Slice Isolation for 5G Transport Networks

Chia-Yu Chang*, Manuel A. Jiménez**, Molka Gharbaoui‡††, Javier Sacido†,
Fabio Ubaldi§, Chrysa Papagianni*‖, Aitor Zabala†, Luca Valcarenghi‡, Davide Scano‡,
Konstantin Tomakh¶, Alessio Giorgetti‡‡‡, Andrea Boddi§, Koen De Schepper*

*Nokia Bell Labs; †Telcaria Ideas S.L.; ‡Scuola Superiore Sant'Anna; §Ericsson Research; ¶MIRANTIS;
‖University of Amsterdam; **IMDEA Networks Institute; ††CNIT; ‡‡IEIIT-CNR;

***Abstract*—Network slicing plays a key role in the 5G ecosystem for vertical industries to introduce new services. However, one widely-recognized challenge of network slicing is to provide traffic isolation and concurrently satisfy diverse performance requirements, e.g., bandwidth and latency. In this work, we showcase the capability to retain these two goals at the same time, via extending the 5Growth baseline architecture and designing a new data-plane pipeline, i.e., virtual queue, over the P4 switch. To demonstrate the effectiveness of our approach, a proof-of-concept is presented serving different service requests over a mixed data path, including P4 switches and Open vSwitches (OvSs).**

I. Introduction

The fifth-generation (5G) marks a paradigm shift beyond the telecommunication industry, paving the way to provide ubiquitous connectivity and value-added services to vertical industries. Therefore, a comprehensive set of new use cases comes into the picture with diverse service requirements, ranging from substantial bandwidth needed by ultra HD streaming to stringent low-latency urged by haptic interaction. To this end, one key requirement for 5G is to provide verticals with networking environment tailored to their needs.

In this manner, the *network slicing* notion arises as one critical feature to serve the diverse 5G ecosystem. In practice, each network slice can be accounted as a part of the infrastructure, comprising virtualized resources, virtual/physical network functions, and network services, configured to support some service's Key Performance Indicators (KPIs). Moreover, network slicing in 5G benefits from both Network Functions Virtualization (NFV) and Software Defined Networking (SDN) techniques for fast service deployment and simplified lifecycle management, exposing a standard interface to each tenant for deploying and managing network services.

However, one basic requirement for network slice instances running over the same physical infrastructure is *slice isolation*; that is they need to be operated independently without one part being affected by the internal operations or performance of another. Several levels of isolation between network slices are introduced in [1]. Among them, traffic and bandwidth isolation are pivotal for network applications. The former aims to ensure that the data flow of one slice will not move to another slice sharing the same infrastructure, while the latter indicates that each slice should not exceed, unless explicitly instructed to, the total amount of bandwidth allocated to it. In general, network slice isolation dimensions and solutions are elaborated in [2] for different network domains. Focusing on *performance isolation*, the requirement to meet service KPIs regardless of congestion level for the rest of the network slices, we provide a programmable solution ensuring bandwidth and delay guarantees, as well as *traffic isolation*, and showcase it for the transport network domain.

978-1-6654-0522-5/21/$31.00 ©2021 IEEE

The remaining article is structured as follows. A system description is in Sec. II, with the introduction to our extensions to legacy 5Growth baseline platform and data-plane infrastructure to facilitate performance isolation. We then validate the solution's effectiveness over a proof-of-concept presented in Sec. III. Finally, some concluding remarks are in Sec. IV.

II. System Description

A. 5Growth Architecture Extensions for Performance Isolation

Our work is based on the baseline 5Growth architecture, consisting of three main building blocks: (a) Vertical Slicer (5Gr-VS), (b) Service Orchestrator (5Gr-SO), and (c) Resource Layer (5Gr-RL), from top to bottom. Due to space limitation, we encourage readers to refer to our prior work in [3] for details. Among them, our focus is on the 5Gr-RL, interacting with the underlying data-plane infrastructure to provide different resources to network slices. In addition, two specific plugins for 5Gr-RL are extended to enable network slice isolation: WAN Infrastructure Manager (WIM) plugin that enables control of Wide Area Network (WAN) domain and a Virtual Infrastructure Manager (VIM) plugin for governing the available computing resources in Data Center (DC) domain. We highlight our particular extensions as follows:

