# Integration of BMv2 and eBPF/XDP for Dynamic Forwarding in 5G Transport Networks

**Jefferson M. O. das Mercês[1,2], José S. da S. Galvão[1,2], Ruan D. Gomes[1,2], Marcelo P. Sousa[1], Moacy Pereira da Silva[1] e Waslon Terllizzie A. Lopes[3]**

[1]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB)

[2]Programa de Pós-Graduação em Tecnologia da Informação (PPGTI)

[3]Universidade Federal da Paraíba (UFPB)

{jefferson.maxmiliano, jose.galvao, ruan.gomes, marcelo.portela moacy.silva}@assert.ifpb.edu.br waslon@ieee.org

***Abstract.*** *This paper analyzes the impact on end-to-end latency of 5G applications by integrating eBPF/XDP-based observability mechanisms with programmable switches in the transport network. This proposal uses eBPF to monitor the UPF and identify UE traffic encapsulated in GTP-U tunnels. An orchestrator processes events related to end-to-end delay variation per-flow and dynamically updates the tables of the switches, enabling real-time route switching according to network conditions. The results indicate that the proposed dynamic forwarding mechanism significantly reduces latency under degraded conditions, achieving up to 78% improvement depending on the duration of the degradation period.*

## 1. Introduction

5G networks establish stringent performance requirements, such as low latency, high availability, and adaptability to different service classes, making them suitable for mission-critical applications, including real-time industrial control and remote procedures [Peterson and Sunay 2020]. In this context, private 5G networks stand out as they operate on dedicated and customized infrastructures, offering greater security, predictability, and performance compared to public networks. Furthermore, they enable hybrid or Network Slicing-based (NS) implementations, according to application needs [Domb et al. 2025].

To meet these requirements, the 5G core (5GC) network adopts the Network Function Virtualization paradigm, which enables the instantiation of elements such as the Access and Mobility Management Function (AMF) and the User Plane Function (UPF) on-demand over heterogeneous computing resources. Even with this capability, dynamic mechanisms in the transport network capable of ensuring low-latency routes between the Radio Access Network (RAN) and the 5GC are still required [Dahlman et al. 2020].

In this scenario, it is necessary to implement mechanisms that provide greater resilience to the transport network. In this context, the use of programmable switches based on Programming Protocol-independent Packet Processors (P4), such as Behavioral Model version 2 (BMv2), constitutes a viable alternative for making forwarding decisions directly in the data plane. Thus, the transport network no longer acts merely as

a passive medium, but provides active support for low latency and dynamic decision-making, contributing to fulfilling the requirements of critical applications in 5G networks [Paolucci et al. 2021].

Despite these capabilities, efficient decision-making in the data plane relies on the availability of accurate and up-to-date information regarding the state of the network and 5GC functions. In this regard, extended Berkeley Packet Filter (eBPF) serves as a complementary technology, as it enables the execution of programs directly within the Linux kernel space without requiring operating system recompilation or system reboots. It allows for the straightforward addition of new functionalities, facilitating code development using well-established languages such as C and Python [Rice 2023]. The applications of eBPF range from observability, which entails operating system instrumentation for the collection of metrics and events, to the detailed monitoring of network traffic.

Among the mechanisms supported by eBPF, eXpress Data Path (XDP) stands out, as it operates at the lowest layer of the kernel network stack and ensures greater efficiency in packet processing [Vieira et al. 2020]. Consequently, it provides elevated privileges and faster access to packets that arrive at a specific interface. Thus, eBPF enables real-time monitoring directly within the 5GC function kernel, allowing for the selective inspection of packets, flows, and signaling messages. Correlating this information with the internal state of elements such as the UPF and Session Management Function (SMF) enables the detection of anomalies that cause performance degradation, as well as signaling-based attacks [Nunziati et al. 2025]. Therefore, integration between the programmable data plane and kernel-level observability mechanisms strengthens the resilience of the 5G architecture and enables proactive and adaptive actions in the transport network.

