SliceNet programmable data plane control in 5G network slicing
Salva-Garcia, Pablo; Chirivella-Perez, Enrique; Alcaraz-Calero, Jose M.; Wang, Qi; Maione, Biagio; Angelo, Ciriac; Borlizzi, Giacomo; Baldini, Luca; Navid, Nikaein; Bernini, Giacomo; Koutsopoulos, Konstantinos; Figueiredo, Ricardo; Iordache, Marius; Patachia, Cristian

Published: 18/06/2019

Document Version
Peer reviewed version

Link to publication on the UWS Academic Portal

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact pure@uws.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
SliceNet programmable data plane control in 5G network slicing

Salva-Garcia, Pablo; Chirivella-Perez, Enrique; Alcaraz-Calero, Jose M.; Wang, Qi; Maione, Biagio; Angelo, Ciriac; Borlizzi, Giacomo; Baldini, Luca; Navid, Nikaein; Bernini, Giacomo; Koutsopoulos, Konstantinos; Figueiredo, Ricardo; Iordache, Marius; Patachia, Cristian

Published: 18/06/2019

Document Version
Peer reviewed version

Link to publication on the UWS Academic Portal

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the UWS Academic Portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact pure@uws.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
SliceNet Programmable Data Plane Control in 5G Network Slicing

Pablo Salva-Garcia*, Enrique Chirivella-Perez*, Jose M. Alcaraz-Calero*, Qi Wang*, Biagio Maione†, Ciriaco Angelo†, Giacomo Borlizzi†, Luca Baldini‡, Nikaein Navid‡, Giacomo Bernini§, Konstantinos Koutsopoulos§, Ricardo Figueiredo‖ Mario Iordache***, Cristian Patachia∗∗

* University of the West of Scotland, United Kingdom; † Ericsson, Italy; ‡ EURECOM, France; ** Orange, Romania

Abstract—Network slicing has become a major networking paradigm in 5G networks to meet the diverse Quality of Service (QoS) requirements from different use cases. This paper presents the service-based control plane framework for QoS-aware network slicing in the EU 5G PPP SliceNet project, and emphasises the QoS control mechanism for network slicing in the 5G backhaul network by leveraging software-defined data plane programmability. Both design and prototyping details are described. Experimental results have validated the proposed technical approach and implementation, and demonstrated the QoS performance gains in terms of latency and throughput.

Keywords—5G; Slicing; Data Plane; Networking; Traffic Control.

I. INTRODUCTION

The emerging 5G networks are expected to bring a major networking paradigm shift from the existing 4G networks [1]. In particular, network slicing is widely recognised as a most promising approach to addressing the diverging Quality of Service (QoS) requirements posed by various use cases [2] and vertical industries. Existing network slicing work have been largely focusing on creating logical networks through organising Virtual Network Functions (VNFs) and resources over virtualised infrastructures or radio-level resource isolation/sharing. Meanwhile, there are other challenges to be further addressed to achieve the committed QoS along the end-to-end (E2E) data path for user traffic.

The EU 5G PPP SliceNet project targets to achieve advanced E2E network slicing to support use cases of varied QoS and Quality of Experience (QoE). A network slice in SliceNet is an independent, end-to-end logical network running on a shared physical infrastructure, which spans across all the network segments and may include a verticals enterprise networks and multiple network service providers domains to offer one or more services. E2E QoS per slice is controlled according to the Service Level Agreement or QoE optimisation requirements. To this end, SliceNet proposes an overlay service-based control plane on top of 4G/5G control and data plane to achieve dynamic QoS-aware network slicing. The aim of the SliceNet CP is to allow the enforcement of specific and dedicated per-slice runtime configurations rules and policies governing the run-time operations of Radio Access Network (RAN), Multi-access Edge Computing (MEC) segment, Core Network (CN) and Backhaul network. In particular, data plane programmability is explored in SliceNet to address QoS control across the various segments after the RAN segment. It is noted that this paper focuses on the backhaul segment in the E2E data path although the proposed approach is applicable to other non-RAN segments.

The rest of the paper is organised as follows. Section II introduces SliceNet control plane architecture. Sections III and IV present the design of the architecture and workflow of programmable backhaul data plane control respectively. Section V describes the implementation of the proposed solution and experimental results. Section VI concludes the paper.

II. SLICENET NETWORK SLICING CONTROL PLANE ARCHITECTURE

This section presents an overview of the SliceNet control plane architecture as depicted in Fig. 1.

