Towards Closed-Loop Management in Private 5G Networks with P4/eBPF Telemetry and Reinforcement Learning

Vinícius S. Simão[1,2], Lucas V. Monteiro[1], Pedro H. Rodrigues Alves[1], Ruan D. Gomes[1,2,3], Leandro C. de Almeida[1,3], and Paulo D. Maciel Jr.[1,2,3]

[1] IFPB Innovation Hub, Federal Institute of Paraíba (IFPB), João Pessoa/PB, Brazil

[2]Postgraduate Program in Electrical Engineering (PPGEE), Campus João Pessoa

[3]Postgraduate Program in Information Technology (PPGTI), Campus João Pessoa

{simao.vinicius,pedro-alves.pa,vieira.monteiro}@academico.ifpb.edu.br,

{ruan.gomes,leandro.almeida,paulo.maciel}@ifpb.edu.br

Abstract. *The growing demand for latency-critical services in private 5G deployments drives the need for autonomous, closed-loop network management that can continuously adapt to changing conditions. An important challenge is to obtain a unified view of both the transport-network behavior and the use of compute resources without incurring excessive overhead. In this context, we introduce a closed-loop design that integrates P4-based out-of-band network telemetry (ONT) with CPU KPIs obtained from the User Plane Functions (UPFs) using the Extended Berkeley Packet Filter (eBPF). The metrics obtained are used as input to a reinforcement learning (RL) agent that adjusts the priority of the switch queue and redirects traffic through different paths in the 5G network, dynamically selecting the UPF through the Network Exposure Function (NEF) API. In the scenario that combines UPF monitoring with RL-based traffic prioritization, a 46.48% reduction in latency and a 57.95% reduction in jitter were obtained, demonstrating the benefit of joint transport-and-core control.*

1. Introduction

5G networks introduce a fundamental paradigm shift in core network design through the adoption of a Service-Based Architecture (SBA). In this model, the core functionalities are decomposed into independent Network Functions (NFs) implemented as microservices and interconnected through standardized APIs (ETSI 2026). This architecture enables greater flexibility, scalability, and rapid deployment, which are particularly attractive in private 5G deployments that target latency-sensitive and resource-constrained environments (Eswaran and Honnavalli 2022).

Although SBA significantly improves modularity and operational agility, it also increases the complexity of coordinating and managing distributed core components (Goshi et al. 2021). The dynamic placement of virtualized NFs (VNFs) and the variability of traffic demand render static configuration approaches ineffective. Consequently, meeting strict performance objectives such as low latency and efficient resource utilization becomes increasingly difficult in operational 5G core networks (Hoa et al. 2025).

These challenges have motivated the adoption of closed-loop management as a fundamental design principle for modern networks (Simão et al. 2025). By continuously observing the system state, analyzing performance indicators, and applying control actions, closed-loop mechanisms enable networks to adapt autonomously to changing conditions, moving beyond reactive and manual management models. Recent advances have further enhanced closed-loop control through the integration of machine learning (ML) techniques, particularly reinforcement learning (RL). RL-based approaches are well-suited to network control problems due to their ability to learn adaptive policies through interaction with dynamic environments (Wu et al. 2023). Unlike rule-based or supervised methods, RL can continuously adjust its decisions in response to fluctuating traffic loads and evolving system states, making it a promising tool for latency-aware traffic management and load balancing.

At the same time, network observability has evolved substantially with the advent of programmable data planes and kernel-level packet processing. Technologies such as P4 and eBPF enable fine-grained monitoring at line rate, allowing telemetry functions to be executed directly in the packet processing path. These capabilities provide unprecedented visibility into both network behavior and computational resource usage while maintaining low overhead (Puttlitz et al. 2024). In particular, In-band Network Telemetry (INT) enables the collection of per-hop network metrics, such as queue occupancy and latency, by embedding telemetry data directly into packets as they traverse the network. Alternatively, diagnostic packets can be used to measure network status (Out-of-band Network Telemetry – ONT), in which metadata is added only to these packets, without interfering with application traffic. When using ONT, there is also a trade-off related to the number of diagnostic packets that need to be inserted into the network to collect metrics with the desired granularity (Simão et al. 2025). Complementarily, eBPF allows for efficient extraction of CPU and memory utilization metrics from VNFs, such as User Plane Functions (UPFs), within the 5G core. Together, these mechanisms enable a holistic and real-time view of both the transport network and the core processing infrastructure.

