Experimental validation of L3 VPN Network Model for improving VPN service design and provisioning

Samier Barguil
Universidad Autonoma de Madrid
Madrid, Spain
samier.barguil@estudiante.uam.es

Oscar Gonzalez de Dios
and Victor Lopez Alvarez
Telefonica I+D
Madrid, Spain

Roque Gagliano and Ignacio Carretero
Cisco Systems
Pully, Switzerland and Madrid, Spain

Ricard Vilalta
Centre Tecnologic de Telecomunicacions de Catalunya (CTTC/CERCA)
Casteldefels, Spain

***Abstract*—Current approaches to automate the design and configuration of transport services face several challenges. Firstly, there is a lack of open interfaces between the Operational Support Systems (OSS) and the network layer. The commercially available interfaces to Network Management Systems (NMSs) or directly to the devices are typically proprietary and with limited programmability. Secondly, for the case of IP/MPLS networks, the definition of how a service is built is particular to each operator.**

To solve this situation, this work presents an approach based on a hybrid SDN in which the definition of a VPN service is made with a modeling language and is not limited to a particular operator design nor a particular set of vendor devices. The IP/MPLS network topology and the VPN services are programmable via standard APIs with RESTCONF/YANG from an SDN Controller. This facilitates the operator on the one hand designing a new VPN service and its full automation. This work demonstrates, for the first time, that such automation is possible with proposed draft standards from IETF and validates its implementation, comparing the time consumed in the controller to process to different workflows from the same model.

***Index Terms*—L3VPN, IP Networks, SDN, Service Provisioning**

I. INTRODUCTION

Network operators are in continuous evolution to satisfy the changing and variable service demands of the end-users. Such evolution affects not only to the service offering but also to the network supporting the delivery of such services. A crucial part of this network is the transport infrastructure. Transport networks are responsible for forwarding aggregated traffic demands from end-users consuming different services across cities, regions, or continents. Transport network architecture was conceived for static scenarios, where were the traffic patterns usually follow the preconceived flow. This assumption allows configuring the network as a slow process since there was no need for the operator to reconfigure the deployed resources dynamically.

Software-Defined Networking (SDN) originally [1]–[3] came with the concept of the decoupling of control and forwarding planes in the network elements. A centralized controller with the complete network view does the control plane functions. It runs intelligent algorithms and applications (either as part of the controller or on top of it) and instructs the nodes accordingly. This original view, when ported to real carrier networks [4], has been evolved slightly towards an architecture where the centralized control assists the network elements on the forwarding tasks. This hybrid SDN approach provides a single and unified control environment for network operations and at the same time, optimizes the usage of network assets. The network elements yet retain control capabilities (in some cases, alleviating some signaling tasks), but leveraging on the centralized controller for end-to-end and cross-layer actions through programmable interfaces.

The use of the SDN principles is aimed to simplify the network operation and to allow a fast reaction and adaptability to network changes motivated by traffic variability or simply because of service configuration. Besides network element control functions, SDN is also considered as an entity to provide support for management functions, such as a collection of real-time information that could permit the automatic configuration creation and activation in network elements, as triggered by the Operational Support Systems (OSS). Day to day network operation for a service provider involves a set of complex and technologically specific tasks. Some of these include periodic operations as capacity planning, traffic forecast analysis, network convergence optimization, new devices installation, or routine tasks like troubleshooting, patching, support in maintenance windows, or service provisioning. Commonly, the first set of tasks requires a high specialization and in-depth technological knowledge. However, the second set is mainly repetitive.

Since years ago, the usage of scripts to automate network tasks has become common. The usage of those scripts combined with some automation frameworks such as Ansible [5] or libraries like Expect [6] or Paramiko [7] are the highest level of automation found in service providers. Lately, similar works has been done using more modern tools such as

NETCONF/RESTCONF [8]–[10] interfaces and YANG [11] abstract models.

In [12] the authors have mentioned some of the issues regarding MPLS networks and briefly described their experience using SDN for MPLS traffic engineering. However, in this work, they have eliminated the core routers (P) part of the network. They have directly linked each pair of the PEs using OpenVSwitch2 and Mininet; due to lack of support of more than one MPLS label on the emulates switch. In this work, they have used Openflow but not Netconf neither a set of YANG models; so this solution may not be something valid for a carrier-grade scenario. In [13] the work enables automatic setup and tear down of VPN instances via an easy to use web portal, without needing the help of network administrators to do the manual configuration of the network switches. In this work, a rest interface was exposed in the network controller to support all the OpenFlow configuration possibilities. However, the data model structure remains limited to devices supporting Openflow. Newer works includes new features such as the network slicing for 5G deployment [14]. Additional efforts available are focus, on solution the Layer-2 services (L2VPNs) provision [15]–[17].

