This paper presents an experimental evaluation of network protocols for control and management of optical networks and optical network equipment as seen in current trends. This paper presents the YANG data modeling language and its associated RESTCONF/NETCONF protocols. Later, it details multiple data models used in optical networks, such as IETF TEAS, ONF Transport API, OpenROADM, and OpenConfig. It also presents multiple protocols for telemetry (i.e., YANG PUSH, gRPC, and gNMI). Later, a zero-touch SDN controller architecture for multiple standard-defining organizations (SDOs) is presented, to support the usage of multiple protocols in a zero-touch optical network. Finally, the presented protocol implementations are experimentally evaluated and compared in terms of latency and overhead. The paper explores the use of these protocols (e.g., in research, demonstrations, and open-source optical networking projects) and provides results and recommendations for their integration in equipment and networks.

© 2021 Optical Society of America

https://doi.org/10.1364/JOCN.424631

1. INTRODUCTION

Software-defined networking (SDN) introduced the logical decoupling of control and data planes by providing a logically centralized architecture to dynamically control (multiple) networks. Since its definition, SDN incorporated several key concepts such as standardized management interfaces and open interfaces. OpenFlow was the first SDN protocol, which was early extended for optical networks [1].

Optical network control and management has evolved, and today it allows operators to understand the proper monitored data, and it is able to implement automated troubleshooting, leading to what is known as zero-touch optical networking (ZTON). This has been reached, thanks to many innovations. Multiple standard-defining organizations (SDOs) have contributed with data modeling languages and the corresponding data models to describe device capabilities, attributes, operations, and notifications to be performed or received from a device or system. The Internet Engineering Task Force (IETF) has proposed Yet Another Next Generation (YANG) data model language [2]. This introduction has eased evolution of SDN thanks to the associated tools and protocols enabling complex models with complex semantics that are flexible and support extensions and augmentations. Later, the introduction of protocol buffers [3] signified a new step in data model definition. In this paper, we will present and identified how both data modeling languages have succeeded in the definition of open optical network standards, although it is hard to reach consensus.

The proposed data modeling languages have an associated transport protocol, which provides primitives to view and manipulate the data, providing suitable encoding as defined by the data model. Ideally, data models should be protocol independent. Each proposed transport protocol should provide an architecture for remote configuration and control based on the client/server. It should support multiple clients, provide access lists, include transactional semantics, and deploy roll-back functionalities in case of an error. The Network Configuration Protocol (NETCONF) [4] was the first protocol developed specifically for YANG data models. Later, the RESTCONF protocol [5] was presented as a feasible solution to provide the benefits of NETCONF using representational state transfer (REST) with Hypertext Transfer Protocol (HTTP). In recent years, a new breed of protocols for network configuration that are able to deal with protocol buffers have been proposed, such as gRPC [6] and gNMI [7].

On the other hand, SDN-enabled applications are being developed; e.g., on top of open-source-based SDN controllers responsible for the logically centralized control of the optical network. These optical SDN controllers provide the necessary
network simplification through abstraction and virtualization. These are provided using multiple services that include topology, provisioning, path computation, virtualization, and inter-domain connectivity. Several protocols have been defined for northbound interfaces (NBIs) of an optical SDN controller. The Open Networking Foundation (ONF) proposed the set of functional requirements and information model(s) for the Transport API (T-API) [8]. In parallel, the IETF Traffic Engineering Architecture and Signaling (TEAS) working group has worked in data models for a traffic engineering (TE) topology description [9] and for TE connectivity provisioning [10].

As southbound interfaces (SBIs), optical SDN controllers have started to introduce the following data models with regard to network equipment control and management: OpenROADM [11] and OpenConfig [12]. OpenROADM is a multisource agreement between multiple manufacturers to make reconfigurable optical add/drop multiplexers (ROADMs) compatible across vendors. OpenConfig is an informal working group of network operators adopting SDN principles such as model-driven management. It contains both protocol and data model definitions and focuses on router and line card configuration.

Finally, network operations require the continuous monitoring of network data provided through streaming mechanisms, which is known as telemetry. Telemetry has been using Simple Network Management Protocol (SNMP) for the past decade, both in synchronous and asynchronous monitoring. To introduce more significant values and lower performance overheads, several protocols are being proposed: YANG PUSH [13], gRPC [6], and gNMI [7].