In this work, we argue that integrating transport-level telemetry captured via P4/ONT with UPF-level processing metrics collected using eBPF yields a unified, fine-grained view of both network and compute behavior, enabling more informed, timely, and effective closed-loop network control (CLNC) decisions in private 5G networks. We propose a CLNC architecture in which these heterogeneous metrics jointly form the state observed by a controller agent. The agent dynamically controls traffic steering and resource allocation in the core network to minimize end-to-end latency through adaptive load balancing across the transport network and UPFs. The controller leverages RL to adjust transport-switch queue priorities and uses CPU-related key performance indicators (KPIs) to steer traffic across multiple UPFs, continuously adapting its decisions to current network and compute conditions. In general, the evaluated configurations substantially reduced both average end-to-end delay and jitter, achieving up to a 46.48% decrease in latency and a 57.95% decrease in jitter.

The remainder of the paper is structured as follows. Section 2 presents an overview of the related work. Section 3 details the proposed CLNC architecture and describes how the reinforcement learning model was instantiated. The evaluation and

results are detailed in Section 4, including a description of the testbed and workload behavior. Finally, conclusions are drawn in Section 5.

2. Related Work

This section discusses the main related works on the use of control and actuation mechanisms. For this purpose, five criteria are defined to classify and comparatively analyze the studies, as summarized in Table 1.

Table 1. Comparative analysis of related work based on key characteristics. The table contrasts prior studies according to their supported features. Checkmarks denote supported capabilities.

Articles	Load Balancing	Reinforcement Learning	eBPF	Network Telemetry	Private 5G
(Simão et al. 2025)	✗	✗	✗	✓	✓
(Hyun et al. 2018)	✗	✗	✗	✓	✗
(Puttlitz et al. 2024)	✗	✗	✓	✓	✗
(Abranches et al. 2021)	✗	✗	✓	✓	✗
(Pizzutti and Schaeffer-Filho 2019)	✓	✗	✗	✗	✗
(Zheng et al. 2023)	✓	✓	✗	✗	✗
(Tajbakhsh et al. 2026)	✓	✓	✗	✗	✗
(Puttlitz and Schaeffer-Filho 2025)	✓	✓	✓	✓	✗
This Work	✓	✓	✓	✓	✓

In a previous work (Simão et al. 2025), we proposed a closed-loop network control mechanism for industrial edge computing environments in private 5G networks, integrating telemetry collected from a programmable data plane with an AI-based prediction model for proactive network resource management and latency-sensitive traffic prioritization. The experimental results demonstrated significant reductions in end-to-end latency and jitter, validating the effectiveness of the proposed mechanism in industrial settings. Although this previous work used network telemetry and AI to control the transport network in the context of a private 5G network, it does not consider the use of eBPF to obtain metrics from hosts, and neither performs load balancing employing multiple UPFs. In addition, in this present work we used RL, which is more suitable for load balancing tasks considering dynamic workloads.

The work described in (Hyun et al. 2018) proposes a scenario of autonomous networks that integrates knowledge-defined networks (KDN), software-defined networks (SDN) and INT for traffic engineering and anomaly detection. The proposed architecture is organized into four planes: control plane, data plane, management plane, and knowledge plane. In the control plane, the SDN controller implements telemetry mechanisms for programmable switches via P4. In the data plane, INT metadata is generated and extracted directly from packets. In the management plane, this metadata is collected, stored, and aggregated for further analysis. In the knowledge plane, the information obtained feeds ML algorithms that guide dynamic adaptations to the network configuration. Although the approach enables packet-level monitoring, providing metrics such as latency, queue occupancy, and congestion, it is limited by high CPU and memory processing costs and increased bandwidth consumption for transporting telemetry data, which restricts its applicability in virtualized environments.

With the goal of expanding network observability, the work described in (Puttlitz et al. 2024) proposes an end-to-end telemetry solution that integrates INT and eBPF/XDP, allowing simultaneous collection of network and end-host metrics throughout the entire lifecycle of a flow. This approach eliminates the need for additional synchronization or correlation mechanisms to obtain end-to-end insights, providing an integrated view of distributed application performance. However, collected data are not exploited by adaptive decision-making mechanisms, such as dynamic load balancing or flow control policies.