This work investigates the impact on end-to-end latency of 5G applications when using eBPF/XDP observability to trigger dynamic forwarding in a BMv2-based programmable transport network. The scenario considers multiple routes between the UPF and the User Equipments (UEs), with degradation induced on one of the links via the Traffic Control (TC) tool to evaluate network resilience. The orchestrator detects the increase in per-flow latency, associates the issue with the corresponding UE, and performs selective route switching. The results indicate that dynamic route switching can reduce latency by up to 78% compared to scenarios without adaptation mechanisms.

The main contributions of this work are: $i)$ the definition of a dynamic forwarding architecture for 5G transport networks that integrates eBPF/XDP-based observability mechanisms with programmable BMv2 switches; $ii)$ a mechanism for the identification and handling of traffic encapsulated in GPRS Tunnelling Protocol – User Plane (GTP-U) tunnels; and $iii)$ the implementation and experimental evaluation of the solution in a virtualized 5G environment. This paper is organized as follows: Section 2 reviews related work. Sections 3 and 4 describe the proposed methodology and the results, respectively. Finally, Section 5 concludes the paper.

## 2. Related Work

The work presented in [da Silva et al. 2025] evaluates dynamic traffic redirection in a simulated 5G network scenario with three routes: a standard route with a latency of 10 ms, another with approximately 15 ms, and a third defined by the ONOS Software-Defined Networking (SDN) controller. The results demonstrate that SDN-based dynamic routing

outperforms the simple use of backup routes, improving the resilience and stability of the network. In contrast, the present work employs eBPF/XDP observability in the data plane, enabling more granular forwarding decisions suitable for latency-sensitive applications.

The study in [Puttlitz et al. 2024] introduces P4NetIntel, an end-to-end telemetry framework that combines In-band Network Telemetry (INT) with host-level monitoring using eBPF and XDP to address the shortcomings of conventional observability methods. The architecture features a programmable-switch data plane for network telemetry and eBPF-based host components that capture packets via XDP and TC hooks, enabling complete flow lifecycle observation. To limit overhead, P4NetIntel uses packet sampling and data discretization, reducing both the inserted metadata and processing cost. Experiments show that the joint use of INT and eBPF achieves end-to-end visibility with negligible impact on application latency, while the integrated analysis of network and host states removes the need for external data correlation and supports accurate observability in distributed systems. Unlike the system presented in [Puttlitz et al. 2024], this work proposes an approach that functions similarly to an SDN controller, in which eBPF acts as a Southbound API to capture and identify source IPs, while a user-plane orchestrator serves as a Northbound API, populating the BMv2 switch tables based on the collected data.

The article [Ye et al. 2025] introduces Dynamic Path Switching (DPS), a traffic engineering mechanism for SD-WAN that exploits multiple paths over Internet/VXLAN tunnels. The solution attains 99.97% service availability and a 64.26% cost reduction relative to conventional approaches. DPS is implemented with eBPF, enabling real-time header inspection and modification without changes to the Linux kernel. It relies on eBPF programs attached to XDP and TC: the probing module runs STAMP-Collector and STAMP-Reflector in XDP to assess path performance with low CPU overhead, while the switching module operates at TC to rewrite ports and steer flows based on network conditions. User–kernel interaction is realized via eBPF Maps. The experiments demonstrate improved scalability and accuracy, with stable Round-Trip Time (RTT) even under heavy flow loads. In contrast, the present work applies similar principles directly in the transport network using programmable switches, enabling finer-grained control at the UE level.

## 3. Experiment Methodology

This section describes the adopted methodology, including the experimental 5G network topology, the transport network with programmable switches, the use of eBPF/XDP for data collection in the 5GC, and the software responsible for system orchestration.

### 3.1. Topology

The experimental environment consists of a 5G architecture integrated into a programmable transport network with three BMv2 switches and eBPF-based inspection mechanisms implemented in the 5G UPF. The scenario was executed in a virtualized environment on a Windows 11 computer, equipped with an Intel® Core™ i7-12700H processor and 32 GB of RAM. For the development of the experiment, VirtualBox version 7.1.4 was installed[1].