This paper extends [14]. It provides more depth in describing the state of the art of the multiple protocols and data models involved in controlling and monitoring optical networks and elements. It also includes a description of an SDN controller architecture that is able to handle multi-SDO interfaces. This architecture has been generalized from current SDN controller implementations. Later, the concept of ZTON is introduced, including its characteristics. Finally, this paper also includes an experimental evaluation of the proposed protocols using the same information model and scenarios. Although all experimental results might not be comparable, this evaluation provides some insights into the behavior in terms of payload length and related protocol efficiency.

The paper has five sections. Section 2 provides a detailed overview of current protocols for control and monitoring of optical networks and equipment. Section 3 describes current attempts for SDN controllers to implement multiple standards-based interfaces, including the main characteristics of zero-touch optical networking and how the used telemetry can be applied to automate optical networks. Section 4 evaluates the various presented protocols and provides detailed results with regard to protocol bit usage and latency. Finally, Section 5 offers some conclusions.

### 2. Current Control and Monitoring Protocols for Optical Networks and Equipment

In this section, we provide a state-of-the-art of current data models and protocols for (a) YANG modeling language and protocols, (b) control and management of optical networks and equipment, and (c) telemetry mechanisms.

#### A. YANG Modeling Language and Protocols

This section presents the YANG data modeling language and the NETCONF/RESTCONF protocol. Both are used for defining network control interfaces, usually available at the NBI of an SDN controller. The main concepts are summarized in Table 1.

1. **YANG**

YANG is a data modeling language [2]. It is used to define a component configuration, state, and notifications. YANG structures data into data trees within the so-called datastores, by means of encapsulation of containers and lists, and to define constrained data types. It allows the refinement of models by extending and constraining existing models (by inheritance/augmentation), resulting in a hierarchy of models. A YANG model includes a header and imports, and includes statements, type definitions, configurations, and operational data declarations as well as actions (remote procedure call or RPC) and notifications.

As an example, Listing 1 shows a simplified topology data model. It shows the definition of a module (topology), which consists of a container (topology) that includes a list of nodes (list node) and links (list link). Each list is ordered using its identifier (node-id, link-id, respectively). A node includes a node identifier (node-id) and a list of ports (list port). Each port has its own identifier (port-id) and a specific layer (layer-protocol-name). Each link includes a link identifier (link-id) and references (leafref) to the source and target node and port identifiers.

| Table 1. NETCONF/RESTCONF Summary |
|-----------------------------------|
| **Data modeling language** | NETCONF | RESTCONF |
| Transport | YANG | HTTP |
| Encoding | XML | XML/JSON |
| Capability exchange | During session establishment | Retrieval of YANG modules and capability URLs |
| Multiple datastores | YES | YES |
| Datstore locking | YES | NO |
| Security | SSH | TLS |
YANG has had a significant adoption as a data modeling language in open source projects and SDOs. There is a significant ongoing effort to model constructs including optical devices, such as transceivers and ROADMs. This has led to literally hundreds of emerging standards across multiple SDOs.

2. NETCONF

NETCONF is a control and management protocol that supports the configuration of devices based on their known YANG-based data models. It is based on the exchange of XML-encoded RPC messages over a secure (commonly called the Secure Shell or SSH protocol) connection.

NETCONF offers operation primitives to view and manipulate data (e.g., `<get-config>`, `<edit-config>`).

Data is arranged into one or multiple configuration datastores. NETCONF defines the existence of one or more datastores and allows configuration operations on them. Only the running configuration datastore is present in the base model [15].

NETCONF-enabled devices include a NETCONF server, while control and management applications include a NETCONF client. First, they will establish a session over a secure transport. Then, both entities will send a hello message to announce their protocol capabilities, the supported data models, and the server’s session identifier. Finally, when accessing configuration or state data, with NETCONF operations, subtree filter expressions can select subtrees.

One of the first evaluations of NETCONF in optical networks was presented in [16]. The authors propose a YANG model to describe a sliceable transponder to be deployed in an elastic optical network with variable rate, code, modulation formats, and monitoring capabilities. NETCONF is proposed to comply with two use cases: a) transponder configuration and b) notification when a BER threshold is exceeded. We provide an overview of the messaging of NETCONF between client and server, which includes edit-config, subscription, and notification messages.

3. RESTCONF

RESTCONF is a protocol that provides an HTTP-based API to access the data, modeled by YANG [5]. RESTCONF protocol organizes the datastore using uniform resource identifiers (URIs) that reflect data hierarchy. HTTP REST operations such as create, read, update, and delete (CRUD) are applied to the defined URI.

