Vendor Agnostic Network Service Orchestration with Stacked NSO Services

Marc Koerner
Department of Energy / ESnet / Lawrence Berkeley National Laboratory
Berkeley, USA
MKoerner@{ES.net; LBL.gov}

***Abstract*—The Energy Sciences Network (ESnet) is the Department of Energy's internal wide area network provider, delivering connectivity for all US national laboratories including some satellite sites in Europe. The ESnet network is a highly reliable and high bandwidth network, which transports vast amounts of data between laboratories and supercomputing facilities. Thus, ESnet is supplying the scientific community with the connectivity requirements for all sorts of data analytics and simulations.**

One of the major goals within the ESnet6 deployment was to have an orchestrated and fully automated network configuration management system. Hence, ESnet is leveraging tools like the Cisco Network Service Orchestrator (NSO) to deploy router configuration in a centralized and also service oriented fashion. After building the first iteration of services following a significant optimization in the router configuration and eventually deployment time, ESnet decided to take advantage of the accumulated knowledge during the first implementation and started to revise the NSO service architecture. The first prototype using the revised architecture is currently getting implemented in scope of our management router based configuration service. This paper will elaborate on the advantages of minimum functional configuration based stacked service architecture in comparison to plain networks service model based design.

***Index Terms*—Configuration Management, Intent-based Networking, Network Service Orchestration**

I. Introduction

With emerging technologies like Software-Defined Networking (SDN) and Network Function Virtualization (NFV) the foundation for a rethinking of the operation of networks was created. The central control paradigm and the abstraction concept introduced by virtualization triggered a lot of new network management and operational concepts. The entire development started around the OpenFlow technology and quickly extended beyond that. However, the main idea remains the same: a centralized network control entity is managing the network based on the networking fabric paradigm. This means the network acts as one entity and is no longer controlled in a decentralized way. The key is to follow an abstracted network function driven workflow, which will lead to an automated device configuration provisioning across multiple network nodes or even the entire network. The logical consequence of this approach is an intent driven automation. Based on this concept, network operators can configure a network service, like a Virtual Private Network (VPN) or a Label Switched Path (LSP), on demand and in a global manner. ESnet has adopted that technology and implemented a fully automated intent driven network orchestration stack, leveraging multiple tools to provide an automated provisioning of customized network services. Introducing the entire ESnet software stack would exceed the scope of this paper. The focus of this paper will be on the design and implementation of the vendor agnostic stacked network service architecture for NSO.

Fig. 1. ESnet6 network and user facilities

The remaining paper is structured as follows: a related work in section II and background in III. This is followed by an overview of the current implementation in section IV-A, the lessons learned in IV-B, and the revised approach described in section IV-C. Which will be followed by the results of the implemented prototype IV-D. The paper will end with a short conclusion and future directions V regarding the refactoring and integration process.

II. Related Work

Configuration management in general goes all the way back and started with some research around how to deal with general software configuration management. Introducing fundamental ideas around versioning as well as an object oriented approach [1] around composition and aggregation. Significantly later during the introduction of SDN [2] to packet switched networks, the controller logic became more complex and the controller became a mixed architectural management system also dealing with network configuration [3]. The momentum kept going and network configuration management systems and their applications moved more into the focus [4]. This also led to configuration management platforms like NSO [5]. Thus, eventually leading back to the fundamental question of how to architect services on top of network configuration management systems in an efficient manner [6].

978-3-903176-59-1

III. Background

Network configuration management in general is a very challenging task, which is getting even more complicated once devices from multiple vendors are involved. Not only that the user has to deal with different command line interfaces (CLI) even standardized application interfaces (API) like Netconf might look totally different. Moreover, hardware and software specific device implementations look completely different as well. A simple example for instance is the maximum transfer unit (MTU). While normal MTUs of 1500 bytes or more based on e.g. Q-in-Q VLAN tagging are usually not a problem at all, jumbo frame support with regards to the MTU might be slightly different by a few bytes. Hence, configuring a backbone link (BBL) between two core routers from different vendors becomes a very challenging endeavor taking up to multiple days of configuration and testing.

