE 14. RTT for 6 channels & 4 TCP session.

thus, improving overall spectral efficiency by using a flex grid.

F. TCP ROUND TRIP TIME (RTT)

RTT represents the time it takes for a packet to travel from the source to the destination and back, measuring the network latency. We make use of the Wireshark RTT graphs for four special cases to understand the effect of the number of channels on the quality of transmission. The RTT seems very stable and remains around 0.5 ms on average for a single TCP flow irrespective of the number of channels as evident from Figure 11 and Figure 13. On the other hand, we see fluctuations in the RTT for 4 TCP sessions irrespective of the number of channels as shown in Figure 12 and Figure 14. The RTT value goes up to 30 ms for both single channel (6.25 GHz) and 6 channels (37.5 GHz). The RTT plots are also in agreement with the previous results, when only a single TCP session is active we observe stable RTT as there are very small % of packet loss as shown in Figure 7, under heavy load i.e when 4 parallel TCP sessions are active we observe fluctuating RTT due high packet loss and fluctuating TCP window. Thus, the RTT values are not affected by the number of channels but number of active TCP sessions.

G. SPECTRAL EFFICIENCY GAIN

Spectral efficiency (η) is a measure of how efficiently a communication system utilizes the available bandwidth (in Hertz) to transmit data at a certain data rate (in bits per second) expressed in Equation 1. It's typically expressed as bits per second per Hertz (bps/Hz) [48]. We have also defined a new parameter called spectral efficiency gain ($\Delta\eta$) expressed in Equation 2. Spectral Efficiency Gain quantifies the improvement in spectral efficiency achieved by transitioning from a fixed-grid to a flex-grid configuration. It is calculated as a percentage and represents the relative increase in spectral efficiency when using a flexible grid system. A higher ($\Delta\eta$) value indicates a greater efficiency in utilizing the available spectrum, resulting in enhanced network performance and resource utilization.

$$\eta = \frac{R}{B} \tag{1}$$

where:

η : Spectral Efficiency

R : Data Rate (in bps)

B : Bandwidth (in Hz)

$$\Delta\eta = \left(1 - \frac{\eta_{\text{fixed-grid}}}{\eta_{\text{flex-grid}}}\right) \times 100 \tag{2}$$

where:

$\Delta\eta$: Spectral Efficiency Gain

$\eta_{\text{fixed-grid}}$: Spectral Efficiency in a fixed-grid scenario

$\eta_{\text{flex-grid}}$: Spectral Efficiency in a flex-grid scenario

Bar graph for $\Delta\eta$ is plotted against the number of channels. The lumentum ROADM used for this testbed has a granularity of 6.25 GHz. The horizontal axis in Figure 15 represents the number of channels in multiple of 6.25 GHz and the vertical axis represents $\Delta\eta$. We have shown experimentally that we can achieve the same data rate i.e 9.47 Gbps for even a single channel (6.25 GHz) which translates into a spectral efficiency gain of 93.4 % when compared to a 100 GHz fixed grid and translates into a spectral efficiency gain of about 86.8% when compared to 50 GHz fixed grid.

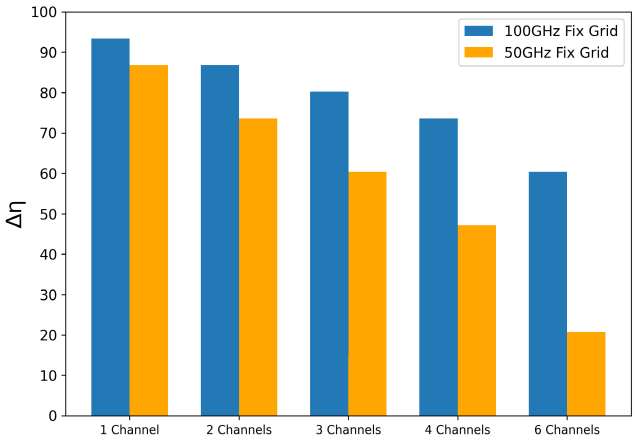

FIGURE 15. Spectral efficiency gain vs no of channels.