State and configuration data can be retrieved with the HTTP GET action. Configuration data can be modified with the POST (create), PUT (update) and DELETE methods. Data is encoded with either XML or JSON.

Current SDN controllers implement their NBI using HTTP servers. The introduction of RESTCONF to ONOS and OpenDayLight has eased the adoption of novel data models, such as the one presented in the next sections.

B. Control and Management of Optical Networks and Equipment

This section describes the most extended YANG data models for control and management of optical networks and equipment, which have been adopted in disaggregated optical networks: IETF TEAS, ONF T-API, OpenROADM, and OpenConfig. IETF TEAS and ONF T-API use RESTCONF, while OpenROADM and OpenConfig use extensively the NETCONF protocol. Table 2 summarizes the presented standards. Complexity refers to the number of YANG lines to describe the data model. Maturity takes into consideration the number of proofs-of-concept and field trials, where the data model has been involved.

1. IETF TEAS

TEAS is an IETF working group that has proposed the YANG data model for TE topologies [9] and for TE tunnels and
Table 2. Standards Summary

| Standards | IETF TEAS | T-API | OpenROADM | OpenConfig Data Model |
|-----------|-----------|-------|-----------|----------------------|
| Objective | NBI transport SDN controller | NBI transport SDN controller | Disaggregated ROADM | Router and line card configuration |
| Data model | YANG | YANG | YANG | YANG |
| Complexity | ++ | ++ | ++ | ++ |
| Maturity | ++ | ++ | ++ | ++ |
| SDO | IETF | ONF, OIF | MSA | MSA |

interfaces [10]. The proposed data models allow network TE and they can be easily extended with optical parameters, such as in wavelength switched optical networks (WSON) [17] or flexi-grid networks [18]. These extensions are described in the Common Control and Measurement Plane (CCAMP) working group.

Szyrkowiec et al. in [19] analyze different data models, detailing their respective strengths and weaknesses. They show their application areas by mapping them to components of a generic optical network SDN controller.

IETF has also proposed the Abstraction and Control of TE Networks (ACTN) framework as high-level abstractions, which includes multitechnology and multidomain transport network services. It enables legacy heterogeneous transport network control and management technologies. Multidomain service coordination based on abstraction/virtualization is achieved through a hierarchy of controllers. Customer network controllers (CNCs) are deployed on-demand by customers to handle their own virtual and abstracted resources. Multidomain service coordinators (MDSCs) are responsible for offering resources to connected CNCs. MDSCs are responsible for coordinating underlying physical network controllers (PNCs).

Casellas et al. in [20] demonstrate the application of the ACTN architecture to control a multidomain flexi-grid optical network. It adopts and extends the hierarchical active stateful path computation element (PCE) architecture to provide per-link partitioning of the optical spectrum based on variable-sized allocated frequency slots, enabling network sharing and virtualization. MDSCs can be mapped to a parent PCE, and PNCs can be mapped to child PCE. Another demonstration of ACTN applicability is cross stratum orchestration (CSO), which is proposed as a feasible solution for NFV points of presence interconnection [21]. In that paper, the demo architecture and the interfaces/APIs were aligned with IETF ACTN.

In parallel to the IETF proposed data models, the ONF T-API has proven to be a feasible solution for multivendor SDN in transport networks [8]. Its use in transport SDN controllers is spreading due to several benefits: (a) a well-known API; (b) it is extensible and has open source software solutions; (c) it has been tested and deployed in multiple interoperability events; (d) it provides proper abstractions; (e) it covers a multitude of use cases, such as multilayer; and (f) it is supported by a great community of vendors.

Figure 1 introduces the main T-API concepts. All interactions between an SDN controller and a T-API client (e.g., application or orchestrator) occur within a shared context, which is defined by a set of service interface points (SIPs). SIPs allow a T-API client to request connectivity services between them. An SDN controller may expose one or more abstract topologies within a shared context. These topologies may or may not map one-to-one to a provider’s internal topology. They are expressed in terms of nodes and links. On one side, nodes aggregate node edge points (NEPs). On the other side, links connect two or more nodes and they terminate on NEPs. These NEPs may be mapped to one or more SIPs.

A T-API client might request a connectivity service (CS) between two or more SIPs. In response to the requested CS, the SDN controller creates one or more connections. The connection end points (CEPs) encapsulate the related connection information regarding underlying NEPs.

