UWS Academic Portal

Toward hardware-accelerated QoS-aware 5G network slicing based on data plane programmability

Sanchez, Rubén; Malagonb, Pedro; Matencio Escolar, Antonio; Wang, Qi; Alcaraz Calero, Jose M.

Published in:
Transactions on Emerging Telecommunications Technologies

DOI:
[10.1002/ett.3726](https://doi.org/10.1002/ett.3726)

Published: 28/08/2019

Document Version
Peer reviewed version

[Link to publication on the UWS Academic Portal](#)

Citation for published version (APA):
Sanchez, R., Malagonb, P., Matencio Escolar, A., Wang, Q., & Alcaraz Calero, J. M. (2019). Toward hardware-accelerated QoS-aware 5G network slicing based on data plane programmability. *Transactions on Emerging Telecommunications Technologies*. <https://doi.org/10.1002/ett.3726>

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.

UWS Academic Portal

Towards hardware-accelerated QoS-aware 5G network slicing based on data plane programmability

Sanchez, Rubén; Malagonb, Pedro; Matencio Escolar, Antonio; Wang, Qi; Alcaraz Calero, Jose M.

Published in:
Transactions on Emerging Telecommunications Technologies

DOI:
[10.1002/ett.3726](https://doi.org/10.1002/ett.3726)

Published: 28/08/2019

Document Version
Peer reviewed version

[Link to publication on the UWS Academic Portal](#)

Citation for published version (APA):
Sanchez, R., Malagonb, P., Matencio Escolar, A., Wang, Q., & Alcaraz Calero, J. M. (2019). Towards hardware-accelerated QoS-aware 5G network slicing based on data plane programmability. Transactions on Emerging Telecommunications Technologies. <https://doi.org/10.1002/ett.3726>

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.

"This is the peer reviewed version of the following article: Sanchez, R., Malagonb, P., Matencio Escolar, A., Wang, Q., & Alcaraz Calero, J. M. (2019). Towards hardware-accelerated QoS-aware 5G network slicing based on data plane programmability. Transactions on Emerging Telecommunications Technologies, which has been published in final form at <https://doi.org/10.1002/ett.3726>. This article may be used for non-commercial purposes in accordance with Wiley Terms and Conditions for Use of Self-Archived Versions."

<https://authorservices.wiley.com/author-resources/Journal-Authors/licensing/self-archiving.html>

ARTICLE TYPE

Towards Hardware-Accelerated QoS-aware 5G network slicing based on data plane programmability

Ruben Ricart-Sanchez¹ | Pedro Malagon² | Antonio Matencio-Escolar¹ | Jose M. Alcaraz Calero*¹ | Qi Wang¹

¹University of the West of Scotland, , Paisley, UK

²Universidad Politecnica de Madrid, , Madrid, Spain

Correspondence

*Jose M. Alcaraz Calero. Email: Jose.Alcaraz-Calero@uws.ac.uk

Present Address

This is sample for present address text this is sample for present address text

Summary

The diverging requirements from various vertical industries have driven the paradigm shift in the next-generation (5G) mobile networks, where network slicing has emerged as a major paradigm for this purpose by sharing and isolating resources over the same 5G physical infrastructure. To truly fulfil the different Quality of Service (QoS) requirements imposed by different network slices for different vertical applications, it is essential to introduce a programmable data plane that is aware of QoS and is configurable to enforce the QoS commitments. In this paper, we focus on designing, prototyping and evaluating a novel QoS-aware Data-Plane Network-Slicing framework for the edge and core network segments of a 5G network. The proposed framework is capable of dealing with differentiated services through hardware-based traffic classification, priority configuration and traffic scheduling. By leveraging the latest open-source FPGA (Field-Programmable Gate Array) platform, we prototype the proposed framework and empirically evaluate the performance of the prototyped system. Experiment results demonstrate the capabilities of the proposed framework in terms of achieving QoS-aware network slicing at the data plane.

KEYWORDS:

Network slicing, Programmable Hardware, 5G, Mobile Computing, Network Architecture and Design, Quality of Service (QoS)

1 | INTRODUCTION

The Fifth Generation mobile networks (5G) have gained global momentum thanks to the recent Research and Development (R&D) and standardization activities. It has now been widely recognized that the software-networking based architecture approach with NFV (Network Function Virtualization) and SDN (Software-Defined Networking) as cornerstones are essential in achieving the 5G vision of being “programmable, software driven and holistically managed to enable a diverse range of services in a profitable way”, as summarized by the Chairman of the 5G-PPP Association¹. In particular, Network Slicing has emerged as a most promising game changer in the remarkable paradigm shift from 4G to NFV/SDN-enabled 5G era. According to the visions by various leading 5G standardization organizations and other stakeholders, network slicing has been identified as a crucial enabler for provisioning flexible, cost-efficient and tailored services in software-networking based 5G networks. A universal, standard definition of a 5G slice is yet to be agreed on, although a most cited definition on 5G slice is provided by New Generation of Mobile Networks (NGMN)². According to NGMN, a 5G slice “supports the communication service of a

FIGURE 1 Different Network Segments involved in the 5G Infrastructure.

particular connection type with a specific way of handling the C- and U-plane for this service. To this end, a 5G slice is composed of a collection of 5G network functions and specific Radio Access Technology (RAT) settings that are combined together for the specific use case or business model.” There are more standardisation organizations which have provided a network slice definition. 3GPP organisation concluded that a network slice is “an end to end logical communication network, within a Public Land Mobile Network (PLMN) and includes the Core Network (CN) Control Plane, User Plane Network Functions and 5G Access Network (AN).”³. In IETF (Internet Engineering Task Force) L.Wang and L.Geng defined E2E (End-to-End) network slice as “a cross-domain network slice which may consist of access network, transport network, (mobile) core network. The concept here of network slice is expanded to a wider area and comprises of several functional components. Driven by the multiple requirements from users, network slice instances may include several service components that each of which may require a set of network resources and attributes in form of a network slice”.⁴. Finally, ETSI (European Telecommunications Standards Institute) rather than defining network slicing, focuses on the mapping of the network slicing to NFV concept and describes how NFV architecture supports it under the standards of diverse organizations such as 3GPP⁵.

The expected benefits of introducing Network Slicing in 5G are multi-folded. Firstly, network slices would maximize the sharing of network resources within and across domains, thereby substantially reducing the capital expenditure (CAPEX) for 5G network operators. Secondly, network slicing allows a high-degree of flexibility of creating dedicated logical networks with customer-specific functions and thus can meet diverging requirements from diverse vertical businesses. Thirdly, network slices can substantially upgrade operational capabilities as with intelligent slicing and slice life cycle management it is possible to offer configurable warranties in Quality of Service (QoS) and/or Quality of Experience (QoE). These configurable warranties in service quality will open a significant number of new markets and enable a wide range of demanding, divergent and innovative use cases. For instance, the safety of autonomous driving can be significantly improved thanks to the warranted ultra-low-latency in sending commands to the car, and reliable remote mobile life assistance could become a reality where an Ultra-High-Definition (UHD) (e.g. 4K) video stream can be transmitted with the warranted ultra-low-latency and mobile broad bandwidth. Therefore, if the full potential is achieved, 5G slicing could be considered as one of the most important innovations in the communications of the decade due to its impact at a worldwide level.

To achieve an end-to-end network slicing, all the different network elements deployed along the data path of the 5G infrastructure should be able to be programmed to set up specific configurations of the resources to cope with specific use cases. Figure 1 shows the main network segments that constitute the communication data path of 5G infrastructures. Vertical businesses and 5G subscribers get access to the network by mean of the Radio Access Network (RAN) which is connected to the Edge to allow a Mobile Edge Computing (MEC) architecture. MEC allows data processing using Cost-of-the-Shelf computing close to the user and fostering the dynamic deployment of network functions. Different Edges are connected to the Core Network where the communications are being controlled and switched to their destinations, mainly to other administrative domains being interconnected using Neural Internet Exchange point of coming back to other edges. 5G is a major technology in growing industrial digitalization, creating different use cases which represent the vertical business (e.g. public safety, automotive, manufacturing, healthcare, etc); in the same way, 5G subscribers are people who will take benefits of the advantages provide by the novel 5G networks.