An even more simplified example is for instance the configuration of a loopback interface address. Different port naming schemes as well as the CLI / Netconf differences make even seemingly simple tasks like an IP address assignment somehow challenging. This is when model driven configuration management comes into play. Similar to the SDN controller principle an abstraction layer is introduced, which communicates with heterogeneous network equipment. Instead of having a unified API / agent on the devices as defined in the OpenFlow paradigm, the abstraction is moved into the controller. Configuration management tools like the Cisco Network Service Orchestrator leverage an internal API which will be wrapped by a Network Element Driver (NED). This driver is a device specific implementation of the device API and allows the configuration management system to access the configuration data. However, the device specific abstraction is not part of this implementation and remains a challenge which needs to be addressed by the service developers. The configuration management platform is just providing the required framework for building service logic on top without implementing southbound Netconf interfaces or northbound CLI/REST interfaces. It is a service development framework with a unified access model. Thus, building an intent driven service like the aforementioned BBL can significantly reduce deployment time from previously two days to 15 minutes for an up and running full functional BBL.

IV. Architecture and Implementation

This section will give a brief overview about the existing implementation as well as the architectural design guide for the revised approach.

A. Current implementation

ESnet is using the Cisco NSO to build network services. The basic set of services is configuring independent device specific configuration sections using different XML based templates for Junipers and Nokia routers, as well as some Ciena optical equipment. Static configuration information is directly mapped by structured XML code while certain model based values e.g the IPv4 and IPv6 address for a loopback interface, which is getting set based on the Yang model, is eventually handed down to the regrading template section.

ESnet's NSO services are typically composed out of the following components:

- The Yang service model which translates into a CLI command structure as well as the JSON based REST API
- Service logic is written in python which usually includes some data validators and template based variables
- The XML based device configuration template for the different devices which directly translates into device configuration

TABLE I
ESnet's NSO services

Service	Descrition	Lines of Code		
		Yang	*Python*	*XML*
Port	basic port settings: MTU, LLDP, etc	641	263	442
BBL	Configures a BBL between two routers	544	652	552
VPLS	Configures a L2 VPLS between n routers and m interfaces	527	920	570

In particular, the BBL service model relies on certain leaf references from the port and routing-domain service. Both services are getting referenced in the Yang model in order to verify that by the BBL service required configuration is in place. In this example the instantiated port and e.g. the. intermediate system to intermediate system (IS-IS) routing protocol settings which are required to set up the BBL related additional configuration information like the IP addresses on both sides. This is why the BBL is also a very good example of an intent based service which is using a configuration management system like NSO in order to create a service instance across multiple devices from possibly different vendors.

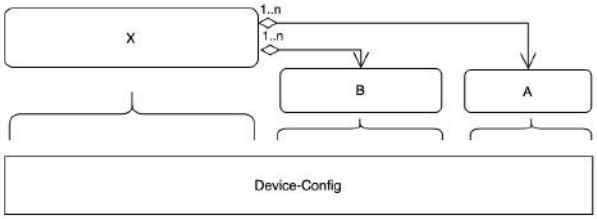

Fig. 2. UML Example of the current NSO Service Architecture

Based on different factors like for instance the amount of knobs a service is providing for a specific customer adopted sdevice setting, the service logic can grow rapidly, the same applies to the validation logic, and the configuration templating.

B. Lessons Learned

Building an intent driven service for a configuration management system utilizing heterogeneous network equipment can very quickly become a challenging task. Vendor specific

configuration and fundamental different configuration principles introduce a high level of complexity and need to be designed appropriately. The first implementation was a significant improvement compared to the diverse tools ESnet has been used before. However, following a top-down approach led to the following issues:

1) Monolithic services with a high complexity: Having a top down approach leads to mostly monolithic services with significant complexity. The initial implementation is usually looking reasonable but typically growing over time. Not covered requirements or previously not identified corner cases tend to blow up the service code base as well as the testing. The service internal value validation is getting more complex with the size and the service in general is getting harder to understand for developers.

2) Validation complexity: Service validation is basically its own challenge. It is supposed to make sure that the engineer or higher automation layer using the service is providing correct data. Moreover, it is also important for consistency and correctness within the network. This could affect internal configuration references like prefix list names associated with a certain filter as well as configuration related ids which should be unique across the system or even the network as a whole.

