A Proof of Concept for QoS-Oriented 5G Network Slicing with eBPF-TC

Ekarani Silvestre, Rian Melo, Ednaldo Junior, Ruan D. Gomes and João Brunet

***Abstract*— This paper presents a proof of concept on dynamically slicing a 5G network using TC and eBPF for granular Quality of Service (QoS). The approach ensures appropriate bandwidth for each slice based on near real-time traffic inspection and classification. eBPF handles packet inspection and classification using the Hierarchical Token Bucket (HTB) queuing discipline for bandwidth adjustment. By enabling reactive adjustments, eBPF provides granular control based on network events, essential for agile Quality of Experience (QoE) monitoring. The PoC considers four User Equipments (UEs) with throughput assessment and performance monitoring using FlexRIC. The observed slice activation and deactivation response times were 0.37s and 0.05s, respectively.**

***Keywords*— 5G, eBPF, Network Slicing, QoS**

I. Introduction

The fifth generation (5G) mobile network architecture brings a set of advances toward a more flexible network, with enhanced resources to support diverse performance requirements, such as enhanced Mobile Broadband (eMBB), Massive Machine-Type Communications (mMTC), and Ultra-Reliable Low Latency Communications (URLLC), needed for a wide range of applications [1][2]. To support this diversity in a single physical infrastructure, network slicing emerges as a key strategy, in which hardware and software resources are divided into logically separated slices that map different Quality of Service (QoS) requirements. In this dynamic scenario, the network must be efficiently reconfigured as the demands for resources change over time [3]. In this context, research developing solutions that enable efficient and low latency slicing is essential, with efforts emerging to reduce control latency and provide real-time system control [4].

One of the technologies that can be used to implement network slicing solutions for 5G networks is the extended Berkeley Packet Filter (eBPF). eBPF is a kernel programming paradigm that enables efficient in-kernel execution of user-defined programs without operating system source code modifications. This capability reduces processing overhead while enhancing packet processing efficiency, which is critical for 5G and 6G networks, where ultra-low latency and high performance are fundamental requirements. eBPF programs enable significant simplification of the kernel networking stack and reduce latency between endpoints, creating more direct communication paths through customized packet processing [5].

Ekarani Silvestre, UFCG e-mail: ekarani.silvestre@ccc.ufcg.edu.br; Rian Melo, UFCG, e-mail: rian.melo@ccc.ufcg.edu.br; Ednaldo Junior, IFPB, e-mail: ednaldo.junior@lsd.ufcg.edu.br; Ruan Gomes, IFPB, e-mail: ruan.gomes@ifpb.edu.br; João Brunet, UFCG, e-mail: joao.arthur@computacao.ufcg.edu.br. This work was funded by FIT - Instituto de Tecnologia/IBM (EMBRAPII PCEE-2310.0243). The authors also thank the support from CNPq (305536/2021-4).

Furthermore, eBPF has been widely explored in virtual network management, particularly in Kubernetes environments through Container Network Interface (CNI) plugins like Cilium [6] and Calico [7]. Since 5G networks are virtualized, from the core network functions to the Radio Access Network (RAN), with virtual and software-defined radio, and often hosted in the cloud, adopting eBPF-based solutions can bring significant benefits in scalability, efficiency, and flexibility [5].

In this context, this work aims to present a Proof of Concept (PoC) using eBPF associated to the Traffic Control (TC) hook to dynamically provide network slices in a 5G network for provisioning varied QoS to different users in near-real-time, in response to network events. A key motivation for adopting an eBPF-based solution lies in its ability to perform deep packet inspection with minimal performance overhead. This capability is particularly relevant in 5G environments where user traffic is often encapsulated using protocols such as GTP (GPRS Tunnelling Protocol). Traditional tools like iptables or TC alone cannot access the inner IP headers of such tunneled traffic. In contrast, eBPF programs can parse these encapsulated packets directly within the kernel, without requiring costly packet redirection to user space.