The main aim of this paper is to design, prototype and validate an effective hardware-accelerated network slicing framework suitable for 5G architectures to allow the programmability of the communication parameters such as delay, packet loss and jitter in the data path, thereby being able to slice the network to cope with different use cases simultaneously for the Edge-to-Core network segment (E2C). To achieve this aim, several challenges have been addressed, which in turn are the main contributions of this work. Firstly, a new hardware packet queuing structure has been designed and prototyped as an enabler for network slicing to allow differentiation of 5G traffic. Secondly, a new packet classification algorithm has been designed to fulfil the demanding requirements imposed by the usage of multi-tenancy and user mobility in the 5G E2C network segment. Thirdly, a new packet scheduling scheme has been designed and prototyped in order to keep performance fairness among different users that are sharing the same network slice and to keep performance isolation between users that are allocated to different network slices of different QoS requirements. Finally, extensive experiments have been conducted to test and validate the proposed framework using a real Edge-to-Core scenario where our hardware-based prototype has been deployed, and to demonstrate the effectiveness and scalability of the implementation. The solution proposed is a novel 5G network slicing technique different from the traditional DiffServ and QoS-aware forwarding where isolation of network performance between different slices has been the foundations of the design. In this context, network performance is defined as delay, packet loss, jitter and throughput. Every of these network slices can, in turn, have traditional DiffServ flows in order to prioritize those flows inside of the same slice. Traditional DiffServ or other QoS-aware forwarding policies provide a common semantics applied to all the network flows equality and thus do not expose mechanisms for the network administrators to allow the customization of the behaviour of different network services, i.e. rules. Nevertheless, this approach goes father by allowing the programmability of such rules in order to configure the behaviour of each of the slices.

The rest of the paper is organized as follows. Section 2 reviews the state of the art of related areas including the different software and hardware data path solutions and recent network slicing proposals. Section 3 describes the proposed QoS-aware data-plane network slicing framework with a detailed description of the programmable data-plane path. Section 4 proposes the taxonomy of different types of network slices. Subsequently, section 5 presents technical details on the prototyping of the proposed framework, highlighting the various challenges addressed in the implementation. Section 6 describes the testbed deployed to validate and evaluate the technical contributions through empirical performance tests and analysis. Finally, section 7 summarizes the main findings and outlines future research work.

2 | RELATED WORK

2.1 | Network Slicing Concepts and Approaches

2.1.1 | Network Slicing Concepts

It is noted that there is no universally accepted definition of network slices yet despite numerous studies on this topic. Various standardization development organizations (SDOs) such as ITU/3GPP, IMT-2020, IETF, ETSI and industry alliances or initiatives such as NGMN have proposed their versions of network slice definitions. This paper is inclusive and takes into consideration all the definitions proposed. Each definition uses different terminologies, thus it has been used, as a matter of example, the NGMN terminology² where it is defined a Network Slice Instance (NSI) as a set of run-time Network Functions (NFs), and resources to run these NFs, forming a complete instantiated logical network to meet certain network characteristics (e.g., ultra-low-latency, ultra-reliability) required by a Service Instance(s). Network slice instance examples include enhanced Mobile Broadband (eMBB), massive Internet of Things (mIoT), and Ultra-Reliable Low-Latency Communication (URLLC), as defined by the International Telecommunication Union (ITU). An NSI can include zero, one or more Network Slice Sub-network Instance (NSSI), which itself is a run-time construct and comprises a set of NFs and the required resources. IP Multimedia Subsystem (IMS) is an example of a sub-network Instance. The Resources referred to are assets for computation, storage or transport including radio access, physical or logical. This definition has been widely recognized in other initiatives and SDOs. For instance, 3GPP has largely adopted this definition in network slicing related specifications⁶.

In order to offer performance warranties, Network Slicing in a 5G E2E architecture requires isolation between slices. The performance of a particular slice cannot be influenced by the Control Plane and the Data Plane of any of the other slices. Consequently, multiple slices can be used for different services and a great heterogeneity of use cases can be treated.

Our definition of network slicing includes the isolation of performance of a set of network flows. It has been demonstrated that it is impossible to perform such isolation in software. The reason is that a computer runs multiple software at the same time (kernel processes, drivers, user-space applications, etc) using the same hardware resources. Thus, without the proper support to

isolate hardware, it is impossible for the software to make use of such isolation capabilities. And this is exactly what is being proposed in this contribution.

2.1.2 | Approaches for Network Slicing

Related work on network slicing can be largely classified into three groups. Most of the existing work on network slicing have been focusing on network slices at the NF level, which belongs to the first group. A network slice composed by NFs can be created by concatenating the involved NFs by exploring various mechanisms such as Service Function Chaining(SFC)⁷ or leveraging the Network Service (NS) model in the ETSI (European Telecommunications Standards Institute) MANO (Management and Orchestration) architecture⁸. An example can be found in the 5GNorma project⁹, where both VNFs and PNFs are considered to achieve NF-level network slicing. Another project SONATA¹⁰ envisions a Slice Manager to manage the life-cycle of a slice, and a control API to operate a slice using a slice ID that groups all the resources, network functions and service elements of that slice.

The second group of work concentrates on Radio Access Network (RAN) slicing by investigating RAT-specific schemes to achieve network slicing at different layers of the protocol stack: spectrum sharing, MAC, resource blocks etc. For instance, the SPEED-5G project¹¹ develops a more efficient use of spectrum for network slicing and new and a MAC layer that allows prioritizing and handling traffic across heterogeneous RANs. Moreover, the COHERENT project¹² proposes a real-time controller for RAN slicing. In addition, in a RAN, An et al.¹³ proposes a device-triggered network-control mechanism to enable a UE to discover, select and access the most appropriate E2E network slices, and simulation results show reduced attachment delay and signalling overhead.

The final group emphasizes the importance of QoS awareness for network slicing by exploiting data plane programmability. There is little existing work in this group; however, it is crucial to achieving controllable QoS along the end-to-end data path towards guarantee the QoS committed in the Service Level Agreement associated to the slice-based service for the end user.

These three groups of research are fully complementary to each other since they are focused on different approaches required to achieve a true slicing of the network. This research work has been focused on the programmability of the data plane which is the foundations to allow the other two approaches to be designed, implemented and prototyped.

In¹⁴, a 5G-ICN architecture based on the NFV/SDN framework, where applications and devices seek connectivity through the RAN to ICN gateways, to realize a top-down service-centric platform. This paper proposed the used of domain controllers to ensure slicing SLA management, also allowing mobility as a service. Although this solution presents a theoretical idea about how to establish end-to-end slicing support they do not provide a concrete solution and neither empirical results. They have also proposed that the data plane of their solution can be based on virtual network overlays or deeply programmable hardware, but they do not present any implementation evidence.

Authors provide in¹⁵ a theoretical approach to ensure network slicing in a 5G ecosystem, mainly focusing on the RAN network segment. The approach presented in this contribution is based on the mobile network architecture framework described in the research project 5G NORMA⁹. It proposed that virtualised network elements which composed the RAN segment should form a direct path to ensure the forming of network slices, avoiding the traditional approach with indirect paths. This paper does not provide any empirical analysis of the concept of network slicing presented for the RAN and do not cover edge and core network segments which are those addressed in this research work.

Following the architecture presented in the research project 5G-Crosshaul¹⁶, Li et al.¹⁷ present an SDN/NFV-based control plane that enables multi-tenancy through network slicing. Although this work is mainly focused on the transport network aspects related to the combined fronthaul and backhaul, they only present general concepts of network slicing that can be applied to other segments of a mobile system. They also propose a data plane, with MPLS encapsulation support and VLAN tagging, to store the backbone header and the tenant frame, ensuring the multi-tenancy support. It is noticed that this contribution is far away from our approach where a more complex data path has been implemented. Furthermore, this contribution only presents general concepts with any empirical validation.

Zhang et al.¹⁸ proposed a logical architecture for network-slicing-based 5G systems, and present a scheme for managing mobility between different access networks. The architecture presented in this paper is very simple and does not present too many details, following a basic SDN approach. Although they argue that their architecture support slicing in the access network, edge cloud and core cloud, they only present simulation results with three different level of slice eMBB, uRLLC and IoT slice. There is not a real empirical validation in this contribution and they do not mention any multi-tenant and mobility support in their architecture.

In¹⁹, Mayoral et al. present a 5G slicing architecture with multi-tenant support, where tenants can control the virtualised network functions and cloud resources, thanks to an open API. This contribution only presents a DPI use case where times about the creation of a DPI-VNF are shown.

An et al.¹³ propose an end-to-end network slicing for a 5G communication system. This paper is mainly focused on the RAN segment of the 5G network. They do not present concrete details about how to apply network slicing in the backhaul. Although they present a detailed analysis and deployment of the slicing in the RAN, they do not mention how to achieve this QoS-aware in the edge-core segment and neither an empirical validation.