As illustrated in Figure 1, five Linux Virtual Machines (VMs) were deployed in VirtualBox. Three of them are responsible for running the BMv2 switches, each config-

[1] `https://www.virtualbox.org/wiki/Changelog-7.1\#v4`

ured with 4 GB of RAM and 4 vCPUs. The machine dedicated to the 5GC was configured with 6 GB of RAM and 6 vCPUs. Meanwhile, the VM used to simulate gNodeB and the three UEs utilized 8 vCPUs and 6 GB of RAM. The connectivity between the VMs was established by adding multiple network adapters in VirtualBox, configured as an Internal Network, which allowed the implementation of the topology shown in Figure 1.

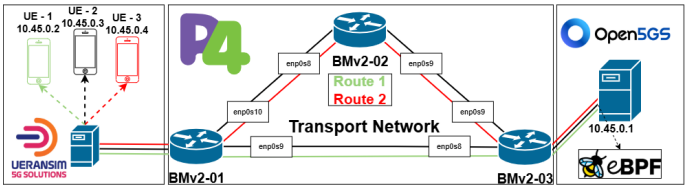



**Figure 1. Topology of the experimental architecture.**

### 3.2. RAN, 5G Core and Transport Network

The implemented 5G network consists of three main parts: the RAN, the Transport Network, and the 5GC. Since this is a simulated environment, the study utilized the UERANSIM[2] software, which is responsible for simulating the gNodeB and the UEs. In this study, one gNodeB and three UEs were configured and executed within the same VM. The 5GC manages the connectivity, mobility, and data traffic policies for registered mobile devices. The 5GC was implemented using the Open5GS[3] software.

In the Transport Network, programmable switches were used with support for the P4 language [Liatifis et al. 2023]. This choice allows part of the decision logic to be offloaded to the data plane, optimizing processing. BMv2 switches are software implementations designed for the development, debugging, and functional validation of programs written in P4 [Tsareva et al. 2021]. Although they do not offer sufficient performance for production-scale traffic, they are used to prototype and validate P4 programs before deployment on specialized hardware, such as Tofino switches [Makhroute et al. 2023]. Finally, the last mechanism implemented in the transport network operates on the BMv2-01 switch, at the interface (`enp0s9`). This mechanism uses Linux TC and enables the controlled degradation of Route 1, with the goal of evaluating network behavior in adverse scenarios [Borda and Ermont 2022]. The TC is executed five seconds after the start of the experiment and introduces latency variations periodically every five seconds, allowing for the analysis of the network's response to this condition.

### 3.3. eBPF/XDP

The VM hosting the 5GC also runs eBPF/XDP-based network observability software. Operating at the lowest level of the Linux kernel, this program executes privileged monitoring code on the interface to capture all incoming packets [Rice 2023]. Figure 2 illustrates the data collection flow of the eBPF code[4]. The code was written using the BCC library[5] and was divided into two parts: $i$) The C code is responsible for the logic of inspecting and extracting information from each packet entering the interface, given the XDP technology's limitation of not inspecting outgoing packets [Rice 2023]. When a packet arrives

[2]`https://github.com/aligungr/UERANSIM`
[3]`https://github.com/open5gs`
[4]`https://github.com/jose-galvao/BMv2-eBPF-SBRC-2026`
[5]`https://github.com/iovisor/bcc`

at the interface, the program attached to the XDP hook performs only the parsing and inspection of the packet headers (identifying GTP-U and internal IP headers), using the structures defined in the code. The data collected are sent to the user space through a perf ring buffer via `BPF_PERF_OUTPUT`, an efficient mechanism for event transmission; $ii$) In the user space, the Python program periodically reads these maps, which are used to manipulate traffic; furthermore, the user-space program is responsible for the management of the eBPF code, compiling it and attaching the program to the hook.

