Scalable for Cloud-native Transport SDN Controller Using GNPy and Machine Learning techniques for QoT estimation

Carlos Manso[1], **Ricard Vilalta**[1], **Raul Muñoz**[1], **Ramon Casellas**[1], **Ricardo Martínez**[1]

[1] *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA). Castelldefels (Spain)*

carlos.manso@cttc.es

Abstract: This demo shows a cloud-native SDN controller that can estimate end-to-end QoT using both analytical (GNPy) and machine learning algorithms on WDM systems. This transport SDN controller is able to scale horizontally using custom metrics. © 2021 The Author(s)

1. Introduction

In optical networks, it is key to ensure a good Quality of Transmission (QoT) before the provisioning to avoid the degradation of the optical signal during the transmission below a given threshold. The selection of the path between the endpoints of the service is critical to maintain the performance during the duration of the connection. With all the physical impairments of the optical fibers and transmission equipment, it is hard to predict what the Bit-Error Rate (BER) of the connection will be. Traditionally, it has been an election between precise and approximate methods. Precise methods can shape physical impairments with a high accuracy but their use of complex mathematical algorithms and formulas makes the computation cost very high, such as the Split Step Fourier (SSF) or analytical models. On the other hand, approximate methods, such as the Gaussian Noise model (GN), although not as precise, they make good approximations but with a fraction of the computational cost that require analytical models [1].

The GNPy (Gaussian Noise model in Python) is an open-source software library built for route planning and optimization of optical networks based on the GN model [2]. It is being developed by the Telecom Infra Project (TIP), an engineering-focused initiative driven by operators that at the same time collaborates with suppliers, developers and startups to ease the disaggregation of the traditional optical networks. To achieve this, it is critical to accurately plan and predict the performance of disaggregated optical networks based on the simulation of the optical parameters, that is what GNPy is mainly developed for.

Nonetheless, in the last few years, Machine Learning (ML) techniques have also been studied for QoT prediction. They offer similar computational cost (comparing the inference phase) to the approximate methods, while removing the need to model some components that are not accurate, adding some impairments that are too slow to compute, and the dependence on certain parameters on the models [3]. The main problem with ML techniques, is that they need previously obtained data to be fed to the ML model, which sometimes are difficult to obtain.

ML techniques and complex path computation models have been typically deployed in dedicated hardware and software apart from transport SDN controller, in order to ensure enough resources for these operations. Current novel SDN controller architectures based on microservices allow the introduction of these features inside the SDN controller. For this, the infrastructure running the transport SDN controller should allow scable resource allocation based on horizontal scaling of the computation resources.

In this demonstration, we present an extended cloud-native Software-Defined Networking (SDN) controller known as uABNO [4] based on microservices. This demo features a new path computation module able to use both the GN model based on GNPy, and ML model for QoT prediction. This two models are also built as microservices that communicate with the path computation microservice, but unlike the other microservices of the SDN controller, they are automatically scaled under different user-defined metrics to avoid excessive queuing of requests. They are deployed using Docker, the monitoring of the metrics is performed by Prometheus, while the orchestration and auto-scaling will be carried out by Kubernetes.

2. Overview

In Fig.1 (left) the SDN controller architecture is shown. It contains a cloud controller (based in Kubernetes), as well as a monitoring system and time series data base (based on Prometheus), a metrics server that allows multiple metrics export from monitoring system to cloud controller and a Horizontal Pod AutoScaler (HPA). It also depicts the different types of microservices:

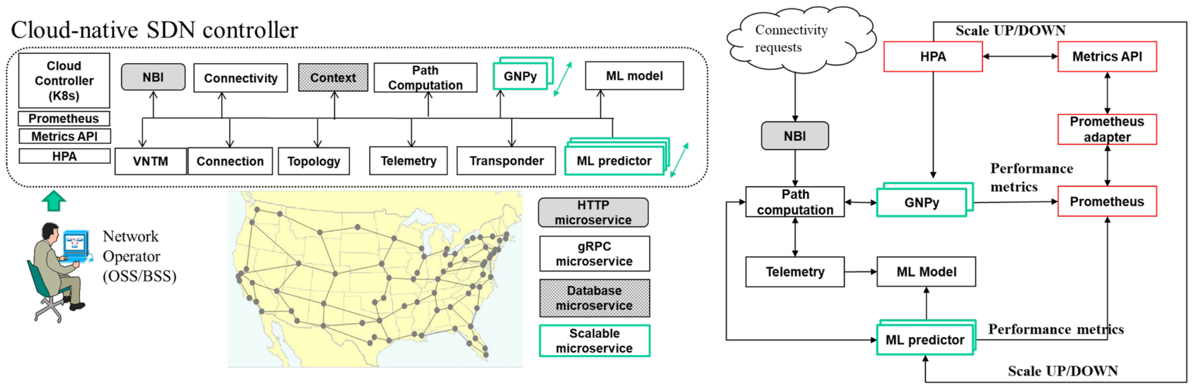


Fig. 1: Architecture and topology (left) and autoscaling workflow (right)