2. ONF Transport API

Mayoral et al. in [22] demonstrated, for the first time, the deployment and use of T-API. Topology and connectivity services were demonstrated as a feasible solution for multivendor SDN in transport networks. Multiple authors have extended T-API to support many novel technologies, such as space division multiplexing (SDM). Vilalta et al. in [23] extended T-API for SDM networks and demonstrated the extensions in an experimental NFV MANO framework for the control of SDM/WDM-enabled fronthaul and packet-based transport networks.

The ONOS SDN controller has demonstrated the recursive usage of T-API in the Open and Disaggregated Transport Network (ODTN) project [24]. The aim of this project is to provide support for Ethernet (connectivity service establishment between client ports) and photonic layers (between transceiver line ports). On the other hand, OpenDaylight includes the TransportPCE project, which also targets the photonic and OTN layers, including the ability to configure
OTN cross-connections and interfaces. It includes a partial implementation of ONF T-API to support the export of an abstracted view of the topology to an SDN controller [25].

3. OpenROADM

OpenROADM is a multisource agreement (MSA) between operators and vendors that defines interoperability specifications for ROADM switches, transponders, and pluggable optics. Specifications consist of both optical interoperability as well as YANG data models.

OpenROADM defines a vendor-neutral model for configuration and management for three network interfaces:

- common single-wave interface between transponders,
- common multiwave interface between ROADMs,
- common YANG models for all components.

Morro et al. [26] show end-to-end carrier Ethernet circuit orchestration in a hierarchical control plane of ONOS SDN controllers. New device drivers were developed for ONOS to directly configure ROADMs using NETCONF and YANG models defined by the OpenROADM project.

In [27], Mirkhanzadeh et al. presented the most complete OpenROADM demonstration. It provided novel functionalities for multilayer use cases using the TransportPCE project of OpenDaylight. TransportPCE communicates with the SDN orchestrator. The SDN orchestrator provides unified control and management of the resources across three domains (i.e., the optical layer, Ethernet layer, and datacenter compute nodes). Several use cases have been demonstrated, such as datacenter backups and massive virtual machine live migrations.

4. OpenConfig

OpenConfig is an informal working group of network operators with the common objective of introducing more programmability and dynamicity to the networks. A key driver for this objective is the adoption of SDN principles that include declarative configuration and model-driven operations.

OpenConfig provides a set of vendor-neutral YANG data models for a variety of network elements, from routers and switches to optical transport. If we focus on optical transport, it provides a configuration and state model for the following:

- Terminal device: terminal optical devices within a DWDM system, including both client- and line-side parameters.
- Wavelength router: configuration and operational state data for an optical transport line system node, or ROADM [e.g., colorless directionless contentionless (CDC) ROADMs, wavelength selective switches (WSSs) and a dynamic gain equalizer].
- Optical amplifier: configuration and operational state data for optical amplifiers, deployed as part of a transport line system.
- Channel monitor: operational state data for an optical channel monitor (OCM) for optical transport line system elements such as wavelength routers (ROADMs) and amplifiers.
- Transport line protection: configuration and operational state data for optical line protection elements, such as an automatic protection switch (APS). The APS provides protection using two dark fiber pairs to ensure the amplifiers can still receive a signal if one of the two fiber pairs is broken.

Paolucci et al. [28] present extensions to the OpenConfig terminal device data model that enable dynamic selection of transmission parameters of 100G/400G transmitters with coherent reception in a filterless metro network.

Dou et al. in [29] provide a demonstration of augmentation of the OpenConfig data model of a wavelength router to demonstrate network disaggregation, including various operations on both degrees and media channels.

The ONOS ODTN project includes OpenConfig models over NETCONF for a transponder configuration. There are several reasons to select these models: OpenConfig is a well-known API that is already supported by many vendors, it provides a proper abstraction model for transponder device capabilities and information, it also defines capabilities at the correct level for programmability but also abstraction from physical details, it includes capability and flexibility to support vendor specific features, and it is extensible and open source.

Although OpenConfig and OpenROADM try to provide similar solutions for modeling optical networks and elements, they are typically used for different purposes. For example, the ONOS ODTN project, as previously detailed, uses OpenConfig for transponder devices, but uses OpenROADM for the OLS [30].