In the rest of this section, we focus on reviewing enablers and potential approaches to design and implement QoS-aware network slicing based on data plane programmability. Furthermore, a 5G infrastructure is a varied and complex ecosystem, with different types of software and hardware components. To ensure an E2E QoS, all the elements involved in the data path, both software and hardware, must be able to honour the established QoS policy. For all the above, the design of a 5G network slicing architecture leveraging data plane programmability is a complex task.

2.2 | Queuing disciplines and algorithms

First of all, packet classification and scheduling techniques are fundamental for achieving QoS awareness and control. There are many well-known packet classification algorithms and packet scheduling disciplines and their software implementations have been extensively tested over the years. They all handle concepts, data structures and components such as queues, flows, tokens, buckets, shaping, scheduling, policing, classifying, marking or dropping. The choice of a particular algorithm or discipline depends on the scenario, the type of traffic and the problem to be solved (e.g., limiting bandwidth, preferred bandwidth for a particular application or user, or ensuring that a particular type of traffic is dropped, etc.). For example, the traffic control module of the Linux kernel²⁰ implements several of these queuing algorithms and scheduling disciplines. Some of the best known queuing disciplines are Round Robin (RR), Priority Queuing (PQ), Stochastic Fair Queuing (SFQ), Generic Random Early Drop (GRED), Token Bucket Filter (TBF), Hierarchical Token Bucket (HTB), Controlled Delay (CoDel) or Heavy-Hitter Filter (HHF).

Priority queuing (PQ) is a queue algorithm management which is based in a group of queues where each of them has a different priority. The priority is established per queue, it means that items stored in a queue with high priority are served before those items stored in a low priority. The priority of each queue should be previously established. If items are stored in the same priority queue, they are served according to the order in which they were enqueued (FIFO). A disadvantage of this algorithm is that it can cause starvation in lower priority queues.

Round Robin (RR) is one of the most popular algorithms employed by network schedulers and process in computing. These algorithms ensure that different processes or network queues have the same priority, it ensures a totally fair sharing of the time per queue. Furthermore, this algorithm also avoids the starvation problem present in PQ algorithm but does not support prioritization.

The Intel 82599 10 GbE Ethernet NIC²¹ implements a scheduling algorithm composed by a combination of the previous ones (RR and PQ). It presents two levels of network queues, scheduling the packets in each of those levels with a different algorithm. The first level of queues is composed by 8 priority queues, which ensures the prioritization and performance isolation of the network traffic and second level of queues is scheduled by an RR algorithm, ensuring fairness in the resource sharing. A similar approach has been used in this contribution, as a foundation to design the programmability of the network slices in the 5G data path.

Some studies attempt to advance the state of the art, e.g., by introducing new types of queues. For instance, Sivaraman et al.²² propose a programmable packet scheduling in hardware using PIFO (Push-In First-Out) queues rather than the traditional FIFO queues. They show how to implement a PIFO queue in hardware and how PIFO queue allows a packet to be placed in an arbitrary position within the queue. They show how a PIFO queue can be used to perform priority and calendar queues that can be combined to implement a wide range of scheduling algorithms.

Furthermore, there are a number of tools to enable Data Plane monitoring and control. OpenFlow²³ defines rule/policy-based traffic flow related processing, including flow matching, flow forwarding, flow QoS metering etc. Specific control actions may include packet forwarding to a particular port or ports, packet encapsulation and forwarding to the controller, packet forwarding to the normal processing pipeline or packet dropping etc. For monitoring purposes, statistics such as packet and byte count are available. OpenFlow is applicable to both physical and virtual (hypervisor-based) forwarding devices.

2.3 | Software-based Data Path Programmability

Data Plane programmability can be achieved through software- or hardware-based approaches. This subsection focuses on the former. Firstly, DPDK (Data Plane Development Kit)²⁴ is an open source project that provides mechanisms for fast data processing for Data Plane applications, mostly running in the Linux userspace and being agnostic to processors (Intel and others). DPDK employs the kernel bypass approach, which allows the applications in the user space to directly communicate with the hardware (physical) or virtual devices without involving the Linux kernel and thus circumvent the performance limitations of the Linux kernel caused by interrupts, the complexity of the `sk_buff` struct etc.

In contrast to the kernel bypass approach taken e.g., in DPDK, XDP (eXpress Data Path)²⁵ through the IO Visor Project²⁶ addresses the performance limitations of kernel space by enabling bare-metal packet processing at the lowest point in the software stack in the kernel space for improved speed whilst achieving Data Plane programmability. Essentially, XDP creates an integrated fast path in the kernel stack. The in-kernel XDP Packet Processor intercepts the incoming packet before it is sent to the normal kernel process. It processes RX packet-pages directly out of the driver. Basic XDP packet processor actions include packet forwarding, dropping, normal receiving and steering (to another CPU for processing), and generic receive offloading etc. The XDP approach is agnostic to CPU/hardware.

PF_Ring²⁷ is another software technology that speeds up packet capture via kernel bypass. PF_RING polls packets from Network Interface Cards (NICs) through the Linux NAPI, which copies packets to the PF_RING circular buffer (ring), bypassing the kernel stack, and then the user-space application reads packets from the ring. PF_RING can distribute incoming packets to multiple rings, and thus multiple applications can read their packets simultaneously. PF_Ring is not integrated with the mainstream Linux, and special kernel modules need to be launched.

Salva-Garcia et al.²⁸ present a software-based framework which is able to deal with 5G multi-tenant UHD video traffic. This solution is based on Netfilter, which achieves flow control over edge to core network segment using a set of hooks inside the Linux kernel, providing callback functions for every packet that cross the respective hook established by a Netfilter rule. This solution is able to deal with 5G multi-tenant traffic and presents data plane programmability support, but it does not provide slicing support for those scenarios and neither it presents scalability in terms of performance and latency.

In²⁹, a module for low-delay filtering of streams is proposed. This solution employs a function on an access router that controls the mobility signalling for mobile endpoint that is associated with an access link. However, the data-path implemented for filtering the streams present some limitations, it is implemented in the context of LTE Evolved Packet Core (EPC) and it does not present multi-tenant support and thus there is no support for nested encapsulation. In³⁰ they implement GTP header management in software to allow the use of traditional IP hardware-acceleration, however they do not provide user mobility.

In³¹ a software-based Network Intrusion Detection System (NIDS) for 5G multi-tenant traffic is proposed. The data plane presented in this contribution allows the control of 5G multi-tenant HTTP traffic. To achieve it, an internal modification of Snort³² has been carried out allowing the protection for 5G multi-tenant scenarios. Although this solution presents a data plane programmability support, there is no mention to the QoS of network traffic and neither to the slicing support. This solution neither provides scalability in terms of performance to achieve the 5G KPIs.

The above software-based mechanisms achieve data plane programmability although the performance is compromised due to the inherent speed disadvantage in software-based implementations and they have never been used for the design of network slicing of 5G data paths.

2.4 | Hardware-based Data Path Programmability

To accelerate the speed of data plane monitoring and processing, a few hardware platforms have emerged recently. In this category, NetFPGA³³ is a line-rate, open networking platform that enables hardware-based programmable data path. The state-of-the-art NetFPGA SUME is an FPGA-based PCI (Peripheral Component Interconnect) Express (PCIe) board with I/O capabilities for 10 (and up to 100) Gbps operation, and the workflow is based on SimpleSumeSwitch depicted in Figure 4. The platform can be employed as NIC, multi-port switch, or firewall, among other Data Plane networking or testing devices. The open source and low-cost nature of the platform allow prototyping 10 Gbps solutions in R&D projects. Netcope³⁴ is an alternative FPGA-based networking platform, which supports up to 100 Gbps Ethernet interfaces. It provides architecture similar to the SimpleSumeSwitch but with other capabilities such as multiple DMA (Direct Memory Access) channels per interface and DPDK Driver support. P4 is the recommended language for programming both NetFPGA and Netcope platforms (e.g.,³⁵). Both NetFPGA and Netcope provide hardware-based programmable data plane; however, it is noted that both of the current platforms lack advanced packet scheduling and processing required by QoS-aware network slicing.

In³⁶, a NetFPGA-based data path for multi-tenant 5G traffic is proposed, this presents an exhaustive analysis in terms of performance, scalability and reliability of the data path developed. In³⁷ a 5G NetFPGA-based firewall for network segment delimited by edge and core is presented, which owns an internal 5G data path to ensure the correct network traffic matching, processing and control thanks to internal rule-based storage. However, no slicing is implemented in this contribution. Although it is clear that different hardware-based data path solutions are available for 5G multi-tenant networks, none of them provides slicing support for the edge to core segment in multi-tenant scenarios.