Experiments were performed to evaluate the functioning of the mechanism and the latency to enable and disable the slices. The proposed solution demonstrates the feasibility of multiple QoS profiles coexisting within a single network infrastructure, facilitating traffic segmentation into isolated classes and ensuring adherence to the specific requirements of each data flow. The evaluation considered four User Equipment (UE) and the FlexRIC RIC (Radio Intelligent Controller) solution was used to monitor the RAN. The observed slice activation and deactivation response times were 0.37s and 0.05s, respectively.

The structure of this paper is as follows. Section II reviews the related literature, Section III describes the PoC architecture and the scenario considered, Section IV evaluates the scenario and describes the obtained results, and Section V provides the conclusion and future work.

II. Related Works

The work of [5] highlights the complexity of cloud-native 5G and 6G networks. The authors explore eBPF for observability, security, and network optimization, emphasizing its portability and low overhead. However, they note challenges such as programmability restrictions and the complexity of the eBPF verifier. To address these, they proposed Sauron, a platform for writing and loading custom eBPF programs. Their evaluation covers four cases: *(i)* transport layer monitoring, *(ii)* 5G protocol performance, *(iii)* cloud-native resilience, and *(iv)* energy estimation. The results confirm the feasibility of eBPF for 5G and 6G networks and its growing role in networking and security.

The authors in [8] propose a 6G network slicing architecture using eBPF to optimize packet processing and QoS management by distributing traffic across isolated slices. They address key data plane challenges, such as handling encapsulated traffic, kernel network stack limitations, and compatibility with future networks. They introduce an eXpress Data Path (XDP)-based approach that bypasses the kernel in the Netronome Flow Processor driver, enabling low-latency packet forwarding. In addition, they present a Data Plane Programming (DPP)-based application that simplifies network control and monitoring. The architecture leverages offloading via the Agilio CX Smart Network Interface Card (SmartNIC), integrating eBPF, XDP, and Address Family eXpress Data Path (AF_XDP) to accelerate communication while reducing kernel overhead. Experimental validation in a beyond-5G environment demonstrated reduced packet loss, lower CPU usage, and scalability.

The work of [9] describes the application of eBPF/XDP in the deployment of a Mobile Gateway (MGW) for 5G networks to integrate eBPF into edge environments, allowing efficient data processing and traffic control free from the demand for specialized resources of solutions such as DPDK (Data Plane Development Kit). Using rate limiting algorithms such as Token Bucket, Fixed Window Counter and Sliding Window, an MGW prototype in eBPF was built with modules for GTP encapsulation removal, routing and traffic policing. The results revealed that the eBPF presented performance equivalent to DPDK in uplink scenarios and superior to other kernel-based solutions with good scalability.

Another application of eBPF in 5G is presented in [10], which introduces a solution to address the challenge of load balance in communication between the RAN and the AMF (Access and Mobility Management Function). This communication occurs over SCTP (Stream Control Transmission Protocol) at the transport layer and NGAP (Next-Generation Application Protocol) at the application layer. Two primary components are used in the solution: a load balancing mechanism using XDP and eBPF, and the implementation of AMFs as micro-AMFs to meet specific requirements. eBPF is used to create maps within the XDP-SCTP load balancer, using the primary fields of SCTP to optimize communication between multiple services and diverse implementation requirements. This approach improves performance in load balancing while also providing DDoS mitigation.

The work described in [11] uses eBPF and XDP as an approach to improve packet processing efficiency and achieve high availability in network slicing. The proposed solution runs inside the Linux kernel using XDP, while the 5G core network functions are executed in the user space. This ensures a rapid response to network congestion and failure recovery. Experimental results demonstrate that 99.2% of slice availability is achieved, surpassing the value of 96% achieved without using the proposed framework. These results validate the feasibility of integrating eBPF/XDP for efficient low-latency slicing in 5G environments.