At the same time, the study focused on operational efficiency presented in (Abranches et al. 2021) demonstrates the potential of eBPF and XDP for low-overhead network monitoring, exploiting kernel-level packet processing and partial offloading of functions to SmartNICs. The approach is applied in scenarios such as traffic accounting, identification of half-open TCP connections, and DNS flow analyzer, showing significant gains in efficiency and reduced energy consumption. However, the solution focuses predominantly on metrics observation and analysis, without employing automatic decision-making mechanisms or network adaptability.

In the context of load balancing in programmable networks, the paper (Pizzutti and Schaeffer-Filho 2019) presents a multipath routing strategy based on a hybrid SDN/OpenFlow and P4 architecture. The proposal combines an online control loop, executed directly on the switches to enable rapid decision-making, with an offline control loop, implemented on the controller, responsible for analyzing routes and applying global changes to the network. The approach is evaluated using route update policies based on the Exponential Moving Average (EMA) and Weighted Moving Average (WMA) of round-trip time (RTT) measurements, exploiting the programmability of the data plane to enable the use of the Flowlet technique in active load balancing, while the control plane participates passively to react to traffic variations. However, the decisions adopted do not incorporate adaptive learning mechanisms or consider the computational state of the nodes involved.

In a more advanced approach, the work (Zheng et al. 2023) introduces reinforcement learning directly into the data plane, applying Q-learning in programmable switches to dynamically adjust path weights based on queue metrics collected via INT, allowing information on network congestion and utilization to be obtained. This approach eliminates the bottlenecks associated with centralized controllers and offers high scalability. However, the state of the environment is restricted to queue metrics, disregarding relevant information from the computing infrastructure, such as CPU and memory utilization, and does not consider dependencies with the transport network or virtualized network functions.

In this context, the approach proposed in (Tajbakhsh et al. 2026) is noteworthy, since it introduces a load balancer for SDN that operates jointly across the control and data planes and leverages RL to enable adaptive decision-making. In the control plane, an RL agent processes information about the status of computational resources and learns policies that can assign optimal weights to servers and hardware accelerators, including TPUs, GPUs, and SmartNICs. These policies are calculated asynchronously and dynamically in the data plane, which implements a dynamic weighted round-robin (DWRR) mechanism for load distribution according to the learned weights. The

architecture allows for continuous adjustments in response to small or abrupt changes in the environment, with autonomous retraining based on the states observed in the data center and support for resource allocation at a fine granularity level. However, the solution is evaluated in the context of data center networks, not explicitly considering private 5G network scenarios.

Similarly, the work (Puttlitz and Schaeffer-Filho 2025) proposes a decentralized load balancing scheme that combines network telemetry via INT, host metrics collected through eBPF, and RL. The system is structured around two control loops: an intelligent offline loop, operating at the control plane, responsible for analyzing the overall state of the infrastructure and making global decisions; and an online loop at the edge switches, responsible for quick decisions and immediate reactions to network variations. The experiments consider scenarios of host and link congestion, assigning weights to paths based on network metrics, host metrics, or a combination of both in the composition of the state. The results demonstrate that decisions based on end-to-end metrics can provide a more efficient traffic distribution. However, the system remains strongly coupled to the programmable data plane and is targeted at data center environments, not exploring scenarios where VNFs, such as UPFs in 5G networks, play a central role in traffic forwarding and processing.

In summary, the related work presents significant advances in the development of network telemetry mechanisms, efficient metric collection through eBPF/XDP, data plane programmability with P4, and RL-based decision making. However, in contrast to the aforementioned studies, this paper implements and integrates all these mechanisms within a private 5G network, while simultaneously taking into account transport network metrics, as well as CPU and memory utilization data from the UPFs. In addition, it introduces a hybrid strategy that leverages RL to perform both load balancing and adaptive data flow control, thus realizing a full loop of observation, decision, and action.

3. Proposal

This section outlines the proposed approach: the experimental setup used for evaluation, an out-of-band telemetry pipeline to observe network conditions without affecting traffic, and a RL model that uses network queue-related metrics to make adaptive control decisions.

