Cloud-native SDN Controller Based on Micro-Services for Transport Networks

Carlos Manso, Ricard Vilalta, Ramon Casellas, Ricardo Martínez, and Raul Muñoz
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss 7, 08860 Castelldefels, Barcelona, Spain
{ricard.vilalta@cttc.es}

Abstract—Current SDN controllers are monolithic applications that are run on dedicated servers and require specific protocols for synchronization among them. These SDN controllers do not provide the required flexibility to scale-out in case of a cloud-scale number of connectivity service requests.

In this demonstration, we present the control of transport networks based on ONF Transport API using a cloud-native SDN controller based on micro-services. This demo provides insights on novel software implementations of transport network control technologies, such as container-based control architectures and standard interfaces. The proposed SDN controller components synchronize among them using gRPC protocol and a defined protocol buffer.

Index Terms—SDN controllers, architecture, micro-services

I. INTRODUCTION

Control of transport networks is typically performed through monolithic SDN controllers, which are typically based on closed-source implementations of different telecom vendors. In order to foster open innovation, while maintaining room for vendor added-value, network operators are supporting standard defined NorthBound Interfaces (NBI) for transport SDN controllers, being one of the examples ONF Transport API (T-API) [1]. This operator driven initiative is adding pressure to vendors to add support for such interfaces in their controllers, and to refine multi-domain architectures with multiple control domains using SDN orchestrators relying on the interfaces capabilities.

In the scope of a single control domain, transport SDN controllers tend to use large and complex code bases and frameworks that may not be optimized to specific use cases and do not scale when the underlying network infrastructure is large. This need to adapt to cloud-scale, suggests the development and adoption of novel software architectures that have demonstrated to cope with this clear and present challenge, such as a micro-service architecture and container-based orchestration systems (e.g. Kubernetes).

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) [2] 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 [3], the authors have presented the basis for such new generation of SDN controller, which is more flexible and scalable. The proposed novel cloud-native SDN controller is based on IETF Application-Based Network Operations [4], which we refer as µABNO (micro-ABNO).

In this demonstration, we propose to show the clear benefits of such approach, as well as we give the first opportunity to understand the architecture, observe the containerized microservices and experience with the proposed solution.

II. OVERVIEW

Figure 1 shows the proposed architecture, which consists on reformulating the concept of an SDN controller. The operator’s Operation Support Services and Business Support Services (OSS/BSS) are able to interact with the proposed cloud-native SDN controller, thus providing the necessary network dynamicity.

A Cloud Orchestrator (such as Kubernetes) is responsible for life-cycle management of the micro-services (i.e., health checks and resource allocation). The µABNO micro-services can be classified within three types: a) Database micro-service, 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 micro-service, which exposes µABNO NorthBound Interface (NBI) (e.g., ONF Transport API) as a...
RESTconf API and translates the request to internal protocol buffers; and c) gRPC micro-services, which use gRPC protocol and protocol buffers as basis for intercommunication.

Its architecture has been developed based on python container-based micro-services, 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 and topology components has several plugins in order to directly interact with network elements or dedicated SDN controllers.

T-API v2.1 [5] is used at several levels of the system. In particular, the photonic media layer augmentations are used to export the underlying topology, including the status of optical resources such as available optical spectrum and tunability constraints, and to allow a client (or orchestrator) to request for connectivity services, including the ability to constrain the service to a specific frequency slot. Regarding the SouthBound Interface (SBI), it is possible to use a diversity of REST or NETCONF based protocols for the SBI to configure the underlying transport devices.

Figure 2 provides an overview of the involved components and how they relate. Once a connectivity service request is received, the NBI translates this request into the proper protocol buffer and sends the request to connectivity micro-service. The connectivity micro-service first requests a path computation to path computation micro-service, which requires a topology retrieval. Once a feasible path is computed, virtual network topology manager (VNTM) micro-service is responsible for analyzing the need for multi-layer/multi-domain connections. VNTM generates the necessary connection requests towards the connection micro-service. The connection micro-service is responsible for requesting the necessary network element configuration (e.g., NETCONF, OpenFlow), or interacting with underlying SDN controllers.

III. INNOVATION

This demo is highly relevant to NetSoft, mostly in the scope of architectures and software defined control for metro and core networks.

The proposed µABNO cloud-native architecture has several key benefits, due to the orchestration of micro-services, which introduce improved network automation (GUI is shown in Figure 3): a) constant monitoring of micro-services and restart of them in case of failure, which provides self-healing mechanisms to SDN control; b) monitor micro-service resource consumption and scale the micro-service horizontally in case of overload (path computation is a resource consuming process which easily scales horizontally), thus providing auto-scaling to support cloud scale load requests; c) balance the load between replicated micro-services; and d) declarative network status description, which benefits network operators with network programmability and provides network rollbacks.