C. Telemetry Mechanisms in Optical Networks

Streaming of monitoring data is known as telemetry and it is used in the context of network control and management, with the objective to monitor and troubleshoot network status. SNMP has been the most used protocol for this purpose [31]. Transaction Language 1 (TL1) is also an existing, widely used management protocol. Several types of data are exposed and consumed in transport networks, such as network state indicators, network statistics, and critical infrastructure information. During the past decade, new protocols (i.e., YANG PUSH, gRPC, and gNMI) have incorporated significant novel features that make them attractive to network administrators:

- the ability to do bulk retrievals;
- support of multitentancy of the monitored device (e.g., through multiple NMSs);
- the introduction of highly efficient protocol buffers (protobuf);
- novel subscription types defined with the introduction of a push connection and forward targeted telemetry data to a targeted recipient when certain criteria are met.

Table 3 provides a summary of gRPC and gNMI as protocols. The YANG PUSH mechanism uses NETCONF, which has previously been detailed.
Table 3. Telemetry Summary

| Protocol | gRPC | gNMI |
|----------|------|------|
| Data modeling language | Protocol buffers | YANG (protocol buffers) |
| Transport | HTTP/2 | gRPC |
| Encoding | Byte | JSON/byte |
| Capability exchange | No | During session establishment |
| Multiple datastores | NO | YES (Config/State/Operational) |
| Datastore locking | NO | NO |
| Security | TLS | TLS |

1. YANG PUSH

YANG PUSH is a mechanism that allows applications to subscribe to a customized stream of updates from any available YANG datastore [13]. It is provided over the NETCONF protocol. It allows a client to monitor changes to a YANG datastore with notifications instead of with NETCONF GET requests. This provides lower resource consumption and ease of system integration.

Multiple datastores can be selected for subscription (e.g., running, candidate, operational). There are two kinds of datastore subscriptions:

- Periodic subscriptions, where the data is sent to the client every specified time interval, even if the data is not modified. Periodic subscriptions are similar to periodic NETCONF GET requests, but they differ in the fact that the client does not send any request.
- On-change subscriptions, where the data is sent to the client only when the data changes. This subscription method is much more efficient than the periodic requests or subscription approaches.

The work presented in [32] demonstrated for the first time the implementation of YANG push notifications on an open terminal based on OpenConfig models. Examples for both periodic and on-change subscription are provided.

2. gRPC

Google Remote Procedure Calls (gRPC) [6] is a protocol based on HTTP/2 as a transport protocol. It uses protocol buffer encodings for transported messages. Because it is based on HTTP/2 transport protocol and uses byte-oriented encoding, it introduces low latency. gRPC has been used in highly scalable and distributed systems.

Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data [33]. They allow modeling of the data structures in a similar manner as in YANG. Its encoding in byte-oriented messages increases the efficiency compared to XML/JSON encodings. An open-source framework is provided to handle protocol buffers in multiple languages, such as Python, Go, C++ and Java.

Listing 2 provides an example of the same topology information model previously presented using YANG. In this example, a topology service is described and it offers a GetTopology RPC that provides a topology structure. Each structure is described in the message keyword. The topology message provides a list of node messages (a list is declared using a repeated keyword) and a list of link messages. A node message includes an identifier and a list of port messages. Port messages include a port identifier and a layer protocol name. Link messages include a link identifier and references to source and target nodes and ports.

In [34], Vilalta et al. provide the first experimental demonstration of a gRPC-based SDN control and telemetry architecture. They apply gRPC to control and monitor spectral/spatial superchannels with SDM/WDM transceivers monitoring the BER to detect soft failures over an 11 km, 6 mode, 19 core fiber.

3. gNMI

The gRPC Network Management Interface (gNMI) [7] is a protocol for configuration manipulation and state retrieval. It is built on top of gRPC and is described using protobuf. It can use binary or JSON encoding for the payload, which allows the use of YANG data models, so all efforts can be integrated to define them in SDOs.

A gNMI target is the device that acts as the owner of the data being manipulated or reported on. Typically, this will be a network device. A gNMI client is the device using the protocol to query/modify data on the target. Typically, this will be a network management system.
The gNMI telemetry makes use of the SubscribeRequest message. It has been demonstrated to assess and retrieve transmission performance and rapidly determine the most suitable operational mode in [35].

In [36], Sadasivarao et al. have demonstrated optical performance monitoring data across the optical terminal and OADM devices. These data are sent to an analysis collection platform. OpenConfig usage with gNMI has been demonstrated by Vilalta et al. in [37]. In this paper, real-time monitoring of optical transponders using gNMI is demonstrated using a transport SDN controller.