To solve this problem, several YANG models have been released to support the service provisioning based on NETCONF/RESTCONF [8], [9] interfaces and YANG [11] abstract models.

In this paper, we present the intent-based configuration of the MPLS L3 VPN services in a carrier grade ecosystem. A commercial SDN domain controller using RESTCONF NBI (Northbound Interface) is tested. The controller implements a YANG network model named L3NM (A Layer 3 VPN Network YANG Model) used to abstract the complexity of the network configuration. As the main result of this work, the time consumed by two ways to use the same YANG data model to request and delete the service at controller and network level were compared.

This paper has the following structure: Section II explains the functional architecture for an IP SDN solution. Section III explains the Network Service Models used for the L3VPN definition. Section IV explains the experimental validation and the results obtained. Finally, section VI concludes this paper.

II. ARCHITECTURAL PROPOSAL

IP networks usually follow a hierarchical model, mixing equipment from different vendors. The IP routers are interoperable at data and control plane level (e.g., routing protocols such as IS-IS, OSPF, or BGP). Due to scalability reasons, the IP networks have in IP subdomains to confine the routing and control protocols on their particular domains.

Figure 1 shows the role of IP SDN Controller (IPSDNc) under the iFusion network architecture defined by Telefonica [18]. This architecture solution focuses on a single, multi-vendor controller responsible for configuring all the IP network elements. The Northbound Interface (NBI) of the controller is based on standard models defined in YANG. To access it, the protocol supported is RESTCONF with JSON encoding.

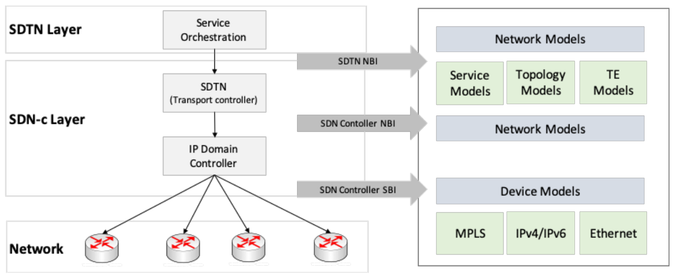

Fig. 1. IP SDN Domain Controller deployment

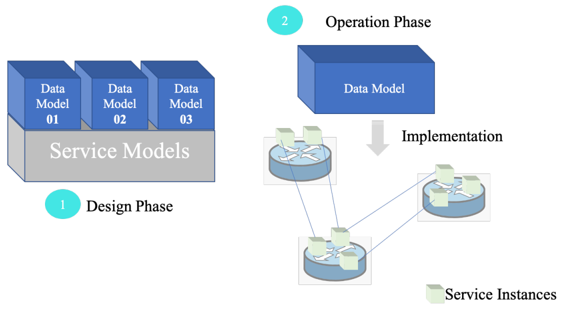

Fig. 2. Design Phase and implementation phases of a MPLS VPN Service

The NBI allows the programmatic configuration of all elements participating in a service. Thus, the IPSDNc must handle the service creation requests delivered by the service orchestration layer and translate it to device-specific configuration. For this task, the target Southbound Interface (SBI) of the IPSDNc is to make a vendor-agnostic device configuration using the NETCONF protocol and a common data model. The set of device configuration data models for the IP segment is still under development, but a lot of YANG definitions have been made in Openconfig or IETF until now.

III. LAYER 3 VPN PROVISIONING

L3VPNs are widely used in carrier-grade networks to deploy 3G/4G/5G, fixed, and business to business services. Traffic policies can be applied in these services to reuse the same transport network, and it also makes it feasible to combine access technologies in a common MPLS transport network.

In a service provider network to improve the L3VPN lifecycle management, two phases need to be analyzed (fig. 2):

- **Design phase**: It Includes the interactions from Support Systems to the network controller to instantiate a new service. Until now, some service models have been defined. These works include generic purpose modeling languages like TOSCA or YANG [19] used with protocols like ReSTconf.
- **Operation phase**: It includes the service deployment on the devices and it's operation in the live network. It implies the ability of the network controller to configure devices from different vendors.