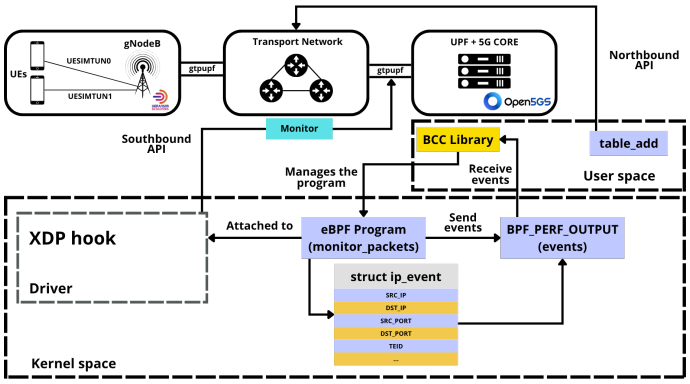



**Figure 2. Experiment pipeline.**

### 3.4. Integration and evaluation of flows

A Python program was developed to orchestrate the experiment, integrating latency monitoring, UPF observability through eBPF/XDP, and decision-making at the BMv2 switch. The system measures the latency between the UEs and the UPF to obtain the RTT in the transport network. Due to the aforementioned limitations of XDP, measurement is performed using the Internet Control Message Protocol (ICMP). The solution centralizes control and automates the adjustment of the switch tables. This allows for dynamic decisions based on network performance. In this context, the ICMP RTT is used as a proxy for transport path latency, assuming route symmetry and homogeneous link behavior throughout the path. In this process, the system sends ICMP packets to the UEs and stores the collected information in a CSV file.

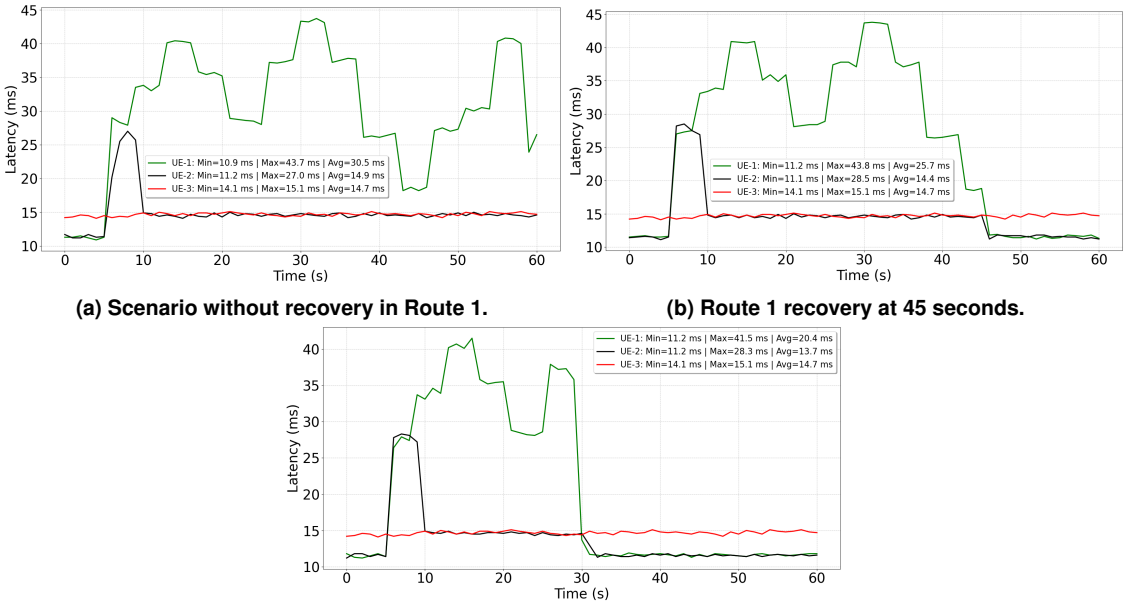
Regarding the latency on the two routes analyzed for UE-1, the traffic traverses the BMv2-01 and BMv2-03 switches, with an average latency of 11.4 ms. In contrast, the route corresponding to UE-3 presents higher latency due to an additional hop as it passes through the BMv2-01, BMv2-02, and BMv2-03 switches, with an average latency of 14.7 ms. To obtain these measurements, the BMv2 switch tables were pre-configured to ensure that the traffic generated by both UEs followed pre-defined routes. As a result, it was possible to continuously monitor the RTT on each route and, additionally, enable the dynamic forwarding of UE-2 through the lowest-latency route.