3. MULTI-SDO SDN CONTROLLER ARCHITECTURE AND ZERO-TOUCH OPTICAL NETWORKING

This section presents a generalized architecture that is able to handle multiple data models, and also shows how to use the presented architecture to provide zero-touch optical networking.

A comprehensive picture of transport SDN is provided by [38], which includes an analysis of the multiple SDN controllers. Current SDN controllers, such as OpenDaylight or ONOS, allow interaction with other SDN controllers and network equipment using multiple defined data models from multiple SDOs. We refer to this kind of SDN controller as a multi-SDO SDN controller.

In this section, we generalize the common functionalities that help current SDN controllers overcome multiple data models and protocols. The multi-SDO SDN controller has its internal data models. They define the main purposes and activities of the SDN controller. The internal data model can be based on well-known data models. Note that, in Figure 2, ONF T-API has been adopted as the internal data model. Typical SDN controller modules and thus data models, might include topology, provisioning, path computation, virtual networks, inter-domain connectivity and operations, and administration and management (OAM).

For the multi-SDO northbound interface, the NBI translator is needed, as well as the multiple servers that offer multiple protocols such as NETCONF and RESTCONF. These servers are responsible for offering the multiple data models that can be obtained through an NBI translator from the internal data models. Figure 2 shows how ONF T-API and IETF TEAS are offered as an NBI using translation from internal data models.

Regarding southbound API, the mechanism is similar. First, multiple clients are provided for the proposed protocols (e.g., ONF T-API, IETF TEAS, OpenROADM, and OpenConfig). An SBI translator also is needed to translate the required internal data model actions toward underlying networks and equipment.

ZTON is related to two main topics: a) sustaining fast network growth automatically without human intervention and b) minimizing human errors (and interventions) by minimizing associated outages [6]. Thus, ZTON requires the proposed control and telemetry protocols for optical network monitoring to receive proper network notifications and to interact with new configuration for network elements.

ZTON includes the following characteristics:

- The only required operator step is the instantiating of intent. Then, all network operations are automated.
- Network-wide intents are decomposed into declarative and vendor-neutral configuration for individual network elements.
- If unintended behavior is detected, any network changes are automatically halted and rolled back.
- Operation violating network policies shall not be allowed by the infrastructure.

Figure 3 shows the suggested state diagram [6] for providing ZTON. It consists of a closed loop that is fed with the proper monitoring data. This data is obtained through the multiple mechanisms detailed in the previous telemetry section. The analysis of the data leads to error detection, which triggers automated troubleshooting. This will lead to a repair action, which requires automated control of the optical network and the declaration of novel intents for repairing it. Finally, the repair will be verified.

4. EXPERIMENTAL EVALUATION

In this section, we propose a methodological approach to evaluate the previously presented protocols that have been demonstrated to be applied to optical networks: (a) NETCONF, (b) RESTCONF, (c) gRPC, and (d) gNMI. To perform this evaluation, two topologies have been defined to evaluate the different protocols: a) the National Science Foundation Network (NSFNET), which includes 14 nodes and 21 links, and b) the European Optical Network (EON), which includes 19 nodes and 38 links. Both topologies have been defined using a JSON file that can be parsed by the multiple servers that we have created for evaluation.
The previously presented topology data models, Listings 1 and 2, have been used to describe NSFNET and EON scenarios. The defined data models include the same amount of information. Listing 1 has been validated using PYANG [39], while Listing 2 has been validated using a protobuf compiler [33]. All measurements have been obtained using an Ubuntu Linux Server with an i7 10th generation CPU and 4 GB RAM. Each of the experiments has been repeated 10 times and the mean value is provided.

The purpose of these experiments is to evaluate numerically the proposed protocols in terms of bit usage and latency. Bit usage refers to the total amount of bits interchanged between a client and a server to provide a complete request/response. Latency refers to the required round-trip time since the issue of a request action from the client and the necessary server time to answer to that action. We believe that, for simple use cases, protocol overhead and latency might be enough for comparison, but the results are presented individually for each protocol because each protocol provides unique features for specific use cases.

A. NETCONF

To evaluate NETCONF performance, the NETCONF server includes PYANGBIND [40] generated code, to properly describe the selected topology using the defined YANG data model (Listing 1). For the NETCONF protocol, we have based our server and client implementations on open source library NETCONF [15]. NETCONF protocol includes an SSH secure connection between client and server; thus, results might not be comparable to other protocols that include different security layers such as TLS.

