Demonstration of Zero-touch Device and L3-VPN Service Management using the TeraFlow Cloud-native SDN Controller

Demonstration of Zero-touch Device and L3-VPN Service Management using the TeraFlow Cloud-native SDN Controller

Ll. Gifre[1], C. Natalino[2], S. Gonzalez-Diaz[3], F. Soldatos[4], S. Barguil[5], C. Aslanoglou[4], F. J. Moreno-Muro[3], A. N. Quispe Cornelio[1], L. Cepeda[5], R. Martinez[1], C. Manso[1], V. Apostolopoulos[4], S. Petteri Valiviita[6], O. Gonzalez de Dios[5], J. Rodriguez[6], R. Casellas[1], P. Monti[2], G. P. Katsikas[4], R. Muñoz[1], and R. Vilalta[1]

[1] Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Castelldefels (Barcelona), Spain;
[2] Electrical Engineering Department, Chalmers University of Technology, Gothenburg, Sweden;
[3] ATOS, Madrid, Spain; [4] UBITECH, Athens, Greece; [5] Telefónica I+D, Madrid, Spain;
[6] Infinera, Espoo, Finland.

lluis.gifre@cttc.es

Abstract: We demonstrate zero-touch device bootstrapping, monitoring, and L3-VPN service management using the novel TeraFlow OS SDN controller prototype. TeraFlow aims at producing a cloud-native carrier-grade SDN controller offering scalability, extensibility, high-performance, and high-availability features.

1. Overview

Software-Defined Network (SDN) controllers are the *de-facto* solution to manage the access, transport, and cloud network segments. However, due to their monolithic software nature current SDN controllers are inherently limited in terms of scalability, flexibility, availability, security, and dynamicity. On the other hand, the stringent requirements of 5G and Beyond (B5G) networks brings forth the need to overcome these limitations. For this reason, B5G-ready SDN controllers need to operate unimpaired regardless of the network load/size, the disaster conditions, or even in the presence of potential human errors.

Cloud-native applications are a potential remedy to the limitations of monolithic software [1, 2]. They are designed as *microservices* (i.e., small, loosely-coupled, computation services) that (via network messages) cooperate to realize the application's mission. This design enables easier replication of the services allowing greater dynamicity, increased scalability, better load balancing, and high-availability features. Additionally, they can be deployed in a geographically distributed fashion, providing appropriate levels of redundancy at the infrastructure and the software layers to prevent possible service interruptions. To this end, two popular industry-grade SDN controllers todate, i.e., ONOS and OpenDayLight, are being re-designed to follow a microservice-based architecture.

Another important trend gaining traction in B5G scenarios is Network Functions Virtualization (NFV) that aims to move network functions (e.g., firewalls, routers, baseband units, etc.) away from proprietary hardware appliances into decoupled and softwarized components running over virtualized infrastructures. Virtualized Network Functions (VNFs) can then be deployed throughout the network where compute resources are available. Then, Multi-Access Edge Computing (MEC) proposes to provide compute resources at the network edge to deploy VNFs in close proximity to the users and improve the user experience. In this paradigm, SDN becomes an enabler for NFV and MEC as it configures the underlying infrastructure to interconnect the VNFs'.

Zero-Touch Automation aims at completely automating the network management process with unquestionable benefits in terms of *(i)* reduced deployment and operational time, *(ii)* increased traceability, and *(iii)* a better consistency and scalability of the network operations given the reduced human intervention, while keeping the total costs of ownership at acceptable levels.

The TeraFlow Operating System (OS), currently under development in the EC H2020 TeraFlow project [3, 4], is one of the first open-source, microservice-based, cloud-native, and carrier-grade SDN controller. It aims at integrating with current NFV and MEC frameworks and automating the management of B5G networks guaranteeing high levels of reliability, redundancy, and security. The objective of this demonstration is to showcase how TeraFlow OS deals with *(i)* zero-touch device bootstrapping, *(ii)* L3 Virtual Private Network (VPN) (L3-VPN) connectivity services' management, and *(iii)* monitoring of the network.