The proposed solution is to apply cloud-native application architecture, which provides the following benefits: a) Flexibility: Enabling the network to be quickly moved, provisioned and scaled to meet the ever changing needs of the infrastructures; b) Automation: Centralized, automated set-up of service chains, within both Layer 3 and Layer 4-7 services, can accelerate the roll out of services, such as firewalls, IPS, etc; c) Programmability: No need to manually reconfigure physical links or endpoint settings on the network; d) Multi-tenancy: The ability to support multiple silos and virtual networks running over the same physical links, allowing multiple networks (even with the same IP space) to share the same physical links; and e) Reliability: The ability to save and restore network topologies and configurations, via snapshotting, check pointing, and rollbacks, to allow for faster recovery from both bad configuration decisions and equipment failure in disaster recovery situations.

[6] provides a complete security and performance analysis to compare the two most widely used SDN controllers: ONOS (Open Network Operating System) and ODL (OpenDayLight).

Although a quantitative analysis is out of the scope of this paper, we can compare the µABNO implementation with current state-of-the-art SDN controllers in at least a few of the aspects: architecture, ease of deployment, programmability, request load balancing and auto-scale.

Both ONOS and ODL architecture is based on a monolithic core using OSGi bundles. The proposed µABNO architecture, based on micro-services, provides a higher degree of flexibility, at the cost of increasing complexity. Another of the benefits of this approach is versatility, as micro-services allow for the use of different technologies and languages. Component auto-scale and load balancing is a useful cloud-native feature in order to increase single modules capacity for responding to high request loads.

There are several withdraws of proposed µABNO architec-
ture. Because the components are distributed, global testing is more complicated. Moreover, there can appear problems of cost (due to the need for scalable cloud resources), efficiency (overhead might be lost in micro-service communication) as well as longer response times due to inter-service communication (it is evaluated in Section IV.

IV. DEMONSTRATION AND RELEVANCE

The proposed µABNO architecture has been developed based on python container-based micro-services, 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.

Figure 4: Micro-services for SDN controller

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 micro-services running on the cluster. A loadgenerator micro-service has also been developed in order to stress the proposed architecture and obtain significant results of cloud-native SDN controller.

By relying on T-API models, it covers open and intent-based APIs for functions such as topology discovery, service management, path calculation and provisioning as well as the use of NETCONF/RESTCONF protocols and Yang modeling language. Finally, a functional architecture is detailed in which the different functional components of the µABNO element are to be deployed in terms of a microservices-based architecture. The demo should be of interest to researchers, developers, industry actors working on SDN for transport networks, and in particular, covering the use of microservices, Kubernetes and container-management platforms. Moreover, the introduction of microservices allows the analysis of novel use cases, such as auto-healing and auto-scale mechanisms for SDN orchestrator, which allow the management of cloud-scale number of connectivity intents.

Figure 2 displays the average aggregated latency per component. To process a create request for connectivity service might take 172ms, from these 94ms correspond to path computation, 60ms to VNTM, or 5ms to create the connection.

In Figure 4, the deployed micro-services are shown, including their internal IP addresses and ports. Although each micro-service uses a TAPI-based protocol buffer for message exchange, all micro-services implement a health-monitoring protocol, which allows to re-start a micro-service in case of failure. Figure 5 shows the message exchange in gRPC between each developed micro-service in order to trigger the instantiation of a connectivity service. In Figure 2 all connections between components are depicted, as well as the latency of those connections.

![Figure 5: Wireshark capture of micro-service message exchange when triggering a connectivity request](image)

V. CONCLUSION AND FUTURE WORK

We have presented a novel and radical SDN controller for transport networks. The proposed SDN controller architecture is based on micro-service components, which provide the required level of adaptability, self-healing and auto-scale that enables cloud-scale traffic load management.

The proposed architecture has been implemented in a cloud-based solution based on python micro-services that interact among them using gRPC and protocol buffers.

The presented SDN controller is a prototype that will be applied to a laboratory trial before end of 2020. NetSoft attendees will benefit from the insights and internal structure of the presented demonstration.

ACKNOWLEDGMENT

This work is part of MINECO project AURORAS (RTI2018-099178).

REFERENCES

[1] 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). IEEE, 2019, pp. 1–7.

[2] R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, S. Beppu, D. Soma, S. Sumita, T. Tsuritani, I. Morita, and R. Muñoz, “Grpc-based sdn control and telemetry for soft-failure detection of spectral/spacial superchannels,” in in proceedings of ECOC, 2019.

[3] R. Vilalta, J-L. de-la Cruz, A. Mayoral, V. Lopez, R. Martinez, R. Casellas, and R. Muñoz, “ubano: A cloud-native architecture for optical sdn controllers,” in Optical Fiber Communication Conference. Optical Society of America, 2020.

[4] A. Aguado, V. López, J. Marhuenda, Ó. G. de Dios, and J. P. Fernández-Palacios, “Abno: A feasible sdn approach for multivendor ip and optical networks,” Journal of Optical Communications and Networking, vol. 7, no. 2, pp. A356–A362, 2015.

[5] C. Janz, L. Ong, K. Sethuraman, and V. Shukla, “Emerging transport sdn architecture and use cases,” IEEE Communications Magazine, vol. 54, no. 10, pp. 116–121, 2016.

[6] S. Secci, A. Diamanti, J. Sanchez, M. Vilchez, M. T. Bah, P. Vizarreta, C. Machuca, S. Scott-Hayward, and D. Smith, “Security and performance comparison of onos and odl controllers,” 2019.