3) Same functionality is getting defined in multiple services: Simple configuration information like the definition of an interface as well as the regarding assignment of an IP address for that interface is getting defined multiple times. Redundancies like these increase the code base and add additional maintenance and test overhead. Adding or changing configuration parameters have to be maintained and tested in multiple places and add significant refactoring complexity.

4) Redundant data: Instead of taking advantage of referencing lower layer services which already contain the configuration information like a particular prefix list, it's getting defined in multiple places on multiple devices and can cause consistency issues once it is subject to change.

C. Revised service architecture

The current implementation provided a significant positive impact on the way ESnet's routers are getting configured. The configuration complexity for the operators was reduced and a significant time reduction accomplished. However, large monolithic services introduced their own complexity and thoroughly automated testing and documentation was required. Overlapping data is challenging when it comes to validation. In order to anticipate the identified issues the current architecture was audited. The outcome was a re-architectured design of the current service model and move towards a more granular and layered service design. In detail, the key idea is to leverage of the NSO internal opportunity to stack services and encapsulate certain configuration information into logical blocks which can be reused across higher layer services. The outcome was a move away from the linear monolithic service architecture towards a 3-tier stacked service approach.

1) Tier-1: Tier-1 services are building the vendor agnostic abstraction of simplified device configuration sections. They reduce the overall amount of configuration options but still cover every lever which is required for the services using this fundamental building block. Since they just abstract configuration information they might not be able to work independently. An example for tier-1 services could be a VLAN or a prefix-list. Another bold concept for the tier-1 services is that instead of getting mapped to a single device the service model will have a device list which is not mandatory. Thus, the config information can exist independently and later mapped against multiple devices in order to generate the same configuration information up to N times. An exception to this model are services which are bound to a unique resource, for instance the physical port on a particular device.

2) Tier-2: Tier-2 services introduce a basic service abstraction. They are composed out of tier-1 services and build the first layer of functional services. In order to compose tier-1 services into a functional block they may introduce some additional device configuration which they use to glue tier-1 services together. Tier-2 services can also be stacked to build further functional service blocks.

3) Tier-3: This is the highest layer of services which are mostly fassade services. They are composed out of tier-2 and tier-1 services and build an operation foundation for ESnet's network operation and customer facing product portfolio. They only expose a minimal amount of variables which allows operators to bring up the service with mostly default settings. These default values are hard coded within the tier-3 service template and can be adjusted by editing the tier-2 or tier-1 services in case a higher layer of adaptation is needed. Again, the automation software or a qualified user can make changes to the underlying services instantiated by the fassade service in order to tweak values individually. These services have zero device configuration in their templates and rely only on the lower layer services. They reflect a fully abstracted and device agnostic definition of a network service.

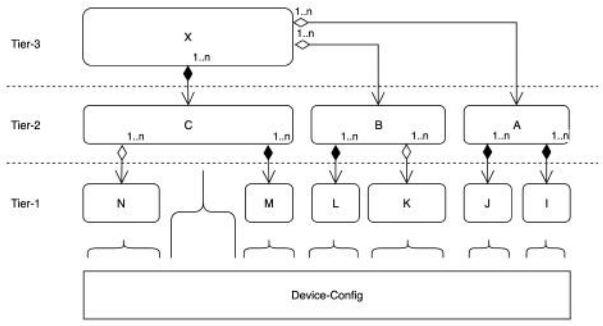

Fig. 3. UML Example of the New NSO Service Architecture

The new architecture is applying object oriented (OO) programming concepts to the configuration management service architecture. It works based on a combination of composition and aggregation of services and leverages the NSO internal Yang leaf reference mechanism as a key principle for an initial data validation as well as a pre-selection of possible sub-services. Most services and their data configuration data can

exist without even getting applied to a device. However, once a tier-3 service is getting instantiated all related tier-2 and tier-1 services are getting either a device information or instantiated all configuration information is coming together and applied to the corresponding device or devices. Thus, large chunks of configuration can be generated and applied with the pre-existing data from lower layer services in the configuration database (CDB) and some minimal input based on the required information for the tier-3 service.