2. Innovation

This demonstration will showcase a proof-of-concept of a B5G-ready cloud-native SDN controller that integrates with NFV/MEC frameworks, and real and emulated equipment, all using standardized interfaces. The capabilities of the TeraFlow OS, and the integration of its components, and with external frameworks, will be validated by

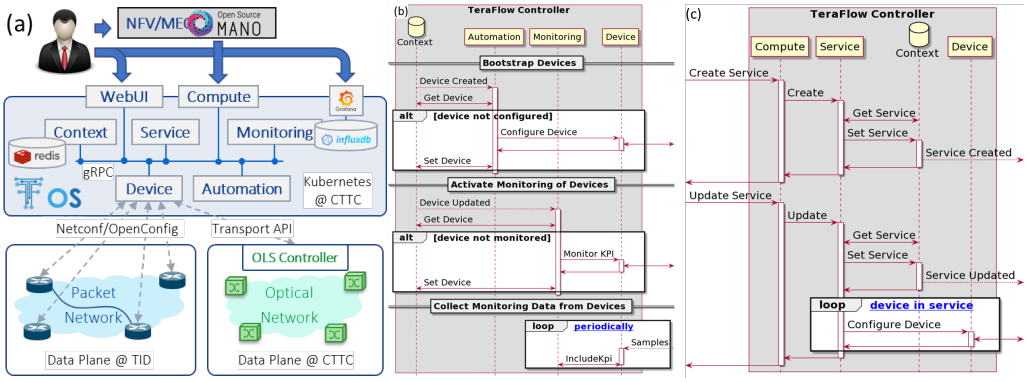

Fig. 1. *(a)* Architecture with the TeraFlow OS components (left), *(b)* the workflow to bootstrap and monitor new devices (center), and *(c)* the workflow to manage the connectivity services (right).

means of two workflows: zero-touch device bootstrapping and monitoring, and Point-to-Point (P2P) L3-VPN service management.

3. Demo Content and Implementation

In this section, we present the TeraFlow OS architecture and workflows relevant for this demonstration.

3.1. Architecture

Fig. 1a illustrates the architecture of the demonstration including the ETSI Open Source Management and Orchestration (MANO) (OSM) [5] to issue L3-VPN connectivity service requests, the TeraFlow OS components to handle the requests, and the data plane. The TeraFlow OS components used in this demonstration are:

(a) Device interacts with the network equipment by means of the South-Bound Interfaces (SBIs) implemented as pluggable drivers. A Driver Application Programming Interface (API) has been defined to facilitate the addition of new network protocols and data models. In this demonstration, we deployed the NetConf [6]/OpenConfig [7] Driver API for the packet routers, and the Transport API (TAPI) [8] for the Optical Line System (OLS);

(b) Context stores the network configuration (e.g., topologies, devices, links, services) managed by the TeraFlow OS in a No-SQL database to optimize concurrent access. Internally, it implements a Database API enabling to switch between different backends; in this demonstration we use Redis. It provides publish-subscribe mechanisms over Google Remote Procedure Call (gRPC) to broadcast change events to other TeraFlow OS components.

(c) Service manages the life-cycle of the different connectivity services handled by the TeraFlow OS. Given the variety of service types that could be demanded, it implements a Handler API enabling network operators to integrate the desired service types, and fine tune their behavior;

(d) Automation implements a number of Event-Condition-Action (ECA) loops [9] defining the automation procedures in the network. These control loops deal with automation tasks such as bootstrapping of new devices, configuration of interfaces and/or forwarding tables, etc. They are triggered by relevant events (e.g., addition of a device), when specific conditions are met (e.g., not configured), and they apply a set of actions (e.g., bootstrap the device) in response to these events.

(e) Monitoring manages the different metrics (Key Performance Indicators (KPIs) in TeraFlow OS terminology) configured for the network equipment and services, stores monitoring data related to these KPIs and provides means for other components to access the collected data. Internally, Monitoring component relies on an InfluxDB database to store the monitoring data as time series, exploiting its powerful querying and aggregation mechanisms for retrieving the collected data.

(f) Compute provides a Representational State Transfer API (REST-API)–based North-Bound Interface (NBI) to external systems, such as NFV and MEC frameworks. Compute translates their requests into internal TeraFlow OS requests. In this demonstration, support for OSM L3-VPN has been implemented;

(g) Web-based User Interface (WebUI) uses the gRPC-based interfaces made available by the TeraFlow OS components to inspect the network state, as well as to issue operational requests to the TeraFlow OS components. WebUI takes advantage of the InfluxDB querying capabilities in the monitoring component to feed Grafana dashboards, thus enabling to visualize the network state and relevant KPIs.