In parallel, the eBPF program monitors the UPF interface, identifying the internal UE IPs within the GTP-U tunnel. The system tracks end-to-end latency by calculating a five-packet sliding window average. If this average exceeds 15 ms, a threshold established by an industrial study on autonomous mobile robots [Segura et al. 2024], the Python orchestrator accesses the BMv2 switches via SSH, updating their flow tables to apply specific rules to the UE-2 traffic. This approach enables granular per-UE control, unlike traditional solutions that affect entire gNodeB traffic.

## 4. Results

Figure 3a presents the end-to-end latency performance of the three UEs evaluated in the experiment: UE-1, UE-2, and UE-3, represented by the colors green, black, and red, respectively. The experiment had a total duration of 60 seconds and was conducted to evaluate the proposed ability of the mechanism to maintain the latency of UE-2 below the 15 ms threshold by dynamic route forwarding, even under conditions of induced degradation using TC.

In the scenario without Route 1 recovery, it is observed that UE-3 maintains a constant latency throughout the experiment, with an average value of approximately 14.7 ms. UE-1 and UE-2 begin the experiment with a latency close to the nominal value, approximately 11.4 ms. However, 5 seconds into the experiment, Route 1 degradation occurs, causing a significant increase in the latency of UE-1 and UE-2, reaching values close to 27 ms. The orchestrator software identifies this degradation and reroutes UE-2 to Route 2, which is reflected in the peak observed in the corresponding curve between 5 and 10 seconds, the interval required to detect and enforce the forwarding. In contrast, UE-1 remains on the degraded Route 1, exhibiting high and unstable latency values throughout the experiment. This behavior demonstrates the effectiveness of the selective forwarding mechanism, which acts exclusively on UE-2, keeping the other UEs on the original route.




**(a) Scenario without recovery in Route 1.**

**(b) Route 1 recovery at 45 seconds.**

**(c) Route 1 recovery at 30 seconds.**

**Figure 3. Results.**

Taking UE-2 as a reference, its average latency is observed to be approximately 14.9 ms, a value very close to that of UE-3, whose average latency is around 14.7 ms. This results in a difference of 1.3%, indicating equivalent performance between both. In contrast, compared to UE-1, which exhibits an average latency of approximately 30.5 ms, UE-2 demonstrates an improvement of around 51%, evidencing that route switching allows UE-2 to remain on a route that performs better, even under degradation scenarios.

Another scenario analyzed, as shown in Figure 3b, considers the recovery of Route 1 at 45 seconds in the experiment. Taking UE-2 as a reference, it is observed that

UE-3 exhibits an average latency only 2.1% higher, indicating equivalent performance between these routes. Even with switching between routes, the recovery of Route 1 at 45 seconds results in better performance than that observed in the previous scenario, where no recovery occurred, reducing the latency difference to 1.3%. These results indicate that the faster the primary path recovers, the lower the average latency exhibited by UE-2. In contrast, UE-1 presents an average latency 78.5% higher than UE-2 in this scenario. This high percentage is due to the persistence of the degradation up to 45 seconds, demonstrating that the mechanism's performance gain is proportional to the duration of the anomaly on the primary route.

Finally, Figure 3c presents the scenario in which the recovery of Route 1 occurs at 30 seconds. In this case, the earlier recovery results in additional performance gains. Taking UE-2 as a reference, it is observed that its performance is approximately 7.3% better than that of UE-3. Compared to UE-1, UE-2 presents an average latency reduction of approximately 48.9%, demonstrating that the faster recovery of the primary route significantly reduces the effects of initial degradation. These results reinforce that early recovery of the primary path is a determining factor in maintaining low latency levels, bringing the performance of UE-2 closer to nominal values.

## 5. Conclusion and Future Work

This work analyzes end-to-end latency in 5G networks by integrating eBPF/XDP with BMv2 switches, enabling granular data plane forwarding. The results show that the proposed mechanism successfully maintained the monitored UE's latency close to nominal levels, even under route degradation, reducing latency by up to 78.5% when redirecting traffic to alternative paths. In scenarios featuring partial recovery of the degraded link, earlier route recovery significantly mitigates the initial degradation impact, aligning the UE's performance with normal operating conditions. In late recovery scenarios, the performance of different forwarding strategies tends to equalize, suggesting that traditional global redirection mechanisms might suffice for less latency-sensitive contexts. However, for environments that require higher granularity, the proposed approach is more suitable.