VII. CONCLUSION

In conclusion, we presented an Open Networking Engine (ONE) that provides flexibility, efficiency, orchestration, and monitoring capabilities for Software Defined Optical Networks SDON. ONE is based on the device's YANG model and NETCONF protocol, it empowers network administrators to dynamically, configure, provision, and monitor the white-box/gray-box ROADMs. This work showcased the promise of flex-grid technology for enhancing spectral efficiency with the help of ONE and reported a notable spectral efficiency gain of 93.4% by performing a comprehensive signal analysis. We also proposed spectral efficiency gain to highlight the impact of channel bandwidth on TCP packet loss, RTT, window size, and throughput. We will add the telemetry component into the ONE and integrate it with the ONOS SDN controller as the next step.

ACKNOWLEDGMENT

The authors sincere gratitude goes to Asi@Connect for their collaborative efforts and to Technology and Innovation Exchange Network (TEIN) for their support.

REFERENCES

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014, doi: 10.1145/2602204.2602219.

[2] R. K. Jha and B. N. M. Llah, "Software defined optical networks (SDON): Proposed architecture and comparative analysis," *J. Eur. Opt. Soc.-Rapid Publications*, vol. 15, no. 1, pp. 1–15, Jun. 28, 2019, doi: 10.1186/s41476-019-0105-4.

[3] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2015, doi: 10.1109/COMST.2014.2330903.

[4] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and SDN," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1956–1971, 3rd Quart., 2021, doi: 10.1109/COMST.2021.3060582.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008, doi: 10.1145/1355734.1355746.

[6] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based software defined networks: Security challenges and countermeasures," *J. Netw. Comput. Appl.*, vol. 68, pp. 126–139, Jun. 2016, doi: 10.1016/j.jnca.2016.04.011.

[7] A. S. Thyagaturu, A. Mercian, M. P. Mcgarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2738–2786, 4th Quart., 2016, doi: 10.1109/COMST.2016.2586999.

[8] S. P. Moledo, A. Rawat, and A. Gurtov, "Vendor-independent software-defined networking," in *Proc. IEEE 2nd Int. Conf. Signal, Control Commun. (SCC)*, Tunis, Tunisia, Dec. 2021, pp. 168–174, doi: 10.1109/SCC53769.2021.9768368.

[9] K. Nsafoa-Yeboah, E. T. Tchao, B. Yeboah-Akowuah, B. Kommey, A. S. Agbemenu, E. Keelson, and M. M. Khan, "Software-defined networks for optical networks using flexible orchestration: Advances, challenges, and opportunities," *J. Comput. Netw. Commun.*, vol. 2022, pp. 1–40, Aug. 2022, doi: 10.1155/2022/5037702.

[10] M. Garrich, C. San-Nicolás-Martínez, F. J. Moreno-Muro, A. M. Lopez-de-Lerma, O. G. de Dios, V. López, A. Giorgetti, A. Sgambelluri, L. Tancevski, D. Verchere, and P. Pavon-Marino, "Gap analysis on open models for partially-disaggregated SDN optical transport environments," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, Mar. 2019, pp. 1–3.

[11] E. Riccardi, P. Gunning, Ó. G. de Dios, M. Quagliotti, V. López, and A. Lord, "An operator view on the introduction of white boxes into optical networks," *J. Lightw. Technol.*, vol. 36, no. 15, pp. 3062–3072, Aug. 12, 2018, doi: 10.1109/JLT.2018.2815266.

[12] A. Ahmad, A. Bianco, E. Bonetto, M. Garrich, and J. R. F. Oliveira, "Switching node architectures in flexible-grid networks: A performance comparison," in *Proc. Int. Conf. Opt. Netw. Design Model.*, Stockholm, Sweden, May 2014, pp. 49–54.

[13] J. Kundrát, O. Havliš, J. Jedlinský, and J. Vojtech, "Opening up ROADMs: Let us build a disaggregated open optical line system," *J. Lightw. Technol.*, vol. 37, no. 16, pp. 4041–4051, Aug. 27, 2019, doi: 10.1109/JLT.2019.2906620.

[14] Q. P. Van, D. Verchere, H. Tran-Quang, and D. Zeghlache, "Container-based microservices SDN control plane for open disaggregated optical networks," in *Proc. 21st Int. Conf. Transparent Opt. Netw. (ICTON)*, Angers, France, Jul. 2019, pp. 1–4, doi: 10.1109/ICTON.2019.8840430.