The service providers can execute these two phases in different timeframes and with separate teams. However, the set of requirements and parameters defined in the design phase would limit the degree of automation provided during the implementation phase.

A. Design Phase: Layer 3 VPN Network API

L3NM [19] is a YANG data model to manage and control the VPN Service configuration based on the service provider requirements: Create and Operate a Multi-point L3VPN Service.

The L3NM model is focused on the parameters to create and operate the service based on the provider-edge (PE) router requirements. The data model can be used to facilitate communication between the service orchestrator and the Network controller. The L3NM model does not keep any of the commercial customer information.

B. Operation phase: SBI Models Support

To instantiate the services on the devices notably YANG data models are available from standard bodies such as IETF, Broadband Forum, 3GPP, MEF and others. Additionally, a number of interest groups have been formed in recent years to create common data models for a specific group of interest. This is the case of OpenConfig which started by addressing the needs of Cloud Service Providers but is now expanding to network service providers and beyond.

As in any standard, data models will not address all the capabilities of any given devices. Therefore, device vendors are implementing their own (many times called native) data models that do cover all the capabilities for a given device type and operating system version.

The typical way these overlaps of different YANG models are implementing is by a native implementation of NETCONF with direct or indirect (for example via the CLI API) access to the configuration database. Consequently, there are several nuances that operators need to be aware when planning which southbound data models to select:

- Typically, only the native data models provide full access to all the device capabilities. This is especially true for some innovative or recent features.
- There will be overlaps where a single configuration object (let's say interface configuration) can be accessible via different modules.
- As the semantic of the different modules are not guaranteed to play well together, a clean cut between modules for a "mix scenario" is not given.

For those use cases where we have defined a target southbound data model, we also need to consider the lifecycle management of these modules, in special as:

- Device vendors will deviate between themselves on what and how they implement standard data models. These deviations are supported by the YANG specifications but in essences means that for a controller, two southbound standard data models would not be equal.
- Device vendors will deviate on how they implement the different data models (even native) across their platforms and operating systems versions. Different platforms and operating system versions have different hardware and software capabilities, and therefore cannot expose the same APIs.
- YANG data modules evolve into new releases, these new releases may or may not follow the RFC6020 and RFC7950 rules for module upgrade. This means that the same namespace may have different meanings depending to which device you are talking to, which violates one of the main principles for YANG (RFC 7950 section 5.3).

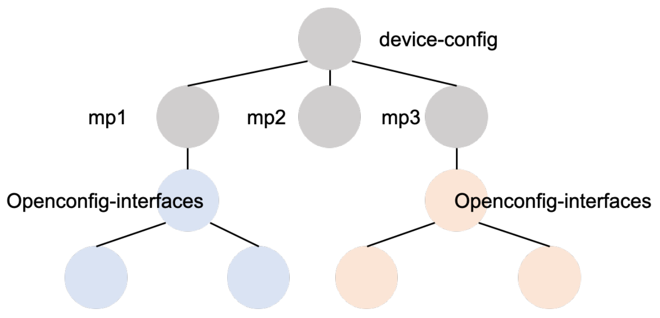

Fig. 3. Schema mount in RFC8528 used to map two implementations of openconfig-interfaces as SBI

The NETCONF/YANG standards do not mandate any specific "workflow", i.e. if the target configuration should be executed as "all at once" or in a specific sequence. Moreover, dependencies between nodes in a YANG models are not annotated via any standard capability. This means that two standard compliant vendors may require different number of total commits for the same target configuration.

We can now summarize then what should we require from a NETCONF/YANG southbound implementation in a network controller (besides supporting the mandatory standard documents) to make it functional in current real deployment scenarios:

1) The controller should be able to support southbound devices that provide their own mix of standard and non-standard device data models, where the operator should be free to choose what combination it desires for every device group.
2) The controller should support multiple copies for the same namespace to allow different implementations across device types and vendors.
3) The controller should support by default the "all at once" single step. Conversely, the controller should support devices that require multiple commits via a different behavior.