D. New Tier-3 Architecture Prototype

Esnet is deploying two different categories of routers on operational sites. The first category are the core routers, which are building the core network. The second category are management routers, which are used to connect certain onsite periphery. They offer mostly management access across various equipment which is usually co-located with the core routers. This can be for instance network performance test equipment, optical transponders, and other hardware. MGMT routers share certain network management services like DHCP, NTP, or syslog based on relay agents. These services are getting whitelisted with with ACLs. The ACLs again are referencing specific prefix lists which contain specific prefixes for bidirectional traffic. Similar to the idea behind the afore mentioned system service, where the router's loopback interface addresses are getting configured and the QoS parameter based queue configurations, the management router base configuration service is supposed to provision a certain foundation of configuration options. This is the first service which was developed as a prototype following the new design pattern. The result is very impressive, since after setting up all dependencies for the management router base configuration like the regarding VLANs, prefixes, and ACLs in NSO as tier-1 services, a complete base configuration can be generated by just providing the IP addresses for the Integrated Routing and Bridging (IRB) interfaces for in the tier-3 service. This can basically be repeated for each device this particular service configuration should get applied to without the need to change any other data. While creating a perfect balance of service abstraction and flexibility a significant reduction in code complexity could be achieved compared to the first implementation. Just to be fair, the current prototype does not introduce any evaluation logic in python at the moment. However, service size in general was reduced dramatically and modules can get reused within multiple services now. One example from the prototype is the IRB service which is getting instantiated five times within the MPR-base-config service. Moreover, ACL and IPL services are getting used within the IRB service and code is getting applied multiple times to the final device configuration without any repetitions inside the NSO XML templating.

TABLE II
ESNET'S NEW MGMT ROUTER BASE CONFIG SERVICES

Service	**Descrition**	**Lines of Code**		
		Yang	***XML***	***Tier***
IPL	Configures a list of prefixes	63	38	1
ACL	Configures the ACLs using internal refs to IPL	61	100	1
Interface	Configures the abstract construct of a logical interface including IPs	133	56	1
VLAN	Configures a VLAN entry	60	19	1
IRB	Configures an IRB interface using the VLAN, Interface and ACL service	82	61	2
MPR-base-config	Creates and configures a complete MGMT router base config with multiple IRBs	92	121	3

V. Conclusion and Future Work

We were able to reduce the code complexity and are further investigating and evaluating the new architecture. Additional base config related services like the static-routing service are currently in development. We will further continue to implement additional tier-1 services and refactor the pre-existing service logic while keeping the service model as a tier-3 facade service. This will help with the transition since the existing REST API endpoint to our automation software on top will remain the same even though the underlying service architecture is getting a complete overhaul. Moreover, the existing system and integration tests can be used to validate the correctness of the refactored layered architecture and should still produce the identical router configuration.

Acknowledgment

This work was created within the scope of the ESnet6 project and the regarding funding of the Department of Energy. I would also like to acknowledge all the work from people across the organization involved in ESnet6 who did not directly participate in this paper.

References

[1] Hal Render and Roy Campbell. 1991. An object-oriented model of software configuration management. In Proceedings of the 3rd international workshop on Software configuration management (SCM '91). Association for Computing Machinery, New York, NY, USA, 127–139. https://doi.org/10.1145/111062.111079

[2] 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 Computer Communication Review, April 2008.

[3] Open Daylight, https://www.opendaylight.org/

[4] Xu Chen, Z. Morley Mao, and Jacobus Van der Merwe. 2009. PACMAN: a platform for automated and controlled network operations and configuration management. In Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09). Association for Computing Machinery, New York, NY, USA, 277–288. https://doi.org/10.1145/1658939.1658971

[5] Cisco Network Service Orchestrator (NSO), https://www.cisco.com/c/en/us/products/cloud-systems-management/network-services-orchestrator/index.html

[6] S. van der Meer, J. Keeney, L. Fallon, S. Feghhi and A. de Buitléir, "Large-scale Experimentation with Network Abstraction for Network Configuration Management," 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2019, pp. 60-65, doi: 10.1109/ICIN.2019.8685922.