The work described in [12] implements a prototype based on Open Virtual Switch (OVS) to perform network slicing. The authors propose a software data plane architecture with QoS guarantees that allows dynamic adjustments on demand and at runtime. The solution was empirically validated, demonstrating its capability for network slice deployment and management. The prototype supports up to 8192 slices at 10 Gbps, ensuring performance isolation between flows. Even under stress, it maintains low latency (max 5.8 µs) and zero packet loss, demonstrating its suitability for 5G networks.

Most of the studies described in this section highlight efforts to integrate eBPF into the 5G infrastructure. Among them, [8] also proposed a solution for network slicing, but the described solution was implemented in a SmartNIC. In terms of functionality, the work described in [12] is the most similar to the work described in this paper, particularly due to the use of HTB to guarantee bandwidth and delay in different network slices. The main distinctions lies in the technologies used to implement the network slicing prototype and the experimental methodology. The solution described in [12] is based on OVS, while the solution described in this paper uses eBPF for packet inspection. Finally, in [12] only traffic generators were used to validate the prototype, while the present study uses a complete 5G softwarized testbed, including RAN, 5G Core and RIC.

III. Overview of the PoC

This section presents the PoC developed to demonstrate a network slicing scenario using the Linux kernel tools TC and eBPF. The main goal of the eBPF program is to inspect and classify packets, which are then forwarded by the TC to the appropriate classes, each with predefined average and maximum transfer rates, ensuring traffic shaping. The bee icon shown in Figures 1 and 2 is commonly used as a visual representation of eBPF. Typically, it indicates where an eBPF program is attached within the Linux kernel, such as in traffic classification or packet handling points.

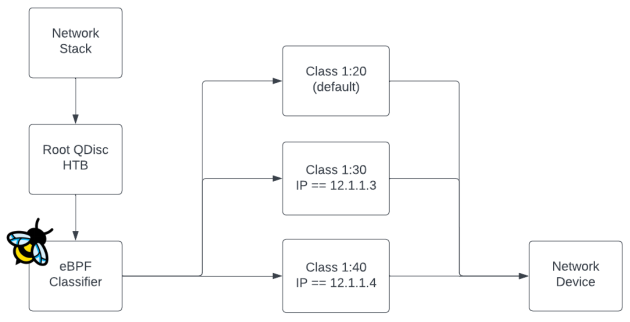

Fig. 1. Packet flow by the Linux traffic control. The bee icon represents eBPF, which plays a central role in the packet classification and handling.

The evaluated scenario considers an eMBB environment where different user profiles require varying bandwidth levels, justifying the use of network slicing. However, due to the lack of access to a real 5G testbed, the PoC was carried out in a test environment using a simulated RAN with a radio frequency simulator. In this setup, the slice allocation for privileged

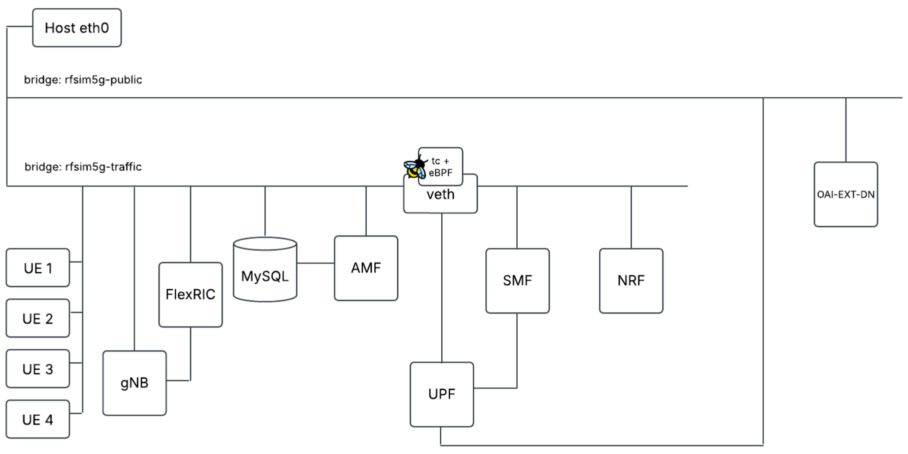

Fig. 2. Architecture of the experiment using containers. The bee icon in the diagram represents the use of eBPF in the traffic control mechanism.

users was based solely on the experienced throughput, without considering mobility aspects or channel condition variations in wireless communication.