One of the main technologies that may help vendors to address these concerns is the use of the YANG Schema-Mount standard (RFC8528). Thanks to this technology, different versions of the same namespace could be added to different nodes in the controller configuration and state trees. This means that if we create a list of implementation, we could "mount" as many variants as desired depicted in fig. 3.

Intents = **5**		Intents = **3**	
INTENT	NUMBER OF PARAMS	INTENT	NUMBER OF PARAMS
Create Site	5	Create Site	5
Create VPN Service	4	Create VPN Service - VPN Node - IE Profile	16
Create Import/export profile	5		
Create VPN Node	7		
Creace VPN Network Access	25	Creace VPN Network Access	25
Total	**46**	**Total**	**46**

Fig. 4. Intents and numbers of parameters used in each test

IV. Experimental Validation

The network set-up and the test done are explained in this section.

A. Network Set-up

For our experiments, we have chosen the data models for Cisco IOS-XR configuration and the Cisco Network Services Orchestrator (NSO) as the IP Controller. The Cisco NSO controller implements the L3NM as the service model in the NBI.

B. L3VPN Provisioning Workflow

The L3NM uses the YANG Modeling Language to describe an MPLS L3VPN service. The model defines two main containers, one for sites and one for the VPN-services. Each of these containers includes a list to create a set of sites and services, respectively. Each item (site or service) is uniquely identified in the list by their id.

C. Experimentations

To evaluate the YANG model defined for the L3VPN and how its design can impact the service instantiation, the following tests were done:

1) Define the minimum set of intents to request the service: Based on the YANG model structure, the minimum number of intents required to complete the VPN onboarding process is three (3).
2) Compare the minimum set of Intents against the number of calls suggested by the model creators: The model specification suggest five (5) intents as the optimal number of requests.
3) Create a hundred (100) services/tests using both provisioning methodologies and capture the time consumed by the controller to process each service request, measured as:

$$\text{Net time} = \text{Creation} + \text{Deletion} \qquad \text{s} \qquad (1)$$

Based on this, two workflows to request the VPN service to the network controller were defined. The workflows differ on N the number of calls done to create the service but globally the total number of parameters is identical, as detailed in fig. 4.

Each workflow follows the next steps: ALl the creation intents are sent to the IPSDNc using RESTCONF/YANG. IPSDNc computes the call and, if the request is valid, the controller returns the creation confirmation code. The network controller does the service deployment on the network only when the intent related to the VPN-Network-Access is received. The other intents create resources at the controller level only.

	Total Time [Sec]		**Deletion Time** [Sec]		**Creation Time** [Sec]	
	Mean (μ)	Std (σ)	Mean (μ)	Std (σ)	Mean (μ)	Std (σ)
Intents = 5	27,36	2,29	12,81	5,55	14,56	3,56
Intents = 3	25,55	2,56	12,64	5,54	12,91	3,23

Fig. 5. Performance results obtained during the testing

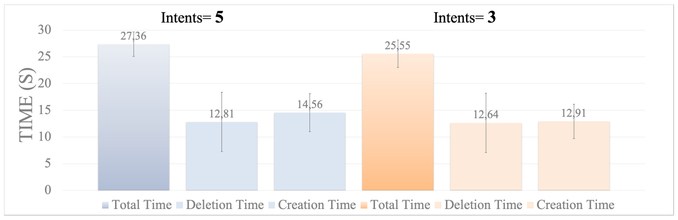

Fig. 6. Performance results obtained during the testing

V. Results

The total time consumed by the IPSDN controller to create and delete a VPN service form the network was measured. The mean u values and the standard deviation d is computed in the for the 100 experiments as depicted in fig. 5. The main difference was found during the creation process. During creation, each additional call introduces a delay of $0,82$ seconds per additional call. This number seems low, however a change in the model structure, for example, avoid the cross-references between the containers of sites and VPN services, can allow a single call creation process. This potentially will reduce the creation time, a bit more. This kind of behavior can be treated as a design rule to make the model implementation efficient.

VI. Conclusions

Telecommunication providers are forced to continuously evolve their networks to provide a better service for their customers. Thus, Network automation initiatives are highly necessary. The implementation of these initiatives requires their adaptability and support of brownfield scenarios in operating networks. This support must include not only the configuration of the devices and all the possible services provisioned in the day-to-day operations across the service provider networks.