The Intel 82599 10 GbE Ethernet NIC²¹ offers QoS mechanisms such as L2 filters, Unicast and Multicast filters, L3/L4 5-tuple filters, SYN packet filters for routing TCP packets. It is also capable of working in virtualised environments, for which it handles VLAN filters and offers support for up to 64 virtual machines per port. Using Flow Director it is possible to identify the received traffic and assign a queue for classification, load balancing and matching between flows and CPU cores. The Rx and Tx queues are all FIFO. The filters and flow rules are programmable through an interface and using the registers of the card. This card provides a novel scheduling algorithm, composed of two levels of network queues, which ensure the QoS of the traffic transmitted. These queues are managed by 8 bits, 3 for up to 8 priority queues and 4 bits for 16 round-robin queues, reaching a total of 128 queues. The selection of these queues is based on the values provided by the 5-tuple previously mentioned in this paragraph. This approach is far advanced from the approach presented by traditional NICs which usually presents one queueing level.

2.5 | Limitations of Existing Work

It is noted that none of the above tools and platforms in either software- or hardware-based Data Plane programmability approaches have achieved the capabilities required by 5G network traffic classification, monitoring or processing. With regard to the limitations in flow classification for QoS monitoring, only IP packets are candidates for the flow filters. This means that non-IP packets and packets with tunnelling such as those in 5G traffic would miss all the filters and thus are not classified. Moreover, the packet processing actions for QoS control are also limited, especially in the existing hardware platforms. For instance, the flow director of the Intel 82599 10 GbE Ethernet NIC is only able to perform two actions: sending or dropping a packet. When a packet matches a flow in the Flow Director Table, it is assigned and sent to a queue. Otherwise, it is dropped. With such limited functions, it is not possible to modify a data path by performing advanced actions for QoS support such as setting the queue assigned based on not only the traditional DSCP (Differentiated Services Code Point) value of the IP header but also in encapsulated traffic, and by copying the DSCP value from the inner IP header to the outer one and by selecting the queue based on policies defined by the administration in order to isolate performance ; All these actions have been supported in this contribution.

Filtering mechanism provided by Intel 82599 10 GbE NIC are limited and do not allow to filter network traffic beyond traditional IP packets or VLAN tagged packets. This card only provides a flexible 2-byte tuple anywhere in the first 64 bytes of the packets, however these 64 bytes are not enough in a MEC infrastructures to identify a final 5G user, due to the fact that final user packets are conforming by two levels of encapsulation (GTP over VXLAN), when they are processed in the core of the architecture. The solution presented in this paper provides a novel hardware-based data path with support for the customization of network services behaviour with any limitation, going beyond traditional IP traffic and thus, providing QoS-aware performance isolation for 5G MEC network traffic.

Based on the above review of the existing work, it can be concluded that there is a gap in the literature in designing and prototyping a QoS-aware hardware-accelerated 5G network slicing framework by exploring data plane programmability and taking advantages of hardware-based accelerations.

3 | MULTI-TENANT 5G ARCHITECTURE OVERVIEW

Figure 2 shows a 5G Multi-Tenant Mobile Edge Computing infrastructure where User Equipment (UE) is connected to 5G RANs belonging to different telecommunication operators. The Mobile Edge architecture allows for the logical separation of hardware resources in a multi-tenant environment where multiple network operators share the same physical infrastructure. These 5G RANs are mainly composed of Distributed Units (DUs) and Centralized Units (CUs), which allow the users to access to the core network. Edge 1 and Edge 2 represent two different physical computers allocated in different geographical locations. They host CU Network Function Virtualization (NFV) services in the context of different operators, represented a different background

FIGURE 2 Proposed NetFPGA slicing multi-tenant 5G architecture.

colour. Edges are connected to the Core network segment. The Core segment is composed of different physical machines, which are employed to deploy different 5G Core network services belonging to different telecommunication operators. Each 5G Core has a set User Plane Function (UPF) in charge of forwarding user data in the data path and a set of control and management functions in charge of authentication, user mobility, handover management, user registry, and so on. A comprehensive description of the roles in each of the components of the 5G architecture is provided by Kim et al.³⁸.

The architecture presents an “Hybrid Approach” based on a combination of both software and hardware forwarding devices to separate the required roles to achieve a slicing-friendly infrastructure between the software and hardware components. Figure 2 shows how the traffic coming from the hardware NIC is now received in the Virtual Switch hosted in the Host machine. In terms of role separation between software and hardware, it is proposed to minimize the possible use of the software-based approach, by off-loading as much as possible workload into the hardware capabilities. Compared with a pure hardware approach, this Hybrid Approach would yield lower performance whereas it provides more flexibility in the Control Plane since it allows having double control points layers for the Infrastructure Provider and Network Operator. The Infrastructure Provider’s control points layer is composed of two control points: one flexible yet slow software control point and a limited yet fast hardware control point. The Network Operator’s control points layer, however, only has one control points layer for the tenant in the kernel space of the VM.

This paper focuses on the hardware-accelerated control points to support 5G network slicing based on data plane programmability designed for the Edge-to-Core network segment. In this network segment, the packets of the 5G UE are encapsulated by the CU in the GPRS (General Packet Radio Service) Tunneling Protocol (GTP) to allow user mobility along different locations and is also encapsulated in VXLAN or Generic Routing Encapsulation (GRE) by the Virtual Switch to allow traffic isolation in

FIGURE 3 Multi-tenant packet definition.

a multi-tenancy environment and thus to ensure that telecommunication operators are logically separated. The proposed framework is deployed on NICs that are deployed in the interconnection between Edge and Core. They are labelled as NetFPGA NIC in Figure 2. Figure 3 shows the frame structure of the packets passing through this network segment to show the challenge required in the design of the packet classifier until it is able to disaggregate the network traffic for each of the 5G users in order to allow network slicing according to the subscription of each 5G user, which is clearly beyond current state-of-the-art classifiers published in literature.

By using such complex hardware-based packet classification, the proposed framework enables different logical data paths (lanes/queues) per each of the different 5G users of the infrastructure, allowing traffic to flow through such lanes/queues without horizontal collisions/interference between users. Each logical data path should be programmed to designated QoS-aware ones and best effort ones to allow the realization of paths with controllable QoS and those with best effort delivery service, respectively. They are isolated from each other so that the traffic in each category would not affect the other category for effective management and fair provisioning even for the best effort slices.

4 | NETWORK SLICING DESIGN

A traditional NIC data path usually comes with 1, 4 or 8 lanes/queues in order to allow different degrees of traffic differentiation. Then, when a network packet enters the NIC, it is queued in one of these lanes/queues available according to the priority assigned to this packet. Thus, if the packet comes from a trusted environment, IP Differentiated Services Code Point (DSCP) values can be used to determine traffic priority, otherwise, traffic rules should be inserted to allow the NIC to determine how to process these packets in terms of priority. The latter scenario is more typical for telecommunication operators where they can apply rules according to the 5G user subscription to determine how its network packets should be processed. It allows a differentiation between DSP (Digital Service Provider) and ISP (Infrastructure Service Provider) which is the main driver with the use of cloud computing. The scenario proposed allows ISP to insert rules in the network that are affecting not only ISP traffic (all traffic) but also specific to each of the DSPs, allowing new business models. In addition, it is noted that for the E2E QoS our proposed scheme is able to address key QoS metrics such as delay, packet loss and jitter end to end in an ISP domain. For multiple ISPs interconnected via Internet, the SLA in place between different ISPs will be enforced and is out of the scope of this paper.

Table 1 proposes a taxonomy of the various types of network slices and how they could be mapped to the existing lanes/queues of the NIC. The table indicates a name for the slice type together with the number of different ways of handling traffic supported by such a slice (Differentiated 5G User Traffic) and the mapping to the main 5G Use Cases (eMBB, URLLC, and mMTC) defined by ITU.