[15] R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Control, management, and orchestration of optical networks: Evolution, trends, and challenges," *J. Lightw. Technol.*, vol. 36, no. 7, pp. 1390–1402, Apr. 15, 2018, doi: 10.1109/JLT.2018.2793464.

[16] R. Muñoz, C. Manso, F. Balasis, R. Casellas, R. Vilalta, R. Martínez, C. Wang, N. Yoshikane, T. Tsuritani, and I. Morita, "Dynamic reconfiguration of WDM virtual network topology over SDM networks for spatial channel failure recovery with gRPC telemetry," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2022, pp. 1–3.

[17] *OpenConfig*. Accessed: Jan. 10, 2024. [Online]. Available: https://www.openconfig.net/

[18] M. Birk, O. Renais, G. Lambert, C. Betoule, G. Thouenon, A. Triki, D. Bhardwaj, S. Vachhani, N. Padi, and S. Tse, "The OpenROADM initiative," *J. Opt. Commun. Netw.*, vol. 12, no. 6, pp. C58–C67, Jun. 2020, doi: 10.1364/JOCN.380723.

[19] R. Casellas, R. Vilalta, R. Martínez, and R Muñoz, "SDN Control of disaggregated optical networks with OpenConfig and OpenROADM," in *Proc. Int. IFIP Conf. Opt. Netw. Des. Model.*, May 2019, pp. 452–464, doi: 10.1007/978-3-030-38085-4_39.

[20] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *J. Netw. Comput. Appl.*, vol. 156, Apr. 2020, Art. no. 102563, doi: 10.1016/j.jnca.2020.102563.

[21] C. Xie, L. Wang, L. Dou, M. Xia, S. Chen, H. Zhang, Z. Sun, and J. Cheng, "Open and disaggregated optical transport networks for data center interconnects," *J. Opt. Commun. Netw.*, vol. 12, no. 6, pp. C12–C22, Jun. 2020, doi: 10.1364/jocn.380721.

[22] D. Valencic and V. Mateljan, "Implementation of NETconf protocol," in *Proc. 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO)*, Opatija, Croatia, May 2019, pp. 421–430, doi: 10.23919/MIPRO.2019.8756925.

[23] M. Jethanandani, "YANG, NETconf, RESTconf: What is this all about and how is it used for multi-layer networks," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, Los Angeles, CA, USA, Mar. 2017, pp. 1–65.

[24] W. Zhong and X. Zhao, "Research on DCI streaming telemetry system based on gNMI," in *Proc. 4th Int. Conf. Inf. Sci., Parallel Distrib. Syst. (ISPDS)*, Guangzhou, China, Jul. 2023, pp. 260–263, doi: 10.1109/ISPDS58840.2023.10235592.

[25] I. Iglesias-Castreño, M. G. Alabarce, M. Hernández-Bastida, and P. P. Mariño, "Towards an open-source framework for jointly emulating control and data planes of disaggregated optical networks," in *Proc. 22nd Int. Conf. Transparent Opt. Netw. (ICTON)*, Jul. 2020, pp. 1–4, doi: 10.1109/ICTON51198.2020.9203453.

[26] A. Sgambelluri, A. Giorgetti, F. Paolucci, P. Castoldi, and F. Cugini, "Open source implementation of OpenConfig telemetry-enabled NETconf agent," in *Proc. 21st Int. Conf. Transparent Opt. Netw. (ICTON)*, Angers, France, Jul. 2019, pp. 1–4, doi: 10.1109/ICTON.2019.8840320.

[27] L. Velasco, P. González, and M. Ruiz, "An intelligent optical telemetry architecture," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2023, pp. 1–3, doi: 10.1364/OFC.2023.M3G.1.

[28] B. Lantz, J. Yu, A. Bhardwaj, A. A. Díaz-Montiel, A. Quraishy, S. Santaniello, T. Chen, R. Fujieda, A. Mukhopadhyay, G. Zussman, M. Ruffini, and D. Kilper, "SDN-controlled dynamic front-haul provisioning, emulated on hardware and virtual COSMOS optical x-haul testbeds," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Francisco, CA, USA, Jun. 2021, pp. 1–3.

