uABNO: A Cloud-Native Architecture for Optical SDN Controllers

Ricard Vilalta[1], Juan Luis de la Cruz[1], Arturo Mayoral López-de-Lerma[2], Victor López[2], Ricardo Martínez[1], Ramon Casellas[1], Raul Muñoz[1]
[1]*Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Spain.*
[2]Telefónica *gCTIO/I+D*, Spain

Abstract: We present a cloud-native architecture for Optical SDN Controllers based on ABNO architecture and gRPC interfaces, which is demonstrated and evaluated. Autoscaling mechanisms for high request loads and auto-healing support are evaluated.
OCIS codes: (060.4250) Networks; (060.4510) Optical communications

1. Introduction

Software Defined Networking (SDN) is a consolidated network architecture paradigm that provides network programmability by separating the control plane logic from the data plane forwarding infrastructure. This decoupling, provides novel benefits to network operators, such as CAPEX savings by replacing dedicated hardware network equipment by software-driven network elements and OPEX thanks to faster new service introduction. SDN has been accompanied with new open standard interfaces, such as NETCONF, gRPC, OpenFlow or P4, which allow to interact with the network elements from centralized entities generally defined as SDN controllers [TEF19].

SDN controllers currently are developed as single monolithic and resource-hungry applications, which might be replicated in case of need for resiliency or extra resources. This leads to non-efficient use of the resources, does not provide scaling mechanisms for high loads of connectivity service requests, and incur on extra delays in each request, not allowing a cloud-scale number of requests. Some network vendors are experimenting with solutions that export the complexity towards applications, but are not dealing with the need to break the monolithic SDN controllers [Nokia19]. The Applications-Based Network Operations (ABNO) framework [ABNO] has been standardized by the IETF and it is based on standard protocols and components to efficiently provide a solution to the transport network orchestration.

Microservices are a software development technique that structures an application as a collection of interconnected and related services. In a microservices architecture, services are simple and detailed and the protocols are lightweight. For example, gRPC Remote Procedure Calls (RPC) [gRPC] is a protocol designed for cloud native high-performance RPC. It uses HTTP/2 as a transport protocol and uses protocol buffers encodings for transported messages. gRPC has been proven as useful in telemetry, due to its low latency and small byte overhead.

In this paper, we propose the application of microservices architecture to the development of an SDN controller. The internal architecture of SDN controller makes it clearly a good candidate for splitting it into microservices, which provide resiliency and scalability features per design. Using ABNO components, several microservices will be defined, including path computation service, NBI service, connectivity service, connection service, topology service, context service, VNTM service, transceiver service, monitoring service, and plugin services. Each microservice will interact with each other using protocol buffers and gRPC interfaces. A common gRPC interface will be provided for health check of the microservices.

This novel cloud-native architecture named uABNO (Fig. 1, left) provides multiple technological benefits, which have been clearly demonstrated in other cloud computing applications. The most significant is application resiliency, where microservices are monitored and restarted in case of misbehavior. Another benefit is application scalability, which tackles request load increase, with deployment of new instances of necessary microservices. Finally, our proposal also provides ease of integration in a cloud-native solution, considering that most upcoming network software solutions will be cloud-native. This paper demonstrates the feasibility and advantages of the proposed uABNO architecture for optical SDN networks and provides interesting results on connectivity provisioning delay, as well as it validates auto-scaling and self-healing mechanisms.

2. uABNO architecture

The proposed architecture consists on reformulating the concept of SDN controller, which is responsible for network control and management. SDN controller receives network intents from the network operator, describing how network should behave and which connectivity requests need to be addressed while properly configuring the underlying network elements. The operator's Operation Support Services and Business Support Services (OSS/BSS) are able to interact with the proposed cloud-native SDN controller, through Standard interfaces, such as ONF Transport API. Then, the SDN controller is responsible for network control and management, thus providing the necessary network dynamicity, through interaction with underlying network elements.