Firstly, this article provides an overview of the latest state-of-the-art standardization efforts of the L3 VPN life cycle management. Secondly, the work implements for the very first time the L3NM. Two workflows were used to request the service using the same network model. The time consumed using each workflow to implement the service on the device were measured and compared. The work demonstrates that YANG design rules can directly affect the time consumed to create services and explains how device models implementations can limit the deployment of common interfaces

Acknowledgment

This work was supported partially by the European Commission H2020-ICT-2016-2 METRO-HAUL project (G. A. 761727), H2020-ICT-2018-2020 5G-VINNI (G. A. 815279) and Spanish AURORAS (RTI2018-099178).

References

[1] O. S. Brief, “Openflow-enabled sdn and network functions virtualization,” *Open Netw. Found*, vol. 17, pp. 1–12, 2014.

[2] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “Nox: towards an operating system for networks,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[3] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, “Applying nox to the datacenter.,” in *HotNets*, 2009.

[4] S. Bemby, H. Lu, K. H. Zadeh, H. Bannazadeh, and A. Leon-Garcia, “Vino: Sdn overlay to allow seamless migration across heterogeneous infrastructure,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 782–785, IEEE, 2015.

[5] D. Libes, *Exploring Expect: A Tcl-based toolkit for automating interactive programs*. " O’Reilly Media, Inc.", 1995.

[6] L. Hochstein and R. Moser, *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. " O’Reilly Media, Inc.", 2017.

[7] M. Zadka, “Paramiko,” in *DevOps in Python*, pp. 111–119, Springer, 2019.

[8] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network configuration protocol (netconf),” 2011.

[9] A. Bierman, M. Bjorklund, K. Watsen, and R. Fernando, “Restconf protocol,” *IETF RFC 8040*, 2017.

[10] R. Vilalta, R. Muñoz, G. Landi, L. Rodriguez, M. Capitani, R. Casellas, and R. Martínez, “Experimental demonstration of the bluespace’s nfv mano framework for the control of sdm/wdm-enabled fronthaul and packet-based transport networks by extending the tapi,” in *2018 European Conference on Optical Communication (ECOC)*, pp. 1–3, IEEE, 2018.

[11] M. Bjorklund, “The yang 1.1 data modeling language,” tech. rep., RFC 7950, DOI 10.17487/RFC7950, August 2016,¡ https://www. rfc-editor. org …, 2016.

[12] B. Mirkhanzadeh, N. Taheri, and S. Khorsandi, “Sdxvpn: A software-defined solution for vpn service providers,” in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*, pp. 180–188, IEEE, 2016.

[13] R. van der Pol, B. Gijsen, P. Zuraniewski, D. F. C. Romão, and M. Kaat, “Assessment of sdn technology for an easy-to-use vpn service,” *Future Generation Computer Systems*, vol. 56, pp. 295–302, 2016.

[14] R. Rokui, H. Yu, L. Deng, D. Allabaugh, M. Hemmati, and C. Janz, “A standards-based, model-driven solution for 5g transport slice automation and assurance,” in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 106–113, IEEE, 2020.

[15] P. L. Ventre, S. Salsano, M. Gerola, E. Salvadori, M. Usman, S. Buscaglione, L. Prete, J. Hart, and W. Snow, “Sdn-based ip and layer 2 services with an open networking operating system in the geant service provider network,” *IEEE Communications Magazine*, vol. 55, no. 4, pp. 71–79, 2017.

[16] K. A. Noghani, C. H. Benet, A. Kassler, A. Marotta, P. Jestin, and V. V. Srivastava, “Automating ethernet vpn deployment in sdn-based data centers,” in *2017 Fourth International Conference on Software Defined Systems (SDS)*, pp. 61–66, IEEE, 2017.

[17] Q. Wu, M. Boucadair, C. Jacquenet, L. M. C. Murillo, D. R. Lopez, C. Xie, W. Cheng, and Y. Lee, “A framework for automating service and network management with yang,”

[18] L. Contreras, Ó. González, V. López, J. Fernández-Palacios, and J. Folgueira, “ifusion: Standards-based sdn architecture for carrier transport network,” in *2019 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–7, IEEE, 2019.

[19] A. Aguado, G. d. D. O, V. Lopez, D. Voyer, and L. Munoz, “A layer 3 vpn network yang model,” draft draft-ietf-opsawg-l3sm-l3nm-01, ietf-opsawg, May 20 2019.