[29] Y. Liu, A. Dowling, C. Page, and W. Smith, "Building a mininet-based fiber optic simulator," in *Proc. 30th Wireless Opt. Commun. Conf. (WOCC)*, Taipei, Taiwan, Oct. 2021, pp. 279–283, doi: 10.1109/WOCC53213.2021.9603202.

[30] S. Qureshi and R. M. Braun, "Dynamic lightpath allocation in WDM networks using an SDN controller," *IEEE Access*, vol. 9, pp. 148546–148557, 2021, doi: 10.1109/ACCESS.2021.3124522.

[31] S. Qureshi and R. Braun, "Mininet topology: Mirror of the optical switch fabric," in *Proc. 29th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Auckland, New Zealand, Nov. 2019, pp. 1–6, doi: 10.1109/ITNAC46935.2019.9078014.

[32] A. Giorgetti, A. Sgambelluri, R. Casellas, R. Morro, A. Campanella, and P. Castoldi, "Control of open and disaggregated transport networks using the open network operating system (ONOS)," *J. Opt. Commun. Netw.*, vol. 12, no. 2, pp. A171–A181, Feb. 2020, doi: 10.1364/JOCN.12.00A171.

[33] *Open Network Operating System (ONOS)*. Accessed: Jan. 10, 2024. [Online]. Available: https://onosproject.org/

[34] R. Casellas, R. Martínez, R. Vilalta, and R. Muñoz, "Abstraction and control of multi-domain disaggregated optical networks with OpenROADM device models," *J. Lightw. Technol.*, vol. 38, no. 9, pp. 2606–2615, May 24, 2020, doi: 10.1109/JLT.2020.2969248.

[35] C. Wan, C. Yuan, Y. Li, Q. Zhang, and K. Rundberget, "Demonstration of a software solution to support OpenConfig and OpenROADM," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2019, pp. 1–3.

[36] R. P. Pinto, C. H. Cesila, K. S. Mayer, A. F. E. Portilla, D. S. Arantes, D. A. A. Mello, and C. E. Rothenberg, "Demonstration of packet-optical intent-based survivability using mininet-optical," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2023, pp. 1–3, doi: 10.1364/OFC.2023.M3Z.10.

[37] N. Ellsworth, T. Zhang, S. Troia, G. Maier, and A. Fumagalli, "Enhancing cross layer monitoring on open optical transport networks," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2023, pp. 1–3, doi: 10.1364/OFC.2023.M3Z.14.

[38] G. Borraccini, R. Ambrosone, A. Giorgetti, S. Straullu, F. Aquilino, E. Virgillito, A. D'Amico, R. D'Ingillo, N. Sambo, F. Cugini, and V. Curri, "Disaggregated optical network orchestration based on the physical layer digital twin," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Diego, CA, USA, Mar. 2023, pp. 1–3, doi: 10.1364/OFC.2023.Tu3D.4.

[39] A. Ferrari, M. Filer, K. Balasubramanian, Y. Yin, E. L. Rouzic, J. Kundrát, G. Grammel, G. Galimberti, and V. Cur, "GNPy: An open source application for physical layer aware open optical networks," *J. Opt. Commun. Netw.*, vol. 12, no. 6, p. C31, Jun. 2020, doi: doi:.

[40] M. S. Raza, A. D'Amico, F. Usmani, S. M. Alavi, M. A. Taimoor, V. Curri, and A. Ahmad, "LYNX: A GNPy-based Web application for multi-vendor optical network planning," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, Mar. 2022, pp. 1–3.

[41] 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," *J. Opt. Commun. Netw.*, vol. 15, no. 3, pp. 163–173, Mar. 2023, doi: 10.1364/JOCN.477732.

[42] N. Ellsworth, B. Mirkhanzadeh, T. Zhang, S. Vachhani, B. Bathula, G. Thouenon, C. Betoule, O. Renais, M. Razo, and A. Fumagalli, "A non-proprietary network operations platform for OpenROADM environment," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, San Francisco, CA, USA, Jun. 2021, pp. 1–3.