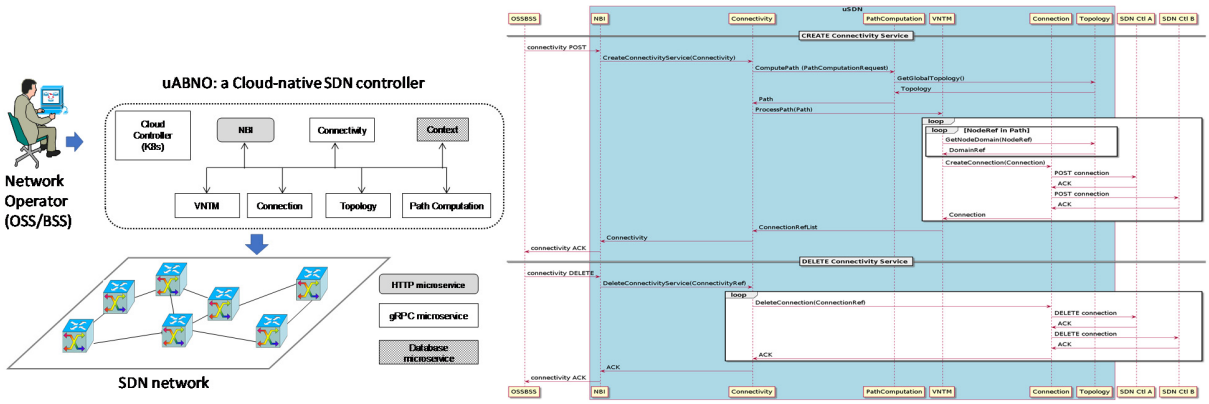

Fig. 1 (left) uABNO cloud-native architecture, (right) uABNO connectivity service create/delete workflow

The proposed uABNO internal architecture consist on microservices that interact using gRPC protocol buffers based on ONF Transport API. This conversion considers [ONF18]. Moreover, a Cloud Orchestrator (such as Kubernetes) is responsible for lifecycle management of the microservices (including health checks and resource allocation). The uABNO microservices can be classified within three types: a) Database microservice, which provides a scalable cloud native database (such as MongoDb) for storing network element topology, status and configuration, as well as connectivity services requested and connections; b) HTTP microservice, which exposes uABNO NorthBound Interface (NBI) (e.g., ONF Transport API) as a RESTconf API and translates the request to internal protocol buffers; and c) gRPC microservices, which use gRPC protocol and protocol buffers as basis for intercommunication.

Fig. 1 (left) shows proposed uABNO architecture. NBI microservice is responsible to interact with network operator's OSS/BSS and translate the connectivity requests into internal protocol buffers. The cloud-native database (Context implemented in MongoDb) is responsible for storing the context that includes controlled and managed connectivity services, connections and topologies. It interacts with their respective components in order to create, read, update and delete (CRUD) the records.

The internal architecture workflows are depicted in Fig. 1 (right). Once a connectivity service request is received, the NBI translates this request into the proper protocol buffer and sends the request to connectivity microservice. The connectivity microservice first requests a path computation to path computation microservice that requires a topology retrieval. Once a feasible path is computed, virtual network topology manager (VNTM) microservice is responsible for analyzing the need for multi-layer/multi-domain connections and generates the necessary connection requests towards the connection microservice. The connection microservice is responsible for requesting the necessary network element configuration (e.g., NETCONF, OpenFlow), or interacting with underlying SDN controllers.

The workflow for removing a connectivity service is also detailed in Fig. 1 (right). The NBI receives a connectivity service delete request, which is forwarded to connectivity microservice. Connectivity microservice requests to connection microservice the deletion of the related connections.

This proposed cloud-native architecture has several key benefits that introduce network automation: a) self-healing properties, due to the constant monitoring of microservices and restart of them in case of failure; b) auto-scaling, which allows to monitor microservice resource consumption and scale the microservice horizontally in case of overload (path computation is a resource consuming process which easily scales horizontally); c) load balancing, related to auto-scaling, in the sense that it allows to balance the load between replicated microservices; and d) automated roll-backs, which allow the declarative network status description, benefiting network operators with network programmability.

3. Autoscaling and self-healing microservices

The inherent features from a Cloud Orchestrator such as Kubernetes provide the necessary support for two key features that are required for a modern Optical SDN controller: autoscaling and self-healing. uABNO microservices might benefit from Horizontal Pod Autoscaler (HPA) component of Cloud Orchestrator. A certain amount of CPU and memory resources are requested per microservice. HPA monitors the microservice behavior and notices if resource limit is reached. Then, a new replica is generated and load between replicas is balanced in order to reduce resource utilization. This feature is useful for resource consuming microservices such as path computation.

Another significant benefit from cloud-native orchestration is the introduction of monitoring the status of the microservices, using a health check regularly (if the service is serving request or not serving). In case of service bad health (which might be caused by some blocking external component or an unexpected microservice behavior), Cloud Orchestrator removes the current deployment of the microservice and a new replica is deployed. This is known as a self-healing mechanism.