Figure 4 shows the results of bit usage in the NETCONF get-config operation for the described topology. This operation requires 140,000 bits for NSFNET topology, while it requires up to 200,640 bits for EON topology. The EON topology requires more than 43% compared to NSFNET.

Figure 5 provides the measured latency, and the results for the NSFNET and EON topologies are very similar: 6111 ms and 6172 ms, respectively, with an insignificant difference of 0.9%. Latency values are really high due to the selected NETCONF library and implementation. It can be observed that data is exchanged 5.10 s after the SSH session has been established.

B. RESTCONF

To evaluate RESTCONF, we have obtained the RESTCONF-based Swagger API definition of the topology YANG using the YANG2SWAGGER tool from ONF [41]. Once the swagger API has been obtained, using the Swagger code generator [42], we have obtained the necessary RESTCONF server that provides the NFSNET or EON topologies, depending on configuration.

Figure 6 shows the results of bit usage in the RESTCONF HTTP GET operation for each proposed topology. This operation requires 112,000 bits for NSFNET topology, while it requires up to 196,000 bits for EON topology. The EON topology requires more than 75% compared to NSFNET. This difference also can be explained by the difference in the bit usage between NSFNET and EON topologies. In Fig. 6, we can observe the results for using RESTCONF with a TLS security layer. HTTPS GET NSFNET topology requires 136,000 bits, increasing the necessary bits due to TLS by 21%. For the EON topology, it requires 216,000 bits, an increase of 10%.
Figure 7 provides measured latency results. For the NSFNET and EON topologies, the results are very similar: 2.883 ms and 5.109 ms, respectively, with a significant difference of 73%. In addition, the RESTCONF HTTPS latency results for NSFNET and EON also are similar: 6.855 ms and 7.517 ms, respectively. The increase of latency due to the TLS is 137% and 47%, respectively.

C. gRPC

To evaluate the gRPC protocol, we will use Listing 2, which includes the protocol buffer definition of the topology service. The described service is equivalent to the previously used YANG-based one. To compile the used protocol buffer and provide client and server stubs, the tools in [43] have been used. The same NSFNET and EON topologies have been used for evaluation of the protocol.

Figure 8 shows the results of bit usage in the gRPC RPC GetTopology operation for each proposed topology. This operation requires 21,600 bits for NSFNET topology, while it requires up to 28,000 bits for EON topology. It can be observed that protocol buffer encoding reduces the bit usage in NSFNET and EON topologies by 81% and 86%, respectively. In Fig. 8, we can also observe the results for using gRPC with a TLS security layer. gRPC with TLS NSFNET GetTopology requires 44,000 bits, increasing the necessary bits due to TLS by 104%. For the EON topology, it requires 50,400 bits, representing an increase of 180%, when compared to the EON topology with gRPC without TLS.

Figure 9 describes the measured latency results. For NSFNET and EON topologies, the results are very similar: 1.159 ms and 1.430 ms, respectively. Compared to the RESTCONF protocol, there is a significant difference of 60% and 72%, respectively. The results for gRPC with TLS in NSFNET and EON are similar: 26.47 ms and 30.75 ms, respectively. The increase of latency due to TLS is very significant because it increases an order of magnitude.

D. gNMI

Using YANG Go Tools (YGOT) [44], we generate the necessary Go language bindings for the previously presented YANG topology in Listing 1. The proposed library uses JSON encoding. Once obtained, a gNMI target (server) is executed using the NSFNET and EON topologies. The GNMI Get client is used to retrieve the topological information. Results might be different in gNMI from that obtained in gRPC, because different libraries and programming languages have been used.
Evaluating gNMI use as a basis for gRPC Python libraries will be left for future work.

Figure 10 shows the results of bit usage in the gNMI GET operation for each proposed topology. This operation requires 48,800 bits for the NSFNET topology, while it requires up to 78,400 bits for the EON topology. It can be observed that although JSON is used for exchange, the introduction of gNMI protobuf reduces the bit usage for both by 60% compared to the RESTCONF results in NSFNET and EON topologies. In Fig. 10, we can also observe the results of using gNMI with a TLS security layer. gNMI GET requires 78,400 bits, increasing the necessary bits due to TLS by 60%. For EON topology, it requires 106,400 bits, representing an increase of 36%.

Figure 11 shows the results for latency in NSFNET and EON topologies are very similar: 2.795 ms and 3.277 ms, respectively. Compared to the RESTCONF protocol, there is a decrease of 3% and 36%, respectively. Regarding gNMI with TLS, the latency results for NSFNET and EON are similar: 8.132 ms and 8.550 ms, respectively. The increase of latency due to TLS is 190% and 161%, respectively, compared to without TLS.