Figure 1 shows the packet flow within the network stack. With traffic shaping applied, each packet is inspected by the eBPF program before reaching the network interface. Classification is based on predefined criteria, including packet headers and deeper-layer data, useful for GTP-encapsulated packets. eBPF provides two key hooks: XDP and TC. XDP runs in the NIC (Network Interface Card) driver for early packet handling, while TC works in the kernel traffic control layer on the RX and TX paths [13]. Despite being later in processing, TC enables advanced traffic shaping, making it the preferred choice for this work. It is important to clarify that the flow is processed in parallel for the three defined traffic classes, allowing simultaneous handling according to the assigned rules and bandwidth constraints.

Traffic shaping was chosen over policing because it queues packets to maintain minimum QoS through managed buffers. Policing mechanisms (e.g., ingress token bucket) can cause packet drops during congestion when tokens run out quickly. In contrast, traffic shaping buffers packets for controlled transmission, providing more predictable performance and QoS even under heavy load. The Hierarchical Token Bucket (HTB) queuing discipline was used to enable hierarchical bandwidth allocation. The *root qdisc* defines the total link bandwidth, while the child classes allocate portions of this bandwidth. Each HTB class corresponds to a network slice.

Figure 2 shows the containerized architecture of the test environment. The OpenAirInterface (OAI) 5G Core and the simulated RAN[1] were used. All elements of the 5G network, including Core, RAN, and FlexRIC, are connected to the internal network *rfsim5g-traffic*. Additionally, the User Plane Function (UPF) is also connected to *rfsim5g-public*, responsible for forwarding packets to the external data network (represented by the OAI-EXT-DN container). As shown in the figure, the eBPF program and the TC commands were applied to the virtual Ethernet (veth) interface created along with the UPF container, as this function handles packet routing. This choice was made to avoid accessing the container internally for traffic control actions. The containerized environment was hosted on a machine with 16GB of RAM, a 10-core 13th Gen Intel Core i7-1355U CPU, and Ubuntu 24.04 running kernel version 6.8.0-52-generic.

The experiment was carried out with four UEs and involved defining the QoS rules as follows, considering that the maximum aggregated throughput in the environment was 45 Mbps:

- UE 3 (priority), with IP 12.1.1.4, must maintain a minimum throughput of 20 Mbps. If the observed throughput drops below this threshold while the device is active on the network, the slicing mechanism is triggered;
- For the other UEs, as long as the priority UE is not using the network, there are no bandwidth restrictions, but no minimum throughput is guaranteed;
- When network slicing is triggered to ensure the priority UE's minimum throughput, one of the UEs is limited to a maximum of 10 Mbps, while the other two are restricted to a maximum of 2 Mbps each.

The prototype developed for this PoC included the execution of the FlexRIC KPM xApp and a Bash script to automate decision making. Given that the execution time granularity of the xApp in OAI's near-RT RIC is approximately 10 seconds by default, it was arbitrarily decided for this PoC that after every three xApp measurements the need to create or remove network slices would be evaluated. For the test execution, each UE ran an iperf3 server, while the OAI-EXT-DN container ran four iperf3 clients, one for each UE, at different times according to the following sequence of events:

1) For the first 5 minutes, a data load was injected toward the priority UE to monitor its received downlink throughput;
2) For the next 10 minutes, the priority UE competed for resources with other UEs, triggering network slicing activation;
3) The priority UE stopped consuming network resources, leading to the deactivation of slicing.