The TeraFlow OS components are implemented as microservices in Python (except Automation that is implemented in Java); they are deployed on top of a Kubernetes-based [10] environment, and rely on well-defined TeraFlow-specific gRPC-based messages and services for their internal communications. The Kubernetes envi-

ronment is deployed at CTTC premises in Castelldefels (Spain). The DataPlane is distributed in two geographical locations, that are interconnected by means of a secured VPN connection. The emulated optical layer, located at CTTC premises, is controlled by means of an OLS Controller and exposes TAPI at its NBI. The packet layer, located at TID premises in Madrid (Spain), consists of 2 Infinera DRX30 routers controlled by means of Netconf/OpenConfig for both the configuration and the monitoring of the devices.

3.2. Workflows

This demonstration will focus on the two workflows depicted in Figs. 1b and 1c.

(i) The *zero-touch device bootstrapping and monitoring* workflow (Fig. 1b) is triggered by the addition of a new device in the TeraFlow OS, for instance, through the WebUI. The *WebUI* adds the new device through *Device* which triggers a connection to the physical device and the retrieval and storage of its current inventory and configuration in the *Context* database. Such action, triggers the distribution of a "Device Created". When *Automation* receives the event, it retrieves the information and configuration of the new device from *Context*. If the device is not configured, it is bootstrapped by *(i)* retrieving from *Device* the initial configuration template the driver defines for this device, *(ii)* populates the template with the appropriate values, *(iii)* configures the device through *Device*, and *(iv)* updates in *Context* database the configuration and new state of the device. The update on the device triggers the distribution of a "Device Updated" event. When *Monitoring* receives this event, if the device is enabled but not being monitored, the former creates a set of KPIs for this device, and activates the monitoring of these KPIs through *Device*. At that point, the samples coming periodically from the device are issued to *Monitoring* that stores and makes them available for the other components.

(ii) The *P2P L3-VPN service management* workflow (Fig. 1c) is triggered when OSM requests the creation of a new L3-VPN service. Such request has two phases; first, a new empty service is created to obtain a service identifier, and then, the endpoints are added to the service. When *Compute* receives the service creation request, it forwards the request to *Service*, that completes the missing required fields with default values, creates the service in the *Context* database, and returns the service identifier to OSM. Upon *Compute* receives the request to add the endpoints to the service, it issues a service update request towards *Service* that identifies the devices owning the endpoints to be connected, identifies the device drivers they support, and chooses the appropriate service handler for the service. In this demonstration, it first chooses and instantiates the L3-VPN Network YANG Model (L3NM)-OpenConfig (OC) (L3NM-OC) service handler to configure a L3-VPN using Netconf/OpenConfig, and then it forwards the service request to that service handler. Next, the service handler creates the configuration rules for each involved device and configures them through *Device*. Finally, it returns a confirmation to OSM.

4. OFC Relevance

This demonstration will be the first to showcase to the OFC audience a proof-of-concept of a cloud-native SDN controller. In addition, it aims at bringing awareness to both network operators and equipment vendors of the features and benefits a cloud-native microservice-based SDN controller can bring. We expect it will motivate the interactions between parties at OFC to further discuss the introduction, adoption, optimization, and standardization of such novel technologies for future generation SDN controllers.

Acknowledgements: This work has been supported by the EC H2020 TeraFlow (101015857), and Spanish AURORAS (RTI2018-099178-I00) projects.

References

1. Q. Pham Van, *et. al*, "Demonstration of Container-Based Microservices SDN Control platform for Open Optical Networks," in Proc. OFC, p. M3Z.5, 2019.
2. R. Vilalta, *et. al*, "uABNO: A Cloud-Native Architecture for Optical SDN Controllers," in Proc. OFC, p. T3J.4, 2020.
3. "TeraFlow - Secured autonomic traffic management for a Tera of SDN Flows", Nov. 2021, https://www.teraflow-h2020.eu/.
4. "GitLab - TeraFlow SDN Controller", Nov. 2021, https://gitlab.com/teraflow-h2020/controller.
5. "ETSI Open Source MANO (OSM)", Nov. 2021, https://osm.etsi.org/.
6. R. Enns, *et. al*, "Network Configuration Protocol (NETCONF)," IETF RFC6241, June 2011.
7. "OpenConfig", Nov. 2021, https://https://www.openconfig.net/.
8. A. Mayoral, *et. al*, "TAPI v2.1.3 Reference Implementation Agreement v1.0," ONF TR-547, Jul. 2020.
9. M. Boucadair, *et. al*,"Framework for Use of ECA (Event Condition Action) in Network Self Management," IETF draft-bwd-netmod-eca-framework-00, 2020.
10. "Kubernetes", Nov. 2021, https://kubernetes.io/.