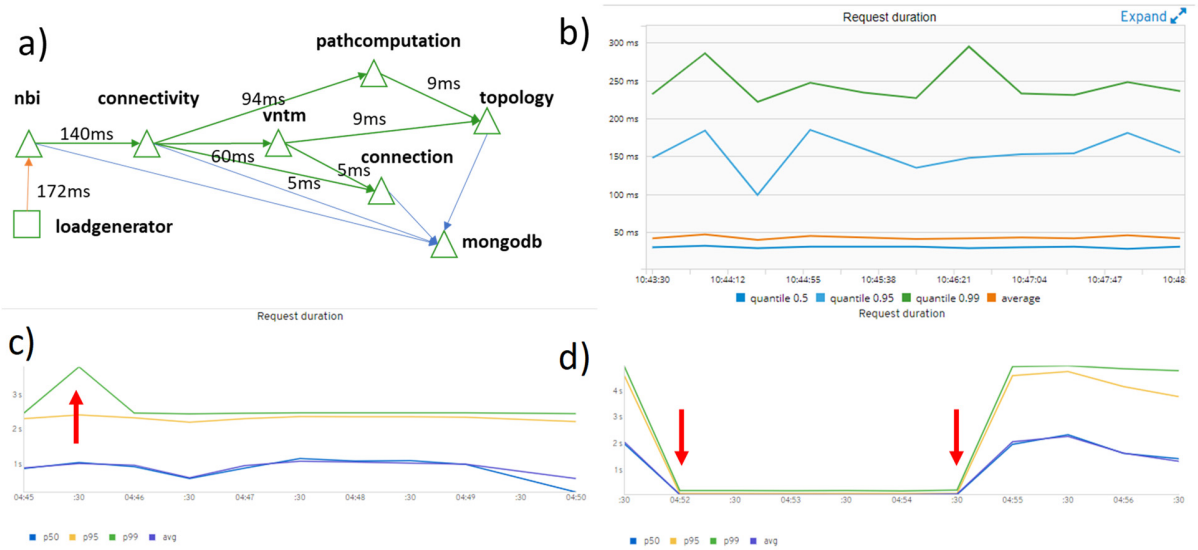

Fig. 2 a) Connectivity Service aggregated latency per component, b) CDF for connectivity service duration, c) Path Computation delay after Horizontal scaling, d) Connectivity Service self-healing mechanism

4. Experimental results

The proposed uABNO architecture has been developed based on python container-based microservices, which use gRPC and protocol buffers interfaces to communicate between them. MongoDB has been deployed for context storage, including connectivity, connection and topology information. The connection component has several plugins in order to directly control NETCONF-based network elements or interact with REST-based optical SDN controllers. In our current setup, we have simulated a 14 node NSF network, where each node is managed by a NETCONF agent. A Kubernetes 1.15 cluster of two nodes using Intel NUC with i7, 32Gb RAM, 1Tb SSD has been deployed on top of ADRENALINE Testbed Cloud Platform. Istio and Kiali have been installed in order to monitor the microservices running on the cluster. A loadgenerator microservice has also been developed in order to stress the proposed architecture and obtain significant results of optical cloud-native SDN controller.

Fig. 2.a shows the relationship between the deployed microservices. It can be observed that three types of protocol are introduced (HTTP orange/gRPC green/TCP blue). It also shows the aggregated latency (in average) between components. For example, a connectivity-service creation request might take 172ms, from these 94ms correspond to path computation, 60ms to VNTM, or 5ms to connection. Fig. 2.b shows the request duration from the NBI perspective. It can be observed that request duration varies depending if it is a connectivity service create request or a connectivity service delete request. Create requests average 150ms, while deletion requests average 30ms.

Fig. 2.c provides an example of horizontal scaling behavior, after increasing path computation complexity in order to demonstrate auto-scaling properties. Once path computation delay reaches a certain threshold (meaning that more CPU resources are needed), a new replica of path computation microservices is deployed, thus leading to a reduction on path computation delay. Fig. 2.d shows the self-healing property, after emulating an error in connectivity microservice. The orchestrator automatically restarts the microservice within the configured timeout (150s).

5. Conclusions

We have successfully validated and demonstrated a novel cloud-native architecture for Optical SDN controllers. The proposed uABNO provides a higher degree of flexibility, stability and scalability than current monolithic SDN controllers. The application of cloud-native network control and management will provide network operators with a higher degree of network automation.

Acknowledgment

The research partially funded from EC METRO-HAUL (761727) and Spanish AURORAS (RTI2018-099178-B-I00).

References

[TEF19] L.M. Contreras, et al., “iFUSION: Standards-based SDN Architecture for Carrier Transport Network”, in IEEE CSCN, October 2019.
[Nokia19] Van, Quan Pham, et al. "Demonstration of Container-based Microservices SDN Control platform for Open Optical Networks." OFC, March 2019.
[ABNO] A. Aguado, et al. "ABNO: A feasible SDN approach for multivendor IP and optical networks." JOCN 7.2 (2015): A356-A362.
[gRPC] R. Vilalta et al, “Grpc-Based SDN Control And Telemetry For Soft-Failure Detection Of Spectral/Spacial Superchannels”, ECOC 2019.
[ONF18] UML to ProtoBuf Mapping Guidelines, TR-544 v1.0-info, February 22, 2018.