[1]https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/develop/ci-scripts/yaml_files/5g_rfsimulator

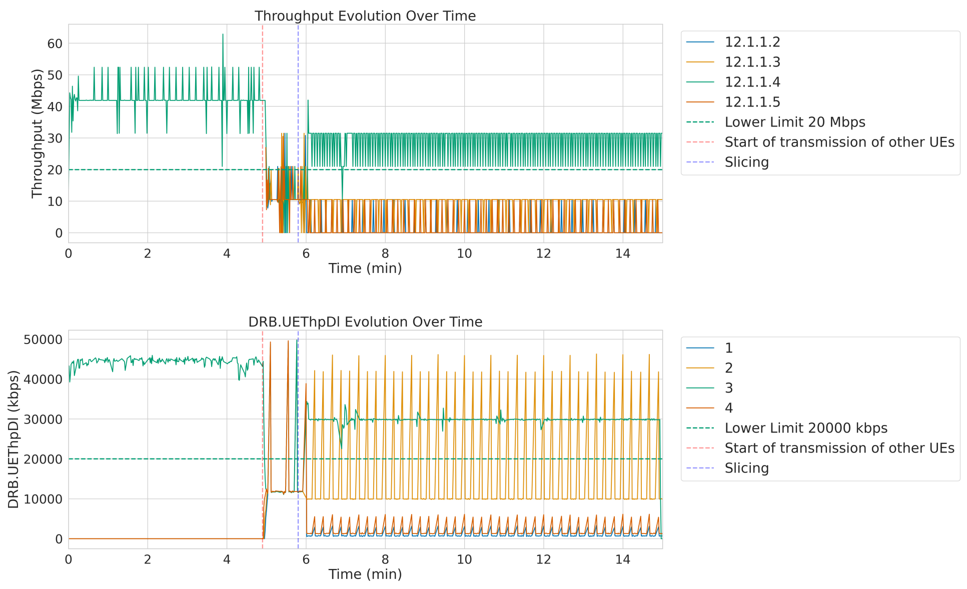

Fig. 3. Throughput over time measured by Iperf (above) and KPM xApp (below) during network slicing activation.