A. Service-based approach

With the intention of not compromising the standards of the 4G/5G Control Plane (CP) in 3GPP [3], SliceNet proposes a standard-compatible overlay approach to extend the functionalities of the 4G/5G Control Plane by defining a set of functional modules [4]. Such modules are deployed in three layers in the SliceNet Control Plane for interacting with the heterogeneous 4G/5G infrastructure to create E2E slices in support of vertical-oriented services. Seen from a bottom-up approach, the lower layer is composed by the Adapters, such components implement a technology-agnostic northbound interface and a technology-specific southbound. In this way, and in order to provide the creation of end-to-end slices, adapters expose the actions to be called by the CPS to prepare each segment of the network. On top of this layer, it is placed the layer responsible for not only implementing the configuration and control functions requested from the Plug&Play (P&P) and QoE Optimiser, but also from other domains of the SliceNet system such as Orchestration, the management and the Cognition sub-planes by using a set of control plane services (CPS). Finally, in the highest layer of this architecture, lies the per-slice runtime customisation component and an the optimisation component (P&P and QoE Optimiser respectively).

Two main principles have been approached to design the SliceNet Control Plane, a Service Based Architecture (SBA) and a technology agnostic APIs abstraction. Thus, allowing individual services to be developed, deployed and upgraded with minimal impact to other services and also enabling a common SliceNet CP information model and logic control.
DPP Adapter) and its subsequently controller, the DPP Adapter) is shown in Fig. 2. It is a software of Service constraints to the different network segments de-

CPS by querying to the CPSR for its reachability. any other authorized SliceNet component can use an specific capabilities. Other service consumers such as P&P, QoE or as well as providing authorization and discovery services register themselves as a service instance in the SBA framework the software component which allows other CP services to software component which allows other CP services to

Each CPS offers specific configuration and control capabilities. Playing thus, a key role in the initial and run time configuration of the slices.

3) QoS Control: It is the responsible of deploying Quality of Service constraints to the different network segments de-

pending on the input parameters gathered from the exposed interfaces.

4) IPC Control: The Inter-Point-of-Presence Connections (IPC) is the responsible, for each slice instance, to deliver a proper interconnections of the slice Network Functions (i.e mostly VNF and MEC applications) deployed in different segments and domains, namely edge (e.g MEC) and Core ones.

C. Adapters and Controllers

Depicted with gray boxes in Fig. 1 are the Adapters and Controllers. There is at least one adapter per each network segment which exposes an agnostic API in its northbound interface for being abstracted by any CPS and exposed to the available service consumers in SliceNet. Furthermore, southbound interfaces of adapters are designed as technology-dependant to allow specific communication with its pertinent controller, that in fact, is the component that will finally enforce the request over its network domain. Under this approach, it is allowed to handle an heterogeneous infrastructure with minimal impact on the overall SliceNet architecture.

III. SLICENET CONTROL OF PROGRAMMABLE BACKHAUL DATA PLANE

This section focuses on the configuration of a slice when QoS control refers to a specific network segment, the backhaul. The Backhaul Data Plane Programmability Adapter (BKH_DPP_ADAPTER) and its subsequently controller, the Flow Control Agent Controller (FCA Controller) are in charge of apply specific QoS restrictions over traffic (crossing through the backhaul data path) that belongs to such configured slice.

Fig. 1 shows a dotted square box that represents the logical flow of the involved SW components when an authorized SliceNet service consumer requests to the QoS CPS to enforce a specific action.

A. Backhaul Data Plane Programmability Adapter

The architecture of the Backhaul DPP Adapter (BKH_DPP_ADAPTER) is shown in Fig. 2. It is a software...
component that provides a SliceNet centralised interface for completing a set of actions over traffic flowing across a specific point, the backhaul. That network traffic can either represent a specific flow or a slice which contains aggregated flows covering the same scope. There is a one-to-many relationship between BKH_DPP_ADAPTER and Slices, which means just one BKH_DPP_ADAPTER instance to cover several flows. In order to deploy an action, BKH_DPP_ADAPTER will receive requests from the QoS Control Service or any other authorized SliceNet functional component (e.g P&P, MP) and interact with the appropriate underlying network controller (FCA Controller) for handling those requests. This component is in charge of providing the agnostic technology abstraction layer capabilities for the consumer by doing a mapping of the technology agnostic API to technology depending APIs provided by the underlying network controller.

![Backhaul DPP Adapter architecture](image)

Fig. 2: Backhaul DPP Adapter architecture

1) bkh_dpp_NBI: it handles the northbound API interface with other control plane service consumers for registering BKH_DPP_ADAPTER instance to CPSR, for receiving the action related requests for a slice.