*1) **5Gr-RL extensions**:* The 5Gr-RL release [4] is extended to support slice isolation. Specifically, a smart resource orchestration algorithm is triggered for each slice, upon receiving a resource allocation request, mapping the abstracted requested resource to a Quality of Service (QoS) policy characterizing the network slice. For instance, the policy can include parameters like minimum/maximum throughput and maximum delay. Moreover, it sets up the corresponding alarms using the monitoring system. The 5Gr-RL southbound interface is also extended to communicate parameters to the WIM plugin that is responsible for configuring transport infrastructures.

*2) **5Gr WIM plugin**:* The WIM plugin enables the interaction between the 5Gr-RL southbound interface and the Open Network Operating System (ONOS) SDN controller. Specifically, upon request (i) it exposes to the 5Gr-RL information regarding the status of the virtualized network resources characterizing the SDN network, e.g., virtual links, available bandwidth and (ii) it allows for requesting the allocation of virtualized network resources to deploy a network service.

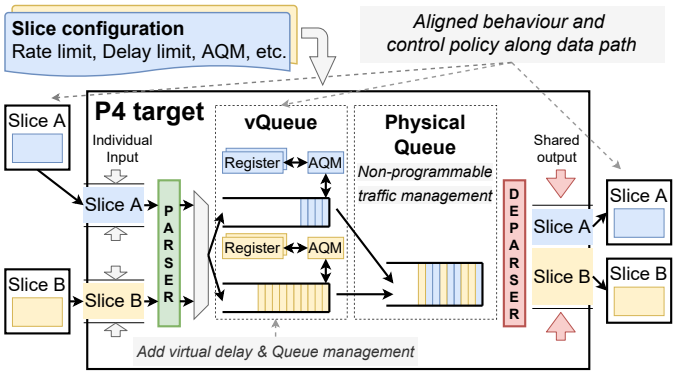

Fig. 1: Virtual queue implementation over P4 pipeline

Our extension to WIM plugin [5] allows to handle the requests that aim to set up network slices with specific QoS characteristics and also to enforce the performance isolation. Therefore, new parameters characterizing the slices are added, such as maximum/minimum bandwidth, maximum burst size, and Active Queue Management (AQM) parameters for delay guarantee, etc. After parameter extraction from the request, the WIN plugin interacts with the SDN controller to set up the slice and enforce QoS guarantees.

*3) **5Gr VIM plugin**:* The 5Gr-RL manages the VIM resources (e.g., computing and communication resources) through the VIM plugin. Particularly, when Kubernetes is used as the VIM, the corresponding Kubernetes plugin is utilized for the 5Gr-RL. The VIM plugin first receives the 5Gr-RL's compute resource creation requests, translates it, and then sends it to Kubernetes to create pods inside the Kubernetes cluster. Note that the initial version of the Kubernetes plugin does not support VLAN-based networks, while different Container Network Interfaces (CNI) must be used by Kubernetes to support different network technologies. Hence, Multus [6] CNI is used to connect the pods to the transport network's external WIM. Multus is a CNI plugin for Kubernetes that enables attaching multiple network interfaces to pods, and it can create a VLAN-based network that connects a physical port to the pod. Thus, the VIM plugin is extended accordingly to provide the interface for managing the Multus CNI.

B. Infrastructure Extensions for Performance Isolation

To support the performance isolation, several extensions are realized in both control-plane (ONOS SDN controller) and data-plane (Open vSwitch [OvS] and P4 switch).

Firstly, to enforce *traffic isolation*, we employ the Virtual Private LAN Service (VPLS), deploying L2 multipoint-to-multipoint connections over the shared transport network. The *VPLS* native ONOS application is employed towards that end. In addition, to be cross-compatible with data-plane infrastructure beyond OvS, a re-engineering of the P4 data-plane is executed to be applicable for P4 switch.