3.1. Experimental Setup

This work proposes a closed-loop management architecture for private 5G networks that extends previous efforts by jointly incorporating telemetry of the transport-network and the observability of the core network functions into an RL-driven control framework, as shown in Figure 1. The proposed architecture is designed to operate in a service-based 5G core environment, dynamically adapting both traffic prioritization in the transport network and traffic steering across multiple UPFs, to minimize end-to-end latency and avoid computational bottlenecks in the core.

The considered topology differs from previous designs by introducing multiple UPF instances connected through an intermediate UPF (I-UPF). In this architecture, the I-UPF acts as a traffic aggregation and steering point, forwarding user-plane traffic to one of two downstream UPFs (UPF1 or UPF2). This configuration enables flexible traffic

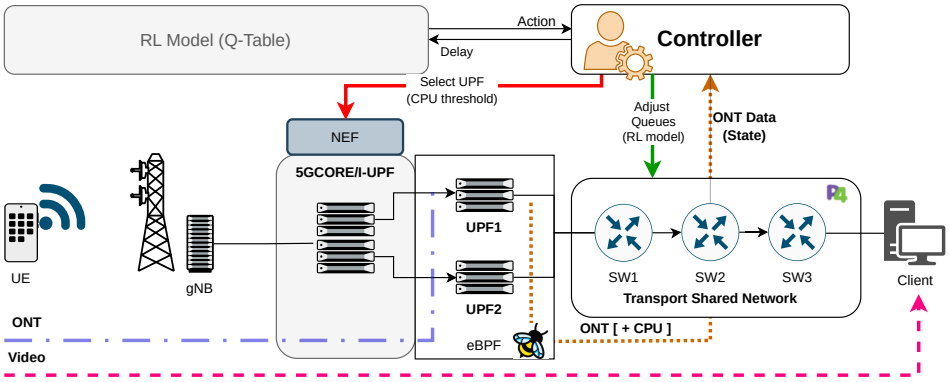


Figure 1. Framework setup configuration.

distribution across parallel processing paths while preserving compatibility with the 5G core service-based architecture.

To support closed-loop decision making, the proposed system integrates heterogeneous telemetry sources. Transport-network observability is provided through ONT (probe packets), which collects fine-grained metrics such as queue occupancy and latency without adding overhead to application packets. At the same time, eBPF programs are deployed at the UPF level to collect real-time computational metrics, including CPU and memory utilization of UPF1 and UPF2. As shown in Figure 1, eBPF intercepts ONT probe packets and embeds UPF performance metrics into a predefined telemetry header, correlating network and processing level observations within a unified telemetry pipeline.

The controller is a software component that integrates an RL model with a monitoring function responsible for validating incoming CPU and memory telemetry against predefined thresholds. It continuously ingests telemetry data and uses this information to construct the environment state observed by the RL agent. Although CPU metrics are collected by telemetry, they are not directly included in the agent's state representation; instead, the controller independently evaluates these metrics to detect overload conditions based on a pre-established threshold. This combined representation of transport-network conditions and core-network resource utilization enables the controller to reason about congestion and overload across both domains. Based on the inferred state, the controller uses the RL agent to select control actions that adjust traffic prioritization in the transport network and, in parallel, uses CPU threshold checks to inform traffic routing decisions at the core network level, forming a closed control loop.

Traffic prioritization actions are applied to the transport network to protect latency-sensitive flows under congestion, while traffic steering actions are triggered when excessive computational load is detected in a UPF. Specifically, when the telemetry indicates CPU utilization above a predefined threshold at UPF1 or UPF2, the controller initiates a UPF reassignment procedure. This reassignment is performed through the Network Exposure Function (NEF) by explicitly leveraging the Traffic Influence API, which materializes traffic steering decisions and allows the controller to influence routing policies in a standardized and 3GPP-compliant manner. Using NEF, the proposed system

dynamically steers traffic between UPF instances without disrupting ongoing sessions, as reflected in the interaction between the controller and the components of the core depicted in Figure 1.

Our proposal assumes varying loads over time, as traffic patterns fluctuate based on user behavior and service demand, while computational loads on UPF instances may vary unpredictably due to protocol processing and data plane operations. The RL agent is designed to operate in this dynamic context. By continuously ingesting the telemetry stream and combining P4-based transport metrics with eBPF-derived CPU utilization, the agent builds a general, real-time representation of the system state. This closed-loop online learning capability allows the controller to move beyond static optimization models (such as the autoregressive model for predicting traffic behavior) or pre-computed traffic policies, which are inherently fragile in dynamic scenarios. Instead, the RL model dynamically adapts its integrated control strategy and learns to balance precise adjustments in the transport network, prioritizing traffic in the priority queue. This continuous adaptation ensures that the system can maintain stringent latency and high reliability requirements, fulfilling the promise of a truly responsive and self-optimizing private 5G core.