2) bkh_dpp_SBI: it handles the southbound API interface towards the Backhaul FCA Controller for a slice, for enforcing intent-based operations.

3) bkh_dpp_Map: it has the logic for mapping the intent related information to FCA Controller API information and vice versa.

4) bkh_dpp_Core: it is the engine module which handles all service operations demanded to the BKH_DPP_ADAPTER, inter-working with the other BKH_DPP_ADAPTER internal SW modules for registering the slice BKH_DPP_ADAPTER instance identifier to CP Service Register, controlling the service operations from northbound to southbound interface. It is stateless, so it is supposed to do not store any data about slice and related service operations ongoing.

B. FCA Controller

The FCA Controller, whose architecture is shown in Fig. 3, consumes Intent-based requests provided by the Backhaul DPP Adapter (BKH_DPP_ADAPTER) with the purpose of selecting the specific network endpoint related to the concrete location of the network where the intent will be enforced. This location is the machine that is under the control of an FCA which is responsible to enforce such an intent into the control plane of such a machine. To do so, the FCA Controller will place the intents in a queue and process them by routing them to the proper machines. Notice that the receiver can be only 1 machine if the intent is associated to a unique point of the network or multiple machines if the request is associated to a global network policy across the whole administrative domain. In addition, this component will provide reliability in the deliver of the message so that it will re-transmit them to the destinations in case of connectivity problem in order to make sure there is a consistent state of the control plane. This controller also provide fault tolerance against failures of the FCA by leaving the message in the queues in case the control agent are not ready to consume them and thus allowing a recovery of the state when they are ready again. This FCA Controller is a centralised logical entry point to the infrastructure control which is physically distributed across all the machines controlled.

![FCA Controller architecture](image)

Fig. 3: FCA Controller architecture

1) fca_ctrl_NBI: it handles the northbound API interface with other control plane service consumers for receiving intent-base massages which are describing the action to apply, the traffic which will be affected and the deployment location.

2) fca_ctrl_SBI: it handles the southbound API interface which in this case, follows a different approach for data communication which is based on Advanced Message Queuing Protocol (AMQP) instead of the Representational State Transfer API used in upper layer components for requesting and transmitting data elements. Therefore, the FCA Controller will act as a provider/publisher of AMQP messages which will be sent to the FCAs deployed and ready for consuming these king of messages. It is important to highlight that just those FCAs interested and binded to the proper routing key will be able for consuming and therefore enforce the incoming actions.

IV. WORKFLOW OF THE DPP CONTROL

Fig. 4 shows the workflow when changing the priority of a specific traffic crossing through the network. In summary, any authorized SliceNet functional component can discovery a registered QoS Service through the CPS discovery function for changing the priority of a specific slice/flow. In order to do so, the said caller will send the priority value and the target slice id to that control plane service. QoS component will be, therefore, in charge to collect required parameters (depending on the adapter) by requesting them from the Inventory. After
that, QoS service will use the agnostic API exposed by the BKH_DPP_ADAPTER for setting the action (change priority) including those parameters. The BKH_DPP_ADAPTER will map the input and will transform it in an understandable format which will be sent to the FCA controller. Finally, the FCA controller will enforce the rule (change priority) in a specific location which will change the priority of all specified matching traffic.

![Fig. 4: Workflow of a QoS DPP constraint onboarding process.](image)

The following steps describe the workflow in more detail:

1) A caller (CPS Consumer) will retrieve QoS Control Plane Service address via CPSR function.
   - (sliceID, CPSType = Qos_CP)

2) Such Caller will also call the set_priority endpoint of the QoS service API to provide:
   - (sliceID, priority value (If the case of new priority))

3) QoS service will retrieve a proper adapter (E.g. the back-haul data plane programmability) by using the CPS discovery function.
   - (sliceID, CPSType = BKH_DPP_ADAPTER)

4) QoS Service will retrieve the required slice parameters from the inventory in order to fulfill the request to be sent to such BKH_DPP_ADAPTER.

5) QoS Service will map those parameters which were gathered from the inventory.

6) Qos Service will call the agnostic BKH_DPP_ADAPTER API in order to apply a priority value over a specified network traffic.
   - (sliceID, slice_parameters)

7) BKH_DPP_ADAPTER will map the retrieved payload data to an intent.

8) BKH_DPP_ADAPTER will publish that intent into the FCA controller message bus.