To meet throughput and delay guarantees, we introduce the *virtual queues* in the P4 pipeline, as shown in Fig. 1. In short, a virtual queue models the sojourn time of the queue, as if the packets arrive at the real queue serving by a link with the capacity lower than the actual link capacity. Via using virtual

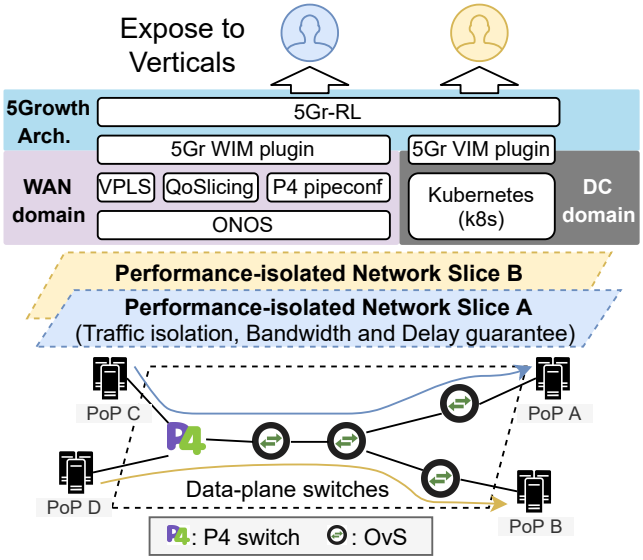

Fig. 2: System overview

queues to perform per-slice policing and leveraging P4's real queuing latency monitoring capability, we can meet the delay requirement of each slice. In contrast, basic traffic metering is used via the OpenFlow protocol over OvSs for bandwidth guarantee. Such meters can only manage bandwidth and burst limits by means of color codes (e.g., two rate Three Color Marker) when thresholds are exceeded.

In addition, we deploy ONOS to control and coordinate SDN protocols for both P4 switch and OvS. However, ONOS nowadays only provides traffic isolation across slices, without the capability of bandwidth and delay guarantees. Therefore, we introduce the *QoSlicing* as an application managing both meters and virtual queues across slices and exposing a unified interface towards the upper layers. Such application utilizes the developed *P4-pipeconf* application as the interpreter to interact with underlying P4 pipeline, e.g., to obtain virtual queues' parameters. Finally, as shown in Fig. 2, the performance-isolated network slices are exposed to verticals across Points of Presences (PoPs), leveraging the ONOS SDN controller (with three aforesaid applications) in WAN domain, Kubernetns cluster in DC domain, and 5Gr-RL with two plugins.

III. Proof-of-Concept

In the demonstration, we investigate four traffic metrics with regards to evaluating the performance of network slices. First is the end-to-end latency by measuring the round-trip time between two end-points, and the second is the application-layer throughput by counting successfully received data payload. Moreover, to capture the latency in the data-plane infrastructure, we also measure two types of delay: virtual delay and real delay. The former is the latency encountered in the virtual queue (cf. Fig. 1), and the latter is the experienced delay of the actual physical queue in the pipeline. Note that these two types of delay show their importance under different conditions. A non-zero real delay represents that the physical queue is built up, while the virtual delay can limit throughput when link capacity is not saturated.