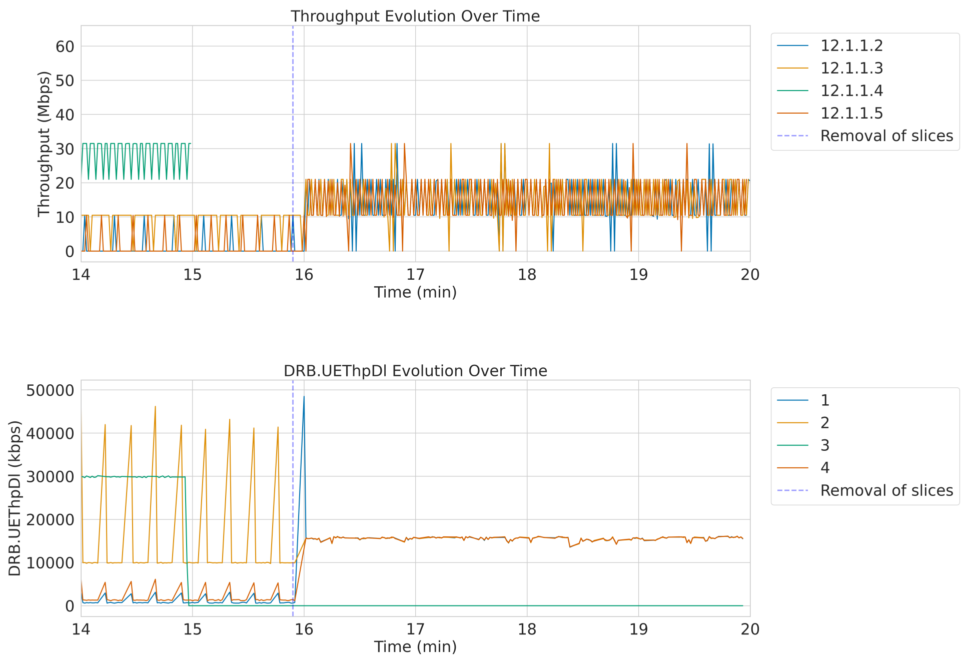

Fig. 4. Throughput over time measured by Iperf (above) and KPM xApp (below) during network slicing removal.

IV. Evaluation

Figures 3 and 4 show network performance in terms of the throughput experienced by the UEs during the experiment, with data collected both by Iperf and KPM xApp. One of the monitoring metrics was DRB.UEThpDL, which represents the downlink throughput of the Data Radio Bearer (DRB) in the RAN.

In Figure 3, a decrease in throughput of the priority UE (UE 3) is observed when other UEs start to consume network resources, triggering the activation of the network slicing. Around the sixth minute, the throughput of the priority UE returns to values above the minimum threshold of 20 Mbps, demonstrating the effectiveness of the network slicing mechanism. On the other hand, Figure 4 shows the throughput of non-priority UEs after slicing removal, which was deactivated because the priority UE no longer required additional resources. It is noticeable that with removal of the slicing, the throughput of each UE increases, but none of them is prioritized with a higher bitrate.

In addition, in Figure 5 the time between the activation or removal of the slices and the observation of the new throughput can be analyzed. The temporal gap between an action and its corresponding reaction is small, although it may not be optimal for certain strict scenarios. The time between slice activation and observation of the new throughput by Iperf3 was 0.37 seconds, while the time for slice deactivation

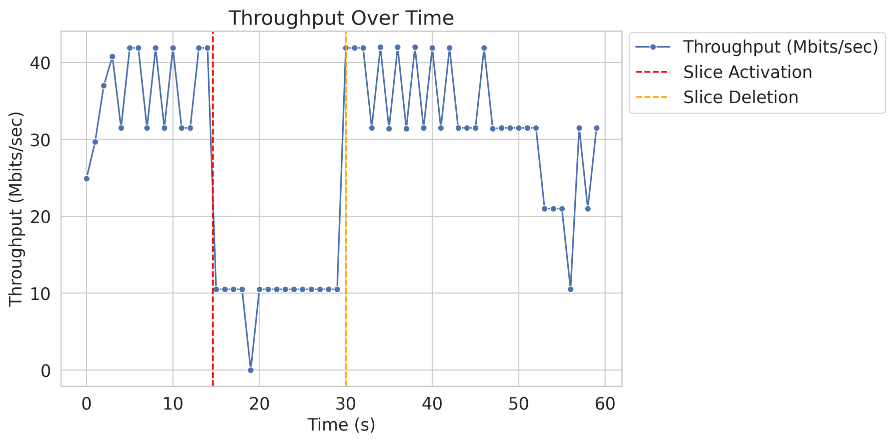

Fig. 5. Changes in throughput experienced by one user equipment, with timestamps indicating the moments of slice activation and removal.

and observation of the previous throughput was 0.05 seconds. These measurements were obtained by comparing the timestamps in form of nanoseconds of the start of Iperf3 execution with the moments of slice activation and removal.

V. Conclusion

In this study, a PoC was implemented to demonstrate the feasibility of performing dynamic slicing on a 5G network using the TC and eBPF tools of the Linux kernel. The method enables granular QoS adjustments by employing eBPF to monitor and classify packets in real-time, supplemented by the HTB queuing discipline enabling hierarchical bandwidth allocation control. The results validate the effectiveness of our approach in providing adaptive network slicing with controlled bandwidth distribution across slices. The PoC was tested through controlled experimentation, where throughput adaptations were confirmed in near-real-time. Although the response times of 0.37s for slice allocation and 0.05s for slice removal are acceptable, further optimizations can be pursued for ultra-low-latency applications.