9) FCA Controller will forward such intent to the specific FCA.

10) The FCA’s Operation result will be returned back to the initial service consumer for providing acknowledgement about the triggered action.

V. IMPLEMENTATION AND EMPIRICAL RESULTS OF DPP BASED ON 5G TRAFFIC CONTROL

A. Implementation

Interfaces involved along the defined slice data-path are configured at the FCA’s deployment time. Multiple configurations are allowed to cover all specific particularities that may be required depending on the needs of each slice, for instance, the number of priority queues as well as bandwidth percentage assigned to them. With the aim of providing clarity, Fig. 5 shows a basic interface configuration where, by using Traffic Control (TC) [5], such interface has been split in four different priority queues, each of them with the same bandwidth percentage (25%), providing thus, a maximum of 250Mbps each one.

![Fig. 5: Interface configuration.](image)

B. Testbed description

Fig. 6 depicts the reference testbed where the proposed implementation is validated. This particular scenario has been setup for analyzing traffic crossing a specific network segment point. It is assumed that an initial interface configuration, like the one described in Section V, has been placed during the deployment time of the FCA. Furthermore, it is understood that from the upper layers of the presented SBA, a CPS consumer has triggered the workflow presented in Section IV, which has consequently, unleashed the on-boarding of a network slice for a specific service network traffic which requires a low latency.

Let us define network traffic $NT_{LL}$ and $NT_{NO}$ as the network traffic which belongs to such low-latency slice and network traffic which does not respectively. Traffic from both, $NT_{LL}$ and $NT_{NO}$ is injected to the testbed for 30 seconds from the beginning of the experiment. As it will be further explained in section V-C, synthetic traffic is also injected to the testbed in order to emulate a real congestion in the network, and then, given the possibility to this study to analyze the behaviour of the traffic depending on whether it belongs to the low latency slice or not. It is noted, that the internal classifier of the FCA has the capability to dissect overlay networks traffic, as well as the capacity to recognize its belonging to a specific slice within context of the data plane.

C. Experimental results

Fig. 7 shows latency information of the empirical performance carried out. Initially all traffic (sliced and no sliced...
one, \textit{NT\_LL} and \textit{NT\_NO} respectively) have a low latency even when they are being dispatched in a different priority queues. To be precise, \textit{NT\_LL} is using the most priority queue and \textit{NT\_NO} (which is not being classified as a part of the slice) is using the default queue. Later, at time 10, a synthetic traffic (\textit{NT\_SYN}) of 260Mbps is injected into the network. Since this experiment defines a maximum bandwidth of 250Mbps per queue, such default queue becomes congested and latency of \textit{NT\_NO} is dramatically increased while the traffic defined as a part of the Low Latency slice it remains stable offering low latency values. At time 20, it is simulated an action coming from the upper layers of the infrastructure which decides to decrease the priority of traffic belonging to the Low Latency slice. From that moment, all traffic is forwarded through the default queue and as it can be seen in the figure, latency becomes unstable.

![Testbed Implementation](image)

**Fig. 6:** Testbed Implementation

Fig. 8 shows results of the throughput performance. As expected, from time 0 to 10, both \textit{NT\_LL} and \textit{NT\_NO} traffic are reaching their hundred percent of the throughput since there is no other traffic in the network. It is important to highlight here, that traffic belonging to the low latency slice (\textit{NT\_LL}) is being dispatched before the traffic which does not belong to any slice (\textit{NT\_NO}), however, because there is no congestion in any queue, there is hardly any difference. Once again at time 10, synthetic traffic (\textit{NT\_SYN}) of 260Mbps is injected into the network and consequently, the default queue is overloading. As it can be seen in the graph, the throughput belonging to \textit{NT\_NO} drops nearly to 50 percent while latency increases (as demonstrated in Figure 7), and drops again from time 20 to 30 up to about 35 percent when, the slice is removed from the system and \textit{NT\_LL} is not prioritized anymore.

![Throughput comparison in percentage](image)

**Fig. 8:** Throughput comparison in percentage

### VI. CONCLUSIONS

SliceNet introduces an overlay control plane on top of 5G control and data planes to enable advanced QoS-aware network slicing. The proposed QoS control scheme leverages data plane programmability and is able to enforce slice QoS configuration and re-configuration on demand. Empirical results validate this technical approach.

### ACKNOWLEDGMENT

This work was funded in part by the EU Horizon 2020 5G-PPP Programme under Grant Agreement Number H2020-ICT-2016-2/761913 (SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks).

### REFERENCES