5 | PROTOTYPING PLATFORM AND IMPLEMENTATION

5.1 | P4-NetFPGA-based Reference Framework

The prototype implemented to test, validate and evaluate the proposed network slicing framework is based on NetFPGA SUME^{39,40}, an open hardware development platform for networking devices as mentioned. The platform has been configured using the P4 language³⁵, as in the P4-NetFPGA project⁴¹. The NetFPGA SUME includes four 10Gbps physical interfaces and one Gen 3 PCIe with RX and TX channels. The P4-NetFPGA project is based on the SimpleSumeSwitch architecture, which includes a processing pipeline generated from a P4 description with Xilinx SDNet, with a maximum processing rate of 51.2Gbps. The packets arriving at the five interfaces are transferred to the processing pipeline using a RR (Round Robin) based input arbiter. An output arbiter distributes the resulting packets to the selected interfaces. Each interface has an input queue (RX queue) and an output queue (TX queue) to accommodate high rate packet bursts. The PCIe interface is controlled by the RIFFA (Reusable Integration Framework for FPGA Accelerators) module, which implements DMA data transactions between the network card

TABLE 1 Proposed Types of Slices for 5G Users

Slice Type	# Differentiated 5G User Traffic	Priority (Queue/Lane)	Possible Use Cases
Best Effort	1	0	Enhanced Mobile Broadband / Massive MTC
Best Effort Reliable	1	1	Enhanced Reliable Mobile Broadband / Massive MTC
Low Latency	1	2	Low-Latency Communications
Low Latency Reliable	1	3	Low-Latency Reliable Communications
Ultra-Low Latency Reliable	1	4	Ultra-Low Latency Reliable Communications
Ultra-Low Latency High-Reliable	1	5	Ultra-Low Latency High Reliable Communications
Critical Communications	1	6	Critical Communications
Network Control	1	7	Reserved to Control and Management Network
Dual Best Effort + Low Latency	2	0/2	Enhanced Mobile Broadband / Massive MTC / Low-Latency Communications
Dual Best Effort + Low Latency Reliable	2	1/3	Enhanced Reliable Mobile Broadband / Massive MTC / Low-Latency Reliable Communications
Dual Best Effort + Ultra-Low Latency	2	0/4	Enhanced Mobile Broadband / Massive MTC / Ultra-Low Latency Reliable Communications
Dual Best Effort + Ultra-Low Latency Reliable	2	1/5	Enhanced Reliable Mobile Broadband / Massive MTC / Ultra-Low Latency High Reliable Communications
Dual Best Effort Reliable + Critical Communications	2	1/6	Enhanced Reliable Mobile Broadband / Massive MTC / Dedicated Ultra-Low Latency Channel
Triple Best Effort + Low-Latency + Ultra Low Latency	3	0/2/4	Enhanced Mobile Broadband / Massive MTC / Low-Latency Communications / Ultra-Low Latency Reliable Communications
Triple Best Effort + Low-Latency + Ultra Low Latency Reliable	3	1/3/5	Enhanced Reliable Mobile Broadband / Massive MTC / Low-Latency Reliable Communications / Ultra-Low Latency High Reliable Communications
Flexible Slice	4	0/2/4/6	Any Communication
Reliable Flexible Slice	4	1/3/5/6	Any Communication with High Reliability

and the server. The RIFFA module included in the NetFPGA project is compliant with Gen 2 PCIe. The server side consists of a Linux Kernel driver.

FIGURE 4 P4-NetFPGA internal architecture

The framework proposed in this paper is based on developing an extension to the SimpleSumeSwitch architecture of the NetFPGA platform and introducing QoS control to the traffic processing. Figure 4 represents the SimpleSumeSwitch architecture of the NetFPGA. In this figure, the TX queue that connects the RR Output Arbiter with the DMA port is rolled with a red square, highlighting that it is a part of the structure modified in this paper, as explained in the next subsection. Furthermore, the metadata structure of the P4-NetFPGA is modified, by adding two fields used to route a packet to the corresponding queue. A two-level set of queues has replaced the output queue connected to the DMA module, which implies a bottleneck in the data path. The reason why the design is built on the extension of that concrete part of the architecture is based on the fact that the rest of queues are virtually never filled since the hardware clocking is able to process at line speed the incoming traffic from the physical interfaces and then the main bottleneck happens when the Linux kernel driver is able to process packets at lower speed than the NIC is able to send to the memory using DMA. This exactly stresses the TX queue that connects the RR Output Arbiter with the DMA port.

5.2 | 5G Slicing on modified P4-NetFPGA

FIGURE 5 Developed 5G Network Slicing Architecture based on NetFPGA-based framework

Figure 5 depicts the extension designed in the TX queues connected to DMA in order to implement 5G network slicing support. The first level queues distribute the packets into 8 different paths, using a 3-bit field named RR as an index. Each path includes a second level of 4 queues, where packets are distributed according to their priority stated in a 2-bit field named PRIO. This introduces a total of 32 queues. At the design stage, the number of queues for the first level is determined by how many users are going to be hosted in the same machine simultaneously since the main purpose of these queues is to be able to distribute fairly the traffic among those belonging to the same slice type. The second level is related to the different lanes/queues that are available for each of the users. At a production level, 16 or 32 for the first level and 8 for the second level would be more appropriate values. 8 and 4 are those used in our NetFPGA prototype mainly due to the processing constraints in the hardware pipeline at these line rates.

For the dequeuing mechanism, each path includes an output register; in total there are 8 output registers. Each output register is filled with the first available packet of the non-empty queue with the highest priority. A final output register is filled with available packets from the different paths according to a round robin policy.

These are hardware modifications implemented in Verilog that we introduce in a modified architecture that is used in this paper in the P4-NetFPGA framework. This new architecture offers a structure to implement QoS and slicing in the data path using a P4 program.

This proposed architecture is then applied in the P4-NetFPGA workflow to implement a network card with 5G slicing support, using only one Ethernet port and the DMA port to connect to a server. A P4 program sets the values of the RR and PRIO fields of the metadata that will be used to enqueue the packets sent to the server through the implemented two-level queue set. A ternary key-value table is included in the P4 program to insert rules to control the data path of the packets, assuming a non-trusted scenario such as the one in a telecommunication operator. It is implemented using a Ternary Content Addressable Memory (TCAM). TCAM allows an application to query and store data using 3 different input values: 0, 1 and X, being “X” the value used as a “wildcard”. The structure of the table (fields used as input keys and available actions) are fixed once deployed, while the content of the table can be modified on-line to add new rules.

A P4 program includes three stages: parser, match/action and deparser. The parser can manage multiple structures of packets, providing support to detect 5G multi-tenant flows, as described in Section 3, with multiple encapsulation layers. In the parser stage, the flows arrive as incoming network packets at the pipeline and the different headers and fields of these packets are analyzed and processed. As shown in Figure 4, the parser stage is directly connected to a round robin Input arbiter that lets packets go into the P4 parser. The P4 program developed can extract the value of inner and outer fields, including the multiple IP headers and transport layers. This prototype implementation considers the values of the innermost IP and transport headers to classify the 5G flows in different slices and priority levels. It has been intentionally decided to use the port and not the DSCP value in order to see that the differentiation of the traffic requirements can be realized at the flow level as well.

Once the packet is parsed, all metadata are extracted from the packet and thus shared with the match/action stage. This metadata information is introduced into the match/action stage, where all these metadata are matched against the set of rules defined to establish the value of the metadata fields. This stage is where the TCAM table and Algorithm 1 have been developed. The last stage is the deparser, where the packet is formed and sent to the output queues programmed according to the slice type inserted in the rule set.

Algorithm 1 Match/Action pseudo-algorithm implementation for the management of the P4-NetFPGA-based queues developed

```

if FPGA.srcPort == physicalPort then
  if table.hit then
    PRIO=actionRule.prio
    RR=actionRule.rr
  else
    PRIO = packet.innerIPheader.DS(2:1)
    RR = packet.innerIPheader.srcAddr(2:0)
  end if
  FPGA.dstPort = DMA;
else
  if FPGA.srcPort == DMA then
    FPGA.dstPort = physicalPort
  end if
end if

```

Algorithm 1 describes the implementation carried out to control the NetFPGA queues developed in this framework. This algorithm sends incoming packets from physical ports to the DMA channel, and packets that come from DMA to the physical interfaces. For packets coming from physical ports, extra treatment is applied. In this case, the default action is applied when the table is empty or the packet does not match any of the rules. The default action uses fields from the innermost IP header to set the value of PRIO and RR fields. The DSCP field is used for the PRIO field while the source address is used for the RR

field. As shown in Table 2, from the 6 bits of the DSCP, the P4 program selects the bits 2-1 for the 2-bit PRIO field. The 3-bit RR field is set with the least significant bits of the source IP address, used to group users into the same slice. At a production grade, this can be managed by the internal DHCP (Dynamic Host Configuration Protocol) management in order to make a smart assignment of IP addresses based on the type of subscription of the user.

If there is a match in the key-value TCAM table, the PRIO and RR fields are set according to the corresponding action to select the QoS queue. The key used has 83 bits by joining four inner 5G User IP header fields: source and destination address, the destination port and the DSCP field. The action indicates the values for the PRIO and RR fields. The grouping strategy of users into different slices is modified online using the table. For example, if there is a scenario with multiple slices, the control algorithm considers classifying using 3 bits from the subnet prefix in order to avoid mixing users from different subnets. Using the ternary values in the table provides flexibility to the grouping strategy at runtime. Ignoring some of the bits of the key (by using wild cards in the rule) enables the application of the same queue policy to all the traffic corresponding to the same slice. It is important to remark that values matched in the TCAM table are related to the most inner headers, which represent the UE. This is a significant enhancement in the classification capabilities of the network card in order to be 5G-compliant, which is a significant step with respect to the state of the art.