Furthermore, this work recognizes the potential of eBPF-based traffic control methods for efficient network slicing in 5G networks. The result shows that eBPF with TC can be a viable solution for dynamic QoS provisioning management. However, since eBPF is used here primarily as a tool, further studies should explore how this approach compares and integrates with other slicing mechanisms already proposed for 5G, such as those based on SDN (Software Defined Network), SmartNICs, or virtualized RAN.

The current proof of concept was evaluated in a fixed, simulated testbed environment that does not consider UE mobility or channel condition variations. This limitation should be emphasized, as it impacts the generalization of results to real-world scenarios where radio conditions and user movement dynamically affect network performance. And to strengthen the assessment of robustness and scalability, future work should explore scenarios with varying numbers of UEs, bandwidth, and radio configurations.

Further research will also focus on integrating this method into real 5G deployments and incorporating additional QoS and QoE metrics— such as latency, jitter, and packet loss - to better evaluate service quality.

References

[1] S. Zhang. “An Overview of Network Slicing for 5G”. *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[2] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina. “Network Slicing in 5G: Survey and Challenges”. *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[3] D. Gonçalves, L. Bittencourt and E. Madeira. “Fatiamento Dinâmico de Redes em Computação em Névoa para Usuários Móveis”. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pp. 57–70, Porto Alegre, RS, Brasil, 2021. SBC.

[4] C.-C. Chen, C.-Y. Chang and N. Nikaein. “FlexSlice: Flexible and real-time programmable RAN slicing framework”. In *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, pp. 3807–3812, 2023.

[5] D. Soldani, P. Nahi, H. Bour, S. Jafarizadeh, M. F. Soliman, L. Di Giovanna, F. Monaco, G. Ognibene and F. Risso. “eBPF: A New Approach to Cloud-Native Observability, Networking and Security for Current (5G) and Future Mobile Networks (6G and Beyond)”. *IEEE Access*, vol. 11, pp. 57174–57202, 2023.

[6] “Cilium Documentation”. https://docs.cilium.io/en/stable/. Accessed: 2025-07-16.

[7] “About Calico eBPF”. https://docs.tigera.io/calico/latest/about/kubernetes-training/about-ebpf. Accessed: 2025-07-16.

[8] P. Salva-Garcia, R. Ricart-Sanchez, J. M. Alcaraz-Calero, Q. Wang and O. Herrera-Ruiz. “An eBPF-XDP Hardware-Based Network Slicing Architecture for Future 6G Front- to Back-Haul Networks”. *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 2224–2239, 2024.

[9] F. Parola, F. Risso and S. Miano. “Providing Telco-oriented Network Services with eBPF: the Case for a 5G Mobile Gateway”. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pp. 221–225, 2021.

[10] B. Sharma, S. Vittal and A. A. Franklin. “FlexCore: Leveraging XDP-SCTP for Scalable and Resilient Network Slice Service in Future 5G Core”. In *Proceedings of the 7th Asia-Pacific Workshop on Networking*, APNet ’23, p. 61–66, New York, NY, USA, 2023. Association for Computing Machinery.

[11] A. A. F. Shwetha Vittal. “HARNESS: High Availability Supportive Self Reliant Network Slicing in 5G Networks”. *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 1951–1963, 2022.

[12] A. Matencio-Escolar, Q. Wang and J. M. Alcaraz Calero. “SliceNetVSwitch: Definition, Design and Implementation of 5G Multi-Tenant Network Slicing in Software Data Paths”. *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2212–2225, 2020.

[13] S. Miano, M. Bertrone, F. Risso, M. Tumolo and M. V. Bernal. “Creating Complex Network Services with eBPF: Experience and Lessons Learned”. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–8, 2018.