Through this design, the proposed architecture enables coordinated control across the transport network and the 5G core. By combining P4-based telemetry, eBPF-based UPF observability, and RL-driven decision making, the system goes beyond single-domain optimization and enables holistic closed-loop management tailored to private 5G deployments with strict latency and performance requirements.

3.2. Out-of-band Network Telemetry

Out-of-band Network Telemetry (ONT) is employed in the proposed architecture to collect detailed transport-network metrics. Unlike in-band approaches, which embed telemetry information directly into application packets, ONT relies on dedicated probe packets that traverse the same forwarding paths used by the data traffic, enabling accurate observation of network conditions without introducing additional overhead or latency to service flows.

In the proposed system, ONT probe packets are periodically generated at the network edge at a rate of five packets per second, and injected into the transport network following the same routing and forwarding policies applied to user-plane traffic. These packets are processed by P4-programmable switches, which append per-hop telemetry information as the packet traverses the network. The metrics collected include queue-related information, such as queue and dequeue timestamps, queue depth, and ingress and exit port identifiers, allowing a precise reconstruction of the path and the queuing behavior experienced by each probe packet.

To allow for a joint analysis of transport-network conditions and the performance of core network functions, the ONT probe packets are extended with a host-level telemetry header. This header is inserted immediately after the IP layer and precedes the INT data structure. Initially, all fields of this header are transmitted with zero values, acting as placeholders for subsequent insertion of metrics collected at the host. When an ONT packet traverses the UPF effectively used for traffic forwarding (UPF1 or UPF2), an eBPF program running on that UPF intercepts the packet and populates the corresponding fields

with the current CPU and memory utilization. As a result, the telemetry information carried in the packet reflects exclusively the performance of the UPF responsible for processing the flow at that moment.

The ONT packet structure therefore consists of three main components: (i) an IP header; (ii) a host-level telemetry header, initially filled with zero values and dynamically populated by the active UPF; and (iii) an INT header, which aggregates telemetry entries collected by P4 switches along the path. This layered structure enables transport-network telemetry, gathered by programmable switches, and computational telemetry, dynamically injected into the UPF, to be conveyed within a single unified telemetry packet.

By decoupling telemetry collection from user-plane traffic and employing host-level telemetry fields that are dynamically populated by the active UPF, the proposed approach allows precise, low-overhead monitoring suitable for latency-sensitive private 5G deployments. This design allows the RL–based controller to estimate the actual state of the system, allowing more informed decisions on traffic prioritization in the transport network and dynamic UPF reassignment in the core.

3.3. Reinforcement Learning Model

The implemented RL model is based on the tabular Q-learning algorithm, specifically designed for dynamic queue priority control. It operates through a continuous cycle that begins with the reception of unified telemetry correlating the transport network queue occupancy with latency to build a discrete representation of the state of the system. The e-greedy decision policy chooses between three actions: increase, decrease, or maintain the priority of the main queue. For each action executed on the programmable switches, a multi-criteria reward function is calculated, evaluating not only the achievement of the target latency (25 ms), but also the stability and efficiency of the adjustment. Figure 2 shows the stages of the execution of the RL and provides further details on how the reward is calculated.