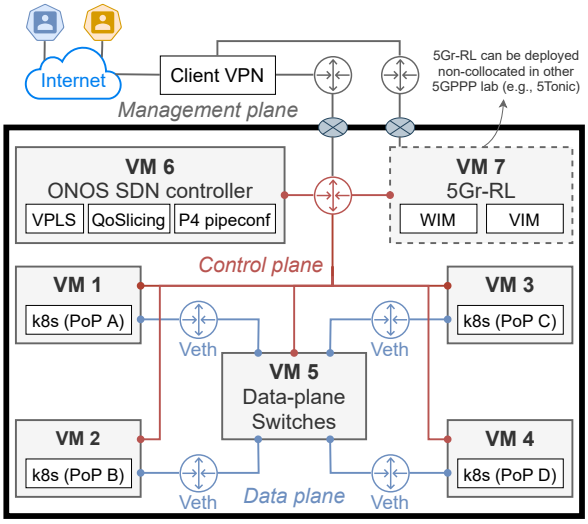

Fig. 3: Proof-of-concept setup

To demonstrate the system in Fig. 2, one server hosting seven virtual machines (VMs) is installed, with their interconnections and functionalities shown in Fig. 3. There are four VMs acting as individual host to transmit/receive traffic via the correspondent PoP, while all data plane switches (OvS and BMv2) are deployed in VM5 using mininet. Further, VM6 is served as the ONOS SDN controller with three aforementioned applications to control underlying data plane switches, and the 5Gr-RL is on VM7 with both VIM and WIM plugins. Finally, four presented phases in our demonstration are explained as follows, and the full video can be found in [7]:

Phase I: We first set the link capacity to 72 Mbps and instantiate one network slice A with a single 100 Mbps UDP traffic flowing from PoP C to PoP A. To show the vanilla behavior of a network slice, neither bandwidth nor delay guarantee is applied. We can observe a very large end-to-end latency (more than 2 second) together with the saturated throughput. Moreover, a large real delay and almost zero virtual delay are observed, due to the link capacity full utilization and the physical queue is filled up.

Phase II: We then activate the performance isolation by configuring the slice A to guarantee 12 Mbps throughput and 10 ms delay. Since the guaranteed bandwidth is smaller than the link capacity, less than 1 ms end-to-end latency is observed together with no real delay, by reason of no physical queue being built up. In contrast, around 10 ms virtual delay is experienced using the virtual queue in the P4 pipeline and it can limit the throughput to the guaranteed 12 Mbps.

Phase III: To see its effectiveness under different guarantees, we increase the guaranteed throughput to 120 Mbps, exceeding the 72 Mbps link capacity. We can see that both end-to-end latency and real delay are bounded to around 10 ms with no virtual delay. This is due to the congestion at the real link, and thus virtual queue contributes no virtual delay.

Phase IV: Another network slice B is brought up with traffic flowing from PoP D to PoP B. Both slices are guaranteed with 48 Mbps throughput, while slice B has 5 ms delay guarantee, being one half of the 10 ms delay guarantee to slice A. The

Fig. 4: Proof-of-concept phase IV results

performance of slice A is then limited by the virtual queue due to its higher delay guarantee, while slice B is limited by the maximum link capacity and the real queue. As the results shown in Fig. 4, slice A gets the guaranteed 48 Mbps throughput with around 10 ms virtual delay, while slice B has less than 20 Mbps with a larger 4 ms real delay. We can see that the slice with the most stringent delay bound decides who suffers from a lower throughput, while such slice prioritization can be modified with adjusted real queue priorities.

IV. Conclusions

In this work, we demonstrate that our network slicing approach can retain traffic and performance isolation providing bandwidth and delay guarantees, based on our extensions over the 5Growth architecture and programmable data-plane infrastructure. Our next goal is to apply the proposed solution to the 5G validation trials deployed in the context of 5Growth.

Acknowledgments

Supported by the EC H2020 5Growth project (Grant 856709).

References

[1] Z. Kotulski *et al.*, "On end-to-end approach for slice isolation in 5G networks. Fundamental challenges," in *2017 Federated Conference on Computer Science and Information Systems*, 2017, pp. 783–792.

[2] A. J. Gonzalez *et al.*, "The Isolation Concept in the 5G Network Slicing," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 12–16.

[3] C. Papagianni *et al.*, "5Growth: AI-driven 5G for Automation in Vertical Industries," in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 17–22.

[4] 5Growth RL code repository, https://github.com/5growth/5gr-rl.

[5] 5Growth WIM plugin code repository, https://github.com/5growth/5gr-rl/tree/master/rl/plugins/WIM/ONOS-OpenFlow-Slicing/onos-plugin.

[6] Multus CNI code repository, https://github.com/intel/multus-cni.

[7] Performance Isolation for 5G Network Slicing, https://www.youtube.com/watch?v=DSGHUBFuvYs.