Figure 5 depicts the structure of the resulting data path, focusing on the structure of the table that contains the rules and the modified output queue of the DMA interface. In this figure, it can also be appreciated how the RR and PRIO queues have been developed internally, using demultiplexers and multiplexers between them to route the incoming packets.

The prototype network card is customized to process 5G multi-tenant flows in different scenarios, where the end users are grouped in different slice instances to guarantee their QoS with low impact from other slices. We consider slice instances by varying the IP subnet for the sake of prototyping simplicity, from 1 to 16, each of them with 32 users generating UDP traffic to 16 different destination ports in order to represent 16 different services, with a DSCP value fixed for each port using a round-robin assignment 0-7 per each of the destination ports. Table 2 shows the values used for the generated traffic and the assigned priorities to these flows. In the proposed solution, queue 0 represents the queue with the lowest priority and queue 3 has a higher priority. It is noted that the programmability shown in Table 2 may fit the slice type "Flexible Slice" previously presented in Table 1, with the adoption of the addressing scheme of the queues to fit the 4 queues available in the prototype.

TABLE 2 Association between inner destination port, inner DSCP and prio queue.

Inner UDP destination port	DSCP (6 bits)	Prio queue
5000, 5002, 5004, 5006	xxx00x	0
5008, 5010, 5012, 5014	xxx01x	1
5016, 5018, 5020, 5022	xxx10x	2
5024, 5026, 5028, 5030	xxx11x	3

6 | EMPIRICAL TESTING

6.1 | EXPERIMENTAL SETUP

The experimental testbed developed to carry out the empirical testing of the proposed network slicing framework is shown in Figure 6. For the development of this experimental environment, Edge and Core networks have been deployed in two separate computers with the same hardware specifications: Dell T5810, Intel Xeon CPU E5-2630 v4, 32768 MB RAM and 512 GB SSD. With regard to network cards, the Edge is composed by two Gtek MLX-6801-1S 10GbE NIC with 2 x 10GbE Interfaces and the Core by only one with the same characteristics. In terms of software specifications, both computers have an operating system Ubuntu 16.04.01 with a Linux kernel 4.10.0-42. Core also has a NetFPGA SUME with 4 x 10GbE Interfaces and three virtual machines, deployed with Virtual Box, where UPF, AMF/SMF and UDM/AUSF are allocated. The CU uses OpenAirInterface⁴² and is developed in a Linux Container. The DU is deployed using Ettus B210⁴³ Software-Defined Radio, and SIM Cards from

SYSMOCOM⁴⁴ are used in the UE. This NetFPGA has been programmed with a new pipeline that allows the slicing in the communication between the board and the user level, thanks to the communication through the PCI port. For the development of the slicing in the NetFPGA, three new cores have been implemented, replacing a single FIFO queue implemented by default in the NetFPGA framework. For this experiment, a new NetFPGA bitstream has been developed, using the cores previously mentioned.

FIGURE 6 Testbed for the experimental setup over the 5G architecture presented.

For setting up the experimental platform, it is needed to execute a script with the rules to manage the network traffic between the different queues. This script introduces rules in the TCAM table of the NetFPGA (see 1 in Figure 6). The maximum number of rules inserted by this script is 64 being used for the bigger case, where the NetFPGA is receiving traffic from 16 different simultaneous slices with 32 users per slice and each user sending traffic for 16 different services belong to 8 different QoS requirement. After rules are loaded, two tcpdumps are started in the sender interface of the Core (see eth1 of Figure 6) and in the DMA reception interface (see nf0 of Figure 6). The pcaps collected by these tcpdumps running in the Linux kernel will be used later on used to acquire the communication statistics between Edge and Core. Once tcpdumps are launched, pcaps are sent with tcpdump (see 2 in Figure 6). For this experiment, five different pcaps files have been utilized representing different scenarios. They are described in Table 3. These pcaps are sent separately, with the idea of testing the scalability of the solution proposed.

For each of the pcaps files sent, the traffic is received by the interface eth2 of the network card situated in the Edge segment. This traffic is forwarded to the other network card (see 4 in Figure 6), situated in the same network segment, and it is resent to the Core segment through the eth1 interface (see 5 in Figure 6). NetFPGA programmed with the proposed slicing framework receives the traffic via the nf0 interface. Different actions will be applied over this traffic. When the traffic is crossing the pipeline of the NetFPGA, these actions will determine the queues that will be used to send the traffic to the Linux operating system, using the NetFPGA driver. All traffic received is captured and save in a pcap file by tcpdump (see 6 in Figure 6). Although not been tested in this research work, it also supports other actions, such as setting the DSCP value of the IP header or, in encapsulated traffic, copying the DSCP value from the inner IP header to the outer one. These actions allow signalling between different hops of the network in order to provide the enabler to allow the E2E networking slicing.

After the transmission of the 5 pcaps described in the Table 3, 10 pcaps are ready in the Core segment to be analyzed, 5 of them with the transmission packets (captured in step 3) and the other 5 with the received packets (captured in step 6). These 10 pcaps are transformed to CSV files (see 7 in Figure 6) for statistics. Transmitted pcaps and received pcaps will be compared (see 8 in Figure 6) to obtain the delay, jitter and packet loss statistics in the 5 different scenarios represented by the 5 pcaps

TABLE 3 Pcaps modified to test the slicing in the experimental setup proposed.

slice	IPs(users) per pcap	Flows/Services	# packets	size per packet	pcap size (Gb)
1	32	512	477696	1500	5,73
2	64	1024	955392	1500	11,46
4	128	2048	1910784	1500	22,93
8	256	4096	3821568	1500	45,86
16	512	8192	7643136	1500	91,72

transmitted. Nowadays, there is no other solution tested in a 5G Edge-Core segment which allows QoS-aware and performance isolation for 5G multi-tenant traffic, so results obtained have been compared with the 5G network performance values expected for different use cases. This comparison and the empirical values obtained are described in the following subsection.

Following enumerate summarizes the different steps shown in Figure 6 :

1. Network Administrator inserts the rules in the NetFPGA to establish the performance indicators of each slice (priority, bandwidth, jitter).
2. Pcap files are sent from the core to the edge.
3. Traffic leaving the core computer is kept for getting results and make the graphs.
4. Network traffic is forwarded to another network card in the edge computer.
5. Traffic is sent back again from the edge to the core.
6. Traffic received in the core is kept in a pcap file.
7. Pcap files obtained in the transmission (step 3) and in the reception (step 6) are transformed into CSV files.
8. CSV files are compared and analysed to obtain the final graphs shown in Section 6.2

6.2 | RESULTS

This section presents the empirical validation of the 5G P4-NetFPGA framework proposed in previous sections. The main aim of the solution developed is to guarantee the QoS of the users that are within the same slice and at the same time the QoS between different slice instances. Thus, to warranty low-latency and reliability services of users who have a high priority associated in scenarios where traffic congestion exists due to the high number of data volume that should be processed by the NetFPGA card at the same time. To validate the efficiency and scalability of the solution developed, three different graphs are represented in this paper. These charts present statistics about delay, jitter and packet loss statistics when the NetFPGA is applying different QoS configurations to the traffic stemming from up to 512 users. Table 3 shows the settings of the numbers traffic senders, the quantity of data sent in each slice and more information about the traffic. The bars presented in each graph are the average of all the packets that are involved in each of the flows used in the experiment (per each service). In fact, the experiment has been carried out 5 times and the values are the average of those 5 executions.