- HTTP microservices: Are the microservices that receive HTTP requests from the Operation Support Services and Business Support Services (OSS/BSS) or another type of clients. In our case, the only HTTP microservice is the NorthBound Interface (NBI) module.
- gRPC microservices: They carry out different essential functionalities for other modules and are the most numerous microservices in the uABNO.
- Database microservices: They store the state of the topology, connections or devices that are used (MongoDB is used in our case).
- Scalable microservices: Are the key concept of this paper, auto-scaling microservices by means of replicating the containers by monitoring different metrics. On this demonstration the GNPy and the ML Predictor will be the two scalable microservices on the SDN controller.

Fig.1 (right) depicts the main workflow for the provisioning of a new connection. The connectivity service request enters the SDN controller through the NBI and ask the Path Computation for a path between the endpoints. The Path Computation calculates the shortest path with available optical resources, and at this point, it has two different choices: a) ML Predictor, and b) GNPy.

a) If the ML Model is not trained yet, the Path Computation asks the GNPy to calculate the Optical Signal-to-Noise Ratio (OSNR) for the previously computed path, and returns it to the Path Computation, where it is converted to BER. If the BER received is acceptable, the provisioning of the connection continues, if it is not, the next shortest path is calculated until an acceptable route in terms of BER is found (or abortion if there are no more left). Finally, the Path Computation sends the connection data and the associated BER to train the ML model.

b) The other option, if the ML model is already trained, is to predict the BER from the data associated to the connection and again calculate another path if the BER is not acceptable.

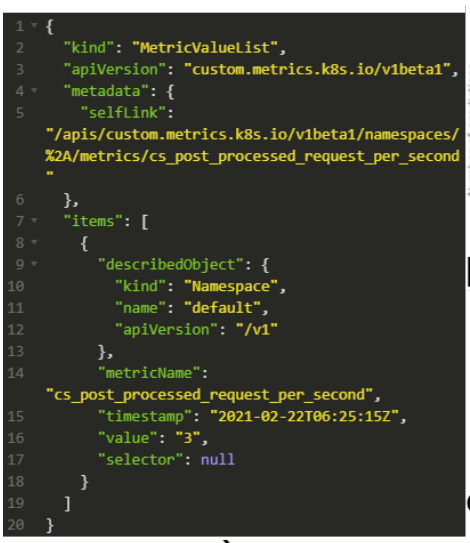
In order to monitorize the multiple microservice status, a Prometheus server is introduced, which is an open-source monitoring system. Kubernetes on its own it is only capable to retrieve simple monitoring information from the microservices orchestrated, i.e. CPU and memory, and Prometheus enables the system to monitor custom defined metrics, such as number of connections. Then, Prometheus, by means of the Prometheus adapter, feed the information to the Kubernetes' Metrics Application Programming Interface (API). HPA is the Kubernetes' service that taking into account the metrics exposed in the Metrics API, scales up the pods (the most basic Kubernetes unit) if the container is getting out of resources or down if the pods have idle resources, allowing for an optimal utilization of them.

3. Innovation

Different innovations are shown in this demonstration: the usage of a metrics APi to scale the resources for the transport SDN controller, the introduction of multiple microservices for path computation (based on GN and ML techniques), and the introduction of GNPy as a microservice, allowing and easing its scalability.

Figure 2.a shows the capacity of our microservice-based transport SDN controller to export internal metrics using Metrics API. It might include standard metrics, such as CPU, RAM and network usage, as well as user-defined metrics. These metrics can be later used for the HPA.