E. Summary

Tables 1 and 3 present an overview of the presented and evaluated protocols. First, they analyze the supported data modeling languages. NETCONF and RESTCONF protocols support only YANG, while gRPC only supports protocol buffers. gNMI is the only protocol that can handle both data models. The tables also detail the transport protocol stack, which has been detailed in the respective protocol subsections. Third, the message encoding is referenced. NETCONF supports XML encoding, while RESTCONF uses both JSON and XML. gRPC uses byte encoding and gNMI supports both JSON and byte encoding. Later, the exchange of capabilities (supported data models) is detailed. Another important feature is support for multiple datastores, which allows the ability to distinguish between config, state, or operational data). Datastore locking possibilities is a key feature to allow multitenant support for devices, as well as a multi-access configuration. Unfortunately, this feature is still only supported by NETCONF. Finally, the supported security protocols are described: NETCONF always uses SSH, while the rest support TLS as optional.

Table 2 shows a summary table regarding the presented data models for control and management of optical networks and equipment. ONF Transport API and IETF TEAS compete as feasible solutions for an NBI of a transport SDN controller. OpenROADM focuses on control and management of disaggregated ROADMs. However, in the latest versions, it provides control for optical network control and management. OpenConfig, on the other hand, focuses on router and line card configuration. All models are described using YANG; thus, we can deduce that it is widely accepted by the industry. The model complexity can be evaluated through the learning curve of the model, as well as the documentation needed to adopt the selected data model, including the length of the data model itself. T-API and OpenConfig provide simpler (and shorter) data models. Maturity can be evaluated by the stage of implementation of the data model, starting from demonstration, up to interoperability tests and production deployments. Finally, we detail the type of agreement, if it is produced inside an SDO or a multisource agreement (MSA).

5. CONCLUSION

This paper justifies the need for data models and protocols to control and monitor optical networks and optical network elements.

We have reviewed the current state of the art with significant contributions to each of the proposed solutions. The introduced protocols (NETCONF, RESTCONF, gRPC, and gNMI) have been evaluated with the same data models and under the same scenarios to provide numerical insights for each protocol in terms of the payload length and latency. This information is required to better propose optical networks and equipment that shall be adapted to the necessary use cases and scenarios.

The authors also have introduced the concept of a multi-SDO SDN controller, as well as presented the main characteristics of ZTON.

For further study, it will be useful to perform some experiments on real optical networks and devices. The results will help our understanding of the performance of the proposed protocols when a real optical data plane is being used, and we can learn how that data plane affects the performance of each protocol.

Funding. European Commission (INSPIRE-5G+ 871808); Ministry of Internal Affairs and Communications (JPM0010316); Ministerio de Economía y Competitividad (RTI2018-099178-B-I00).

REFERENCES

1. M. Channegowda, R. Nejabati, M. R. Fard, S. Peng, N. Amaya, G. Zervas, D. Simeonidou, R. Vilalta, R. Casellas, R. Martinez, R. Muñoz, L. Liu, T. Tsuritani, I. Morita, A. Autenrieth, J. P. Elbers, P. Kostecki, and P. Kaczmarek, “Experimental demonstration of an OpenFlow based software-defined optical network employing packet, fixed and flexible DWDM grid technologies on an international multi-domain testbed,” Opt. Express 21, 5487–5498 (2013).
37. R. Vilalta, C. Manso, N. Yoshikane, R. Muñoz, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, “Telemetry-enabled cloud-native transport SDN controller for real-time monitoring of optical transponders using gNMI,” in European Conference on Optical Communication (ECOC) (IEEE, 2020).

38. R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni, “Comprehensive survey on T-SDN: software-defined networking for transport networks,” IEEE Commun. Surv. Tutorials 19, 2232–2283 (2017).

39. “Pyang: an extensible YANG validator and converter,” 2021, https://github.com/mbj4668/pyang.

40. “PyangBind,” 2021, https://github.com/robshakir/pyangbind.

41. “Yang2swagger generator,” 2021, https://github.com/bartoszm/yang2swagger.

42. “Swagger codegen,” 2021, https://github.com/swagger-api/swagger-codegen.

43. “gRPC quick start,” 2021, https://grpc.io/docs/languages/python/quickstart/.

44. “gNXI tools,” 2021, https://github.com/google/gnxi.