Figure 7 shows the delay achieved when network slicing is applied to the traffic transmitted by up to 512 users. The delay specifies how long it takes for a packet to travel across the network from one node to another and return back. The number of users sending traffic in each slice is always 32. Thus, 32 users are sending traffic when only 1 *Flexible Slice* instance is configured and 512 users when 16 slices are configured (32 users in each slice). The graph shows an exponential trend in the number of slices. It is clear that overall the delays measured in the five scenarios are fairly similar, always around 2ms, in the queues that have been assigned the lowest priority (queue 0), associated to Best Effort traffic. Nevertheless, the delays are reduced dramatically when the traffic processed by queues that have been assigned higher priorities. With the increase of the priority, delays decrease. The importance is that different scenarios with different slices show exactly the same performance in terms of delay. It can be observed that packets processed by the queues 1, 2 and 3 have a maximum delay of 0.25 ms; in the same manner, the delays reached by queue 3 are always lower than 0.1ms. The chart represents the standard deviations of the delay in each case between

FIGURE 7 Delay achieved when Number of Flexible Slice Instances are increased

all the packets submitted; the values are not, in any case, higher than 0.1ms. Overall, these comparable delay results across different cases reveal high scalability of the proposed scheme in terms of the number of users per slice. Noticed that the NGMN Alliance⁴⁵ has defined two relevant 5G KPIs in terms of maximum latency allowed, which are 10ms for end to end traffic in general and 1ms for extremely low latency traffic. In the worst case, shown in Figure 7, the delay reached is close to 2,5ms when the network traffic is processed by the lowest priority queues, however, values obtained with the traffic sent by the rest of the queues is always under 1ms. It means that the solution proposed in this paper is totally under the 5G KPIs in terms of latency. Also, notice how Figure 7 demonstrates traffic isolation between different slices and at the same time there is DiffServ inside of each of such slices.

FIGURE 8 Packet loss achieved when Number of Flexible Slice Instances are increased

The bar chart shows in Figure 8 presents the number of packet loss in each queue when the NetFPGA is working in a congestion scenario. Packet loss refers to packets that fail to reach their destination. Queue 0 has the worst values concerning delay, and it also has the highest rate of packet loss. These packets are discarded in the NetFPGA RIFFA driver when it is working under a stressful circumstance. Again, as can be observed the solution proposed scales well, as the packet loss rate when only

32 users are sending traffic is quite similar to that when 512 users are transmitting. It is clear that, in queue 0 (in the service with port 5000), almost 100% of packets are discarded when they are congestion in the network, mainly for the performance of the software stack. However, the traffic with a higher QoS, transmitted by the others queues, is arriving with less than 1% of packets loss in queues 1 and 2 and with 0% of packet loss in queue 3, which has the maximum priority. Also, notice how Figure 8 demonstrates the same performance between different network slices and at the same time there is DiffServ inside of each of such slices.

FIGURE 9 Jitter achieved when Number of Flexible Slice Instances are increased.

Figure 9 represents the jitter results in the communication between the NIC that is sending the traffic and the NetFPGA, which is receiving it. The jitter describes the variation in the delay of received packets. Although the jitter increase when there is congestion in the NetFPGA card, the graph shows a high stability in the jitter performance for those services that are assigned to queues different to queue 0 which are those services sensitive to low-latency and thus also to jitter variations. Again, notice how Figure 9 demonstrates the same performance between different network slices.

In terms of latency, packet loss and jitter, Ericsson provided a detailed explanation of the 5G network performance requirements needed for different use cases⁴⁶. Table 4 presents a summary of these. As is shown, in terms of latency, all values are over 1ms except for “Factory cell automation” where less than 1ms of delay is required. The solution proposed in this paper is able to achieve values below 1ms, in a congestion situation. In terms of reliability, Ericsson also proposes demanding values, the majority under 1% of packet loss. The solution proposed is able to achieve ambitious packet loss values, close to 0%, when traffic is sent through the most priority queues. Thus, this solution is able to work in the different use cases proposed due to the fact that it achieves the latency and packet loss required. Although the jitter is an important measure in terms of QoS, mainly for VoIP and video communications, there is no value proposed by none organisation, however, it is known that this value should be under 30ms for VoIP and video transmissions⁴⁷. Notice that the values of jitter obtained in this contribution are always below 1ms when traffic is sent through the lowest priority queue.

The QoS-aware solution for 5G MEC scenarios proposed in this contribution allows the programmability of the communication parameters, such as delay, packet loss and jitter in the data path. Table 5, shows the maximum values which would be achieved under a congestion scenario when up to 512 users are transmitting network traffic at the same time. This values can be shown in Figures 7, 8 and 9, corresponding with the peaks reached in each queue with the higher number of users and slices. It demonstrates that with a proper data path programmability, the values of delay packet loss and jitter can be ensured in the communication over a 5G MEC scenario.

TABLE 4 5G KPIs required by Ericsson for different use cases⁴⁶.

Use case	Latency	Packet loss
Autonomous vehicle control	5ms	0,001%
Factory cell automation	Below 1ms	Below $10^{-9}\%$
Media on demand	5s to start, 200ms after link interrupts	5%
Tele-protection in smart grid networks	8ms	0,001%

TABLE 5 Maximum value of delay, packet loss and jitter ensured under a congestion scenario, where up to 512 users are transmitting traffic.

Slice	Delay (ms)	Packet Loss (%)	Jitter (ms)
0	2,3624	88,97	0,94003
1	0,25178	0,19632	0,02468
2	0,20891	0,19601	0,01938
3	0,13805	0,19552	0,01487

7 | CONCLUSIONS

Network slicing is a major paradigm shift from the current 4G networks to the emerging 5G networks. To meet the diverse SLA requirements for network slice based services, QoS-aware network slicing is essential in 5G and beyond networks. This paper has proposed a QoS-controllable network slicing framework by exploring data plane programmability in a hardware and software hybrid approach. The architectural and functional designs of the proposed framework have been described in details, with a new set of network slice types proposed. The prototype of the framework leverages and extends a latest open-source data plane networking platform NetFPGA. The empirical results have validated the effectiveness of the proposed priority-queues based QoS control for network slices of diverse QoS requirements. Moreover, the comparable and good performance results in terms of delay, packet loss rate and jitter across various scenarios (including best and worst cases e.g., in terms of user numbers) clearly show the high scalability of the proposed design and prototype.

Future work will seek to integrate this programmable data plane with a 5G infrastructure platform especially a Mobile/Multi-access Edge Computing platform, and further evaluate the performance in such an integrated system. Furthermore, a network slicing control plane is to be designed and implemented to operate over the 5G data plane to configure and optimize the QoS at runtime in order to enable dynamic reconfiguration and optimization of 5G infrastructure and network slice based services.

ACKNOWLEDGMENTS

This work was funded by the European Commission Horizon 2020 5G-PPP Program under Grant Agreement Number H2020-ICT-2016-2/761913 (“SliceNet: End-to-End Cognitive Network Slicing and Slice Management Framework in Visualized Multi-Domain, Multi-Tenant 5G Networks”), and by the Spanish Ministry of Economy and Competitiveness under contract RTC-2016-5434-8.

References

1. Mohr Werner. *Telecom industry and European academia join forces to develop a multiservice mobile network architecture for the 5G era.* <https://5g-ppp.eu/telecom-industry-and-european-academia-join-forces-to-develop-a-multiservice-mobile-network-architecture-for-the-5g-era/>. Accessed Jun 14, 2018; 2015.

2. Alliance NGMN. Description of network slicing concept. *NGMN 5G P.* 2016;1.
3. 3GPP . System Architecture for the 5G System. *Reference: 23.501. Released 15.* 2017;.
4. Wang L., Geng L.. Review of Network Slicing Research in Diverse Standards draft-wang-nfvrg-network-slice-diverse-standards-00 <https://tools.ietf.org/html/draft-wang-nfvrg-network-slice-diverse-standards-00#page-3>[Online; accessed 09-Dec-2018]; .
5. ETSI . Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework V3.1.1 (2017-12) https://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf[Online; accessed 09-Dec-2018]; .
6. Tony Saboorian Laurent Thiebaut. *Network Slicing and 3GPP Service and Systems Aspects (SA) Standard.* <https://sdn.ieee.org/newsletter/december-2017/network-slicing-and-3gpp-service-and-systems-aspects-sa-standard>. Accessed Jun 14, 2018; 2017.
7. Pignataro Carlos, Halpern Joel. *Service Function Chaining (SFC) Architecture.* 2015.
8. ETSI . *Network Functions Virtualisation(NFV); Management and Orchestration.* ETSI TR, GS NFV 003, v1.1.1: ETSI; Dec. 2016.
9. *5G-NORMA - 5G Novel Radio Multiservice adaptative network Architecture.* <https://5g-ppp.eu/5g-norma/>. Accessed Jun 14, 2018; 2015.
10. *SONATA NFV: Agile Service Development and Orchestration in 5G Virtualized Networks.* <http://sonata-nfv.eu/content/about-sonata>. Accessed Jun 14, 2018; 2015.
11. *Speed 5G - quality of Service Provision and capacity Expansion through Extended-DSA for 5G.* https://speed-5g.eu/wp-content/uploads/2015/06/speed-5g_project_factsheet_v0.6.pdf. Accessed Jun 14, 2018; 2015.
12. Alexandros Kostopoulos Deng Junqua Dorin Panaitopo Fang-Chun Kuo Kostas Katsalis Navid Nikaein Mariana Goldhamer Tao Chen Rebecca Steiner Roberto Riggio. *System Architecture and Abstractions for Mobile Networks.* Deliverable D2.2: 5GPP; Jul. 2016.
13. An Xueli, Zhou Chan, Trivisonno Riccardo, et al. On end to end network slicing for 5G communication systems. *Transactions on Emerging Telecommunications Technologies.* 2017;28(4).
14. Ravindran Ravishankar, Chakraborti Asit, Amin Syed Obaid, Azgin Aytac, Wang Guoqiang. 5G-ICN: Delivering ICN services over 5G using network slicing. *IEEE Communications Magazine.* 2017;55(5):101–107.
15. Rost Peter, Mannweiler Christian, Michalopoulos Diomidis S, et al. Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Communications magazine.* 2017;55(5):72–79.
16. *5G-Crosshaul - the integrated fronthaul/backhaul.* <http://5g-crosshaul.eu/>. Accessed Dec 12, 2018; 2015.
17. Li Xi, Casellas Ramon, Landi Giada, et al. 5G-crosshaul network slicing: Enabling multi-tenancy in mobile transport networks. *IEEE Communications Magazine.* 2017;55(8):128–137.
18. Zhang Haijun, Liu Na, Chu Xiaoli, Long Keping, Aghvami Abdol-Hamid, Leung Victor CM. Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine.* 2017;55(8):138–145.
19. Mayoral Arturo, Vilalta Ricard, Casellas Ramon, Martinez Ricardo, Munoz Raul. Multi-tenant 5G network slicing architecture with dynamic deployment of virtualized tenant management and orchestration (MANO) instances. In: :1–3VDE; 2016.
20. Brown Martin A.. *Traffic Control HOWTO Verion 1.0.2.* <https://www.tldp.org/HOWTO/pdf/Traffic-Control-HOWTO.pdf>. Accessed Jun 14, 2018; 2016.