[43] J. Kundrát, J. Vojtěch, P. Škoda, R. Vohnout, J. Radil, and O. Havliš, "YANG/NETCONF ROADM: Evolving open DWDM toward SDN applications," *J. Lightw. Technol.*, vol. 36, no. 15, pp. 3105–3114, Aug. 2, 2018, doi: 10.1109/JLT.2018.2822268.

[44] J. D'souza, M. J. Kaur, H. A. Mohamad, and P. Maheshwari, "Transmission control protocol (TCP) delay analysis in real time network," in *Proc. Adv. Sci. Eng. Technol. Int. Conferences (ASET)*, Dubai, United Arab Emirates, Feb. 2020, pp. 1–6, doi: 10.1109/ASET48392.2020.9118173.

[45] *One GitHub Repository*. Accessed: Jan. 10, 2024. [Online]. Available: https://github.com/Mati86/ONE-DBT

[46] *Optical Networks and Technologies (ONT) Lab*. Accessed: Jan. 10, 2024. [Online]. Available: https://ont.seecs.edu.pk

[47] H. M. U. Rahman, G. Bahoo, L. Zafar, I. AL-Oqily, H. A. Khattak, and A. Ahmad, "Empirical performance evaluation of open source SDN controllers in different network topologies," in *Proc. NOMS-IEEE/IFIP Netw. Operations Manage. Symp.*, Miami, FL, USA, May 2023, pp. 1–6, doi: 10.1109/NOMS56928.2023.10154394.

[48] A. Ahmad, A. Bianco, H. Chouman, V. Curri, G. Marchetto, and S. Tahir, "Exploring the effects of physical layer parameters in WDM based fixed and flexible-grid networks," in *Proc. Int. Conf. Opt. Netw. Design Model. (ONDM)*, Pisa, Pisa, Italy, May 2015, pp. 128–133, doi: 10.1109/ONDM.2015.7127286.

HAFIZ MATI UR RAHMAN received the Bachelor of Science degree in electronics and the Master of Science degree in computer engineering. He is currently pursuing the Ph.D. degree with SEECS, NUST. He is an Assistant Manager with xFlow Research. He possesses excellent team management and interpersonal abilities. He has extensive knowledge of computer networks, network programmability, and DevOps. He is primarily in charge of SONiC-related research, development, and support services. In addition, he is a LiFT Scholar-2022 with Cloud Captain category. His research interests include open and disaggregated optical networks, and software-defined networking (SDN).

RIDA HANIF received the bachelor's degree in software engineering from Fatima Jinnah Women University, in 2018. She is currently pursuing the M.S. degree in computer science with the Military College of Signals, NUST. She was a Researcher and a Developer with the Ad Hoc Wireless and 5G Networks Laboratory, AIR University, Islamabad. She is a Senior Development Engineer with xFlow Research. She is heading the SONiC Development Team, xFlow Research. She is a LiFT Scholar with Women in Open Source category. She is an Active Member with Women Who Code. Her research interests include ad-hoc wireless networks, software-defined networking, open networking, network orchestration, monitoring, and automation.

SALMAN GHAFOOR received the B.Sc. degree in electrical engineering from UET Peshawar, Peshawar, Pakistan, in 2006, the M.Sc. degree in electronic communications and computer engineering from the University of Nottingham, Nottingham, U.K., and the Ph.D. degree from the University of Southampton, Southampton, U.K. He is currently a Professor with the National University of Sciences and Technology (NUST), Islamabad, Pakistan. His areas of research interests include optical communication systems, photonic signal processing, ultra-wideband over fiber, radio over fiber systems, and optical sensors.

ARSALAN AHMAD (Senior Member, IEEE) received the M.Sc. degree in communication engineering and the Ph.D. degree in electronics and communication engineering from Politecnico di Torino, Italy, in 2010 and 2014, respectively. Later, he was a Postdoctoral Researcher with Politecnico di Torino and a Research Fellow with CONNECT Research Centre, Trinity College Dublin, Ireland. He joined the National University of Sciences and Technology (NUST), Pakistan, in 2015, where he is currently an Associate Professor and the Director of the Optical Networks and Technologies (ONT) Lab. Since January 2023, he has been a Visiting Professor with the Department of Electrical and Computer Engineering, Oklahoma State University, USA. His main research interests include optical networks, network planning, virtual PONs, network slicing, and software-defined networking.