Future work includes enhancing the orchestrator software through tighter integration with P4Runtime to improve control over programmable data planes and enable more efficient and fine-grained network reconfiguration. We also plan to evaluate scalability with additional UEs and multiple concurrent paths, as well as to analyze the impact of the mechanism on system resource consumption.

## 6. Acknowledgments

This work is supported by EMBRAPII (BFA 2301.0001), Cisco, Prysmian, and MPT Cable. The authors also thank CNPq (307108/2025-2), CPQD, Inatel, Taggen, Data Machina, and the IFPB Innovation Hub.

## References

Borda, S. T. and Ermont, J. (2022). An evaluation of software-based tsn traffic shapers using linux tc. In *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*, pages 1–4.

da Silva, R. L., das Merces, J. M. O., Sousa, M. P., Araujo, A. S., Alencar, A., Meneses, T. F., Dias, M., and Santos, D. F. S. (2025). Redirecionamento dinâmico de tráfego em rede de transporte 5G definida por software. In *Anais do XLIII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2025)*.

Dahlman, E., Parkvall, S., and Skold, J. (2020). *5G NR: The Next Generation Wireless Access Technology*. Elsevier Science.

Domb, M., Cg, B., Suman, M., A., G., and Joshi, S. (2025). Securing 5G networks by mitigating cybersecurity risks for transformative applications. *International Journal of Interactive Mobile Technologies (iJIM)*, 19:227–255.

Liatifis, A., Sarigiannidis, P., Argyriou, V., and Lagkas, T. (2023). Advancing SDN from openflow to P4: A survey. *ACM Comput. Surv.*, 55(9).

Makhroute, E.-M., Elharti, M.-A., Brouillard, V., Savaria, Y., and Ould-Bachir, T. (2023). Implementing and evaluating a P4-based access gateway function on a tofino switch. In *2023 6th International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1–7.

Nunziati, G., Fiandrino, C., Foschini, L., and Bellavista, P. (2025). Monitoring 5G core networks vulnerabilities with eBPF. *IEEE Networking Letters*, 7(3):220–223.

Paolucci, F., Scano, D., Cugini, F., Sgambelluri, A., Valcarenghi, L., Cavazzoni, C., Ferraris, G., and Castoldi, P. (2021). User plane function offloading in P4 switches for enhanced 5G mobile edge computing. In *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 1–3.

Peterson, L. and Sunay, M. O. (2020). *5G Mobile Networks: A Systems Approach*. Synthesis Lectures on Network Systems. Morgan & Claypool Publishers.

Puttlitz, C., Parizotto, R., and Schaeffer-Filho, A. (2024). P4netintel: End-to-end network telemetry with ebpf and xdp. In *2024 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–6.

Rice, L. (2023). *Learning eBPF: Programming the Linux Kernel for Enhanced Observability, Networking, and Security*. O'Reilly Media, 1 edition.

Segura, D., Damsgaard, S. B., Kabaci, A., Mogensen, P., Khatib, E. J., and Barco, R. (2024). An empirical study of 5G, Wi-Fi 6, and multi-connectivity scalability in an indoor industrial scenario. *IEEE Access*, 12:74406–74416.

Tsareva, I., Scholz, D., and Gallenmüller, S. (2021). Taxonomy of the performance of P4 targets. Seminar IITM SS 21, network architectures and services, Technical University of Munich.

Vieira, M. A. M., Castanho, M. S., Pacífico, R. D. G., Santos, E. R. S., Júnior, E. P. M. C., and Vieira, L. F. M. (2020). Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications. *ACM Comput. Surv.*, 53(1).

Ye, M., Han, B., Fang, X., Zou, X., Bao, X., Xie, X., Xiao, S., Lin, Y., and Chao, H. J. (2025). Dynamic path switching for traffic engineering in SD-WAN with eBPF. In *2025 IEEE 26th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–7.