21. Intel . *Intel 82599 10 Gigabit Ethernet Controller: Datasheet*. <https://www.intel.la/content/www/xl/es/embedded/products/networking/82599-10-gbe-controller-datasheet.html>. Accessed Jun 14, 2018; Mar 2016.
22. Sivaraman Anirudh, Subramanian Suvinay, Agrawal Anurag, et al. Towards Programmable Packet Scheduling. In: HotNets-XIV:23:1–23:7ACM; 2015; New York, NY, USA.
23. McKeown Nick, Anderson Tom, Balakrishnan Hari, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*. 2008;38(2):69–74.
24. Intel DPDK. *Data plane development kit*. <http://dpdk.org/>. Accessed Jun 14, 2018; 2014.
25. Project IO Visor. *XDP - eXpress Data Path*. <https://www.iovisor.org/technology/xdp>. Accessed Jun 14, 2018; 2016.
26. Projects Linux Foundation. *IO Visor Project*. <https://www.iovisor.org/>. Accessed Jun 14, 2018; 2016.
27. Deri Luca, others . Improving passive packet capture: Beyond device polling. In: :85–93Amsterdam, Netherlands; 2004.
28. Salva-Garcia Pablo, Alcaraz-Calero Jose M, Alaez Ricardo Marco, Chirivella-Perez Enrique, Nightingale James, Wang Qi. 5G-UHD: Design, prototyping and empirical evaluation of adaptive Ultra-High-Definition video streaming based on scalable H. 265 in virtualised 5G networks. *Computer Communications*. 2018;118:171–184.
29. Mysirlidis C., Lykoyrgiotis A., Dagiuklas T., Politis I., Kotsopoulos S.. Media-aware proxy: Application layer filtering and L3 mobility for media streaming optimization. In: :6912-6917; 2015.
30. Kourtis Michail-Alexandros, Koumaras Harilaos, Xilouris George, Liberal Fidel. An NFV-based video quality assessment method over 5G small cell networks. *IEEE MultiMedia*. 2017;24(4):68–78.
31. Mamolar Ana Serrano, Pervez Zeeshan, Calero Jose M Alcaraz, Khattak Asad Masood. Towards the transversal detection of DDoS network attacks in 5G multi-tenant overlay networks. *Computers & Security*. 2018;79:132–147.
32. *Snort: free open source network intrusion detection system (IDS) and intrusion prevention system(IPS)*. <https://www.snort.org/>. Accessed Mar 11, 2019; 2019.
33. Lockwood John W, McKeown Nick, Watson Greg, et al. NetFPGA—an open platform for gigabit-rate network switching and routing. In: :160–161IEEE; 2007.
34. Martinek Tomas, Kosek Martin. Netcope: Platform for rapid development of network applications. In: :1–6IEEE; 2008.
35. Bosshart Pat, Daly Dan, Gibb Glen, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*. 2014;44(3):87–95.
36. Ricart-Sanchez Ruben, Malagon Pedro, Salva-Garcia Pablo, Perez Enrique Chirivella, Wang Qi, Calero Jose M Alcaraz. Towards an FPGA-Accelerated programmable data path for edge-to-core communications in 5G networks. *Journal of Network and Computer Applications*. 2018;124:80–93.
37. Ricart-Sanchez Ruben, Malagon Pedro, Alcaraz-Calero Jose M, Wang Qi. Hardware-Accelerated Firewall for 5G Mobile Networks. In: :446–447IEEE; 2018.
38. Kim Junseok, Kim Dongmyoung, Choi Sunghyun. 3GPP SA2 architecture and functions for 5G mobile communication system. *ICT Express*. 2017;3(1):1–8.
39. Zilberman Noa, Audzevich Yury, Covington G Adam, Moore Andrew W. NetFPGA SUME: Toward 100 Gbps as research commodity. *IEEE micro*. 2014;34(5):32–41.
40. Organization NetFPGA GitHub. *NetFPGA-SUME*. <https://github.com/NetFPGA/NetFPGA-SUME-public/wiki>. AccessedJun14,2018; 2017.
41. Organization NetFPGA GitHub. *P4-NetFPGA*. <https://github.com/NetFPGA/P4-NetFPGA-public>.AccessedJun14,2018; 2017.

42. Interface Open Air. Software-defined 5G System. Cloud RAN (C-RAN) http://www.openairinterface.org/?page_id=466[Online; accessed 08-Dec-2018]; .
43. Ettus . USRP B210 Board <https://www.ettus.com/product/details/UB210-KIT>[Online; accessed 08-Dec-2018]; .
44. sysmocom . sysmoUSIM-SJS1 SIM + USIM Card <http://shop.sysmocom.de/products/sysmousim-sjs1>[Online; accessed 08-Dec-2018]; .
45. Alliance NGMN. 5G white paper. *Next generation mobile networks, white paper*. 2015;:1–125.
46. Ericsson . *Ericsson White Paper UEN 284 23-3251 rev B*. <https://www.ericsson.com/assets/local/publications/white-papers/wp-5g-systems.pdf>.AccessedDec6,2018; 2017.
47. Datapath . *What is Acceptable VoIP Jitter?*. <https://datapath.io/resources/blog/what-is-acceptable-voip-jitter/>.AccessedDec6,2018; 2016.

AUTHOR BIOGRAPHY

Ruben Ricart-Sanchez is a PhD candidate at the University of the West of Scotland, United Kingdom, where he is actively involved in the EU H2020 5G-PPP Phase II SliceNet project. His main research interests include reconfigurable hardware, programmable network control plane, configurable hardware management plane, mobile edge computing data planes, cognitive network management and 5G mobile networks, among others.

Pedro Malagon obtained a PhD in Telecommunication Engineering by Universidad Politecnica de Madrid, Spain, in 2015. He is an assistant professor since 2010 in UPM. His professional interests are mainly reconfigurable hardware and security and energy aware computing.

Antonio Matencio-Escolar is a PhD candidate at the University of the West of Scotland, United Kingdom. He is actively involved in the EU H2020 5G-PPP Phase II SliceNet Project. His main research interests include 5G Network Slicing, Software Defined Networks(SDN), Network Function Virtualization (NFV) and Software Datapaths, among others.

Jose M. Alcaraz-Calero is a Full Professor in networks and security at the University of the West of Scotland, and he is the technical co-coordinator of the EU H2020 5G-PPP Phase I SELFNET and Phase II SliceNet. His professional interests include network cognition, management, security and control, service deployment, automation and orchestration, and 5G mobile networks. Alcaraz Calero has a PhD in computer Science, University of Murcia.

Qi Wang is a Full Professor at the University of the West of Scotland, and he is the technical co-coordinator of EU H2020 5G-PPP Phase I SELFNET and Phase II SliceNet. His research primarily focuses on 5G mobile networks. Wang has a PhD in mobile networking from the University of Plymouth, UK.