The main aspect found in this demonstration is the scalability of microservices using HPA. Both the ML Predictor and the GNPy microservices are scalable microservices. This will ensure that the requests can be served no matter the load and enable the scalability for a cloud-level number of requests. The use of GNPy enables fast



```json
{
  "kind": "MetricValueList",
  "apiVersion": "custom.metrics.k8s.io/v1beta1",
  "metadata": {
    "selfLink": "/apis/custom.metrics.k8s.io/v1beta1/namespaces/%2A/metrics/cs_post_processed_request_per_second"
  },
  "items": [
    {
      "describedObject": {
        "kind": "Namespace",
        "name": "default",
        "apiVersion": "/v1"
      },
      "metricName": "cs_post_processed_request_per_second",
      "timestamp": "2021-02-22T06:25:15Z",
      "value": "3",
      "selector": null
    }
  ]
}
```

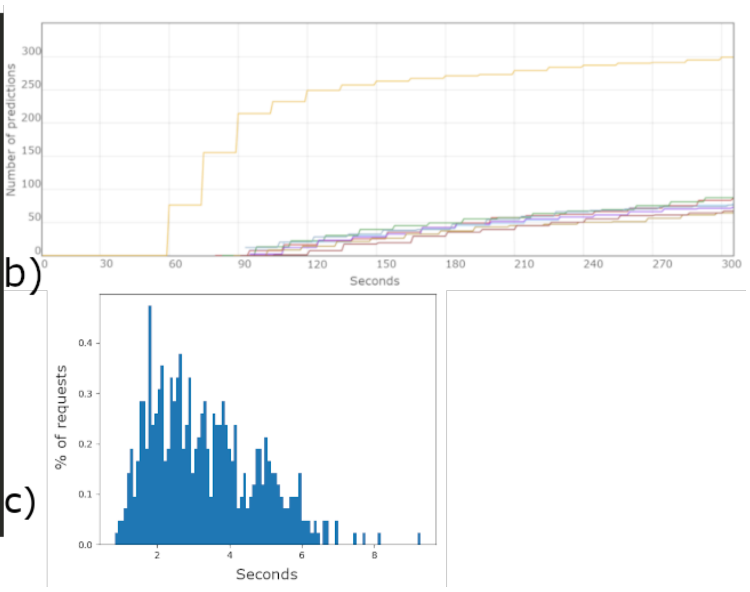


Fig. 2: Experimental results. a) Example of a custom metric (requests per second) retrieved from the Metrics API in JSON. b) Example of the scalability in terms of spawn of replicas. c) Second elapsed to compute path requests and QoT estimation on the CORONET CONUS topology using GNPy

and easy end-to-end QoT prediction for disaggregated optical networks, still, the SDN controller, presents enough flexibility to chose between different QoT prediction methods. Figure 2.b shows how several instances of ML predictor are deployed as a function of the request load.

Another important aspect to be demonstrated is the deployment of GNPy as a microservice. This allows to scale it depending on the work load of path computation requests, or even the mean time to process a request (as the increase of this mean time might indicate a lack of resources for path computation). In Figure 2.c, the histogram of path computation delay for GNPy using CORONET topology is shown.

4. Relevance

The relevance of this demonstration for the OFC would be of interest for the N3 (Architectures and Software-defined Control for Metro and Core Networks) and N2 (Optical Networking for Data Center and Computing Applications) subcommittees. The demo showcases an cloud-native SDN controller, capable of serving a very high number of requests because of the scalability features it supports. The GNPy microservice is an example of this, which at the same time, allows for accurate and fast QoT prediction for disaggregated optical networks. The demonstrated transport SDN controller also relies on standard protocols, as the NBI works by means of the Open Networking Foundation (ONF) Transport API (TAPI), a standard NBI interface for transport SDN controllers. The demonstration can be of interest to the industry and researchers interested in optical networking control and management in general, and anyone interested in centralized cloud SDN controllers worried about scaling possibilities in particular.

Acknowledgments

Work partially supported by the EC H2020 TeraFlow (101015857) and Spanish AURORAS (RTI2018-099178-I00) projects.

References

1. D. Azzimonti, C. Rottondi, and M. Tornatore, “Using active learning to decrease probes for qot estimation in optical networks,” in *OFC,* (2019).
2. A. Ferrari, M. Filer, K. Balasubramanian, Y. Yin, E. Le Rouzic, J. Kundrát, G. Grammel, G. Galimberti, and V. Curri, “Gnpy: an open source application for physical layer aware open optical networks,” J. Opt. Commun. Netw. **12**, C31–C40 (2020).
3. Y. Pointurier, “Machine learning techniques for quality of transmission estimation in optical networks,” J. Opt. Commun. Netw. **13**, B60–B71 (2021).
4. R. Vilalta *et al.*, “uabno: A cloud-native architecture for optical sdn controllers,” in *OFC,* (2020).