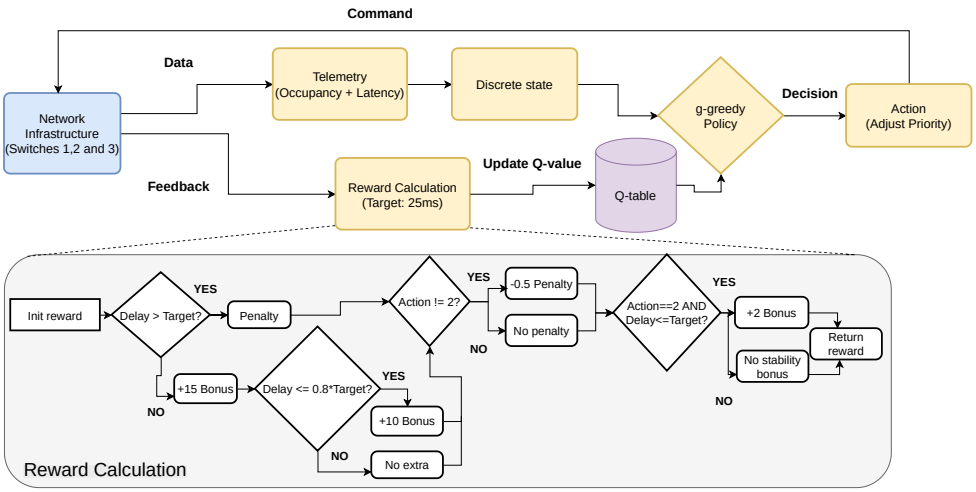


Figure 2. Details about the execution of the RL model.

The implementation adopts a lightweight multi-agent architecture, in which three independent instances of the model operate in parallel, one for each switch (**`SW1`**, **`SW2`**,

and **SW3**). Each instance maintains its own Q-table and interaction history. This separation allows for granular and specific adaptation to the particular conditions of each network path, while the model as a whole learns to map complex and non-stationary traffic and computational load patterns to optimal prioritization actions. The model's output is directly translated into executable commands in the network infrastructure, closing the automatic control loop.

In this way, the model goes beyond static approaches based on fixed rules or autoregressive prediction, offering a closed-loop online learning mechanism. It continuously adapts to inherent fluctuations, autonomously ensuring compliance with the low latency and high reliability requirements essential for critical applications.

4. Experimental Evaluation

This section evaluates the proposed approach: it describes the methodology, presents the results, and discusses the main findings and trade-offs.

4.1. Evaluation Methodology

The experimental evaluation aimed to assess how the proposed control mechanisms affect network performance, with emphasis on dynamic conditions characterized by load fluctuations in both the transport network and the 5G core. Accordingly, we defined four experimental scenarios to allow for a systematic comparison: **Scenario 1**, a baseline without control; **Scenario 2**, control driven solely by the observability of UPF; **Scenario 3**, RL-based control; and **Scenario 4**, a hybrid approach that integrates the observability of UPF with RL-based control.

In **Scenario 1**, which serves as the baseline for comparison, no control mechanism is applied. Primary and background traffic are forwarded through separate queues in the transport network, but no prioritization, dynamic adaptation, or control action is enforced in the queues or in the UPF instances. As a result, the network operates under a purely static configuration, in which the performance experienced by the primary traffic is directly affected by fluctuations in background load and by computational overload in the 5G core.

In **Scenario 2**, in addition to assigning primary and background traffic to separate queues within the transport network, we introduce continuous monitoring of UPF performance. CPU utilization is collected through eBPF, and whenever a UPF exceeds a predefined utilization threshold (60%), the controller triggers a flow-level reassignment procedure to select an alternative UPF, using the NEF API, which allows traffic to be dynamically rerouted to a different UPF instance while remaining completely transparent to the applications.

In **Scenario 3**, we introduce an RL agent to enable dynamic traffic prioritization in the transport network. The agent aims at minimizing the queueing latency experienced by the primary traffic. To this end, it observes the state of the system and selects control actions that regulate the secondary (background) queue. Specifically, the agent can decrease, increase or maintain the processing rate assigned to background traffic, thereby indirectly shaping the bandwidth available to primary traffic.

Scenario 4 combines the mechanisms evaluated in scenarios 2 and 3. Specifically, the system simultaneously performs UPF CPU utilization monitoring, triggers dynamic

UPF reassignment via the NEF API, and applies RL-driven traffic prioritization in the transport network. This scenario enables an assessment of the benefits of an integrated control strategy that coordinates decisions across both the transport plane and the core processing plane.

In all scenarios, traffic generation was performed using **`iperf3`** to produce both primary and background flows. Each run lasted 10 minutes. Primary traffic remained active for the entire duration, while background traffic followed a cyclic load pattern implemented via a script: an initial 1-minute period with no background traffic, followed by eight parallel background flows active for 2 minutes, and then another interval with no background traffic. This cycle was repeated until the end of the experiment, replicating the methodology adopted in a previous work (Simão et al. 2025) and ensuring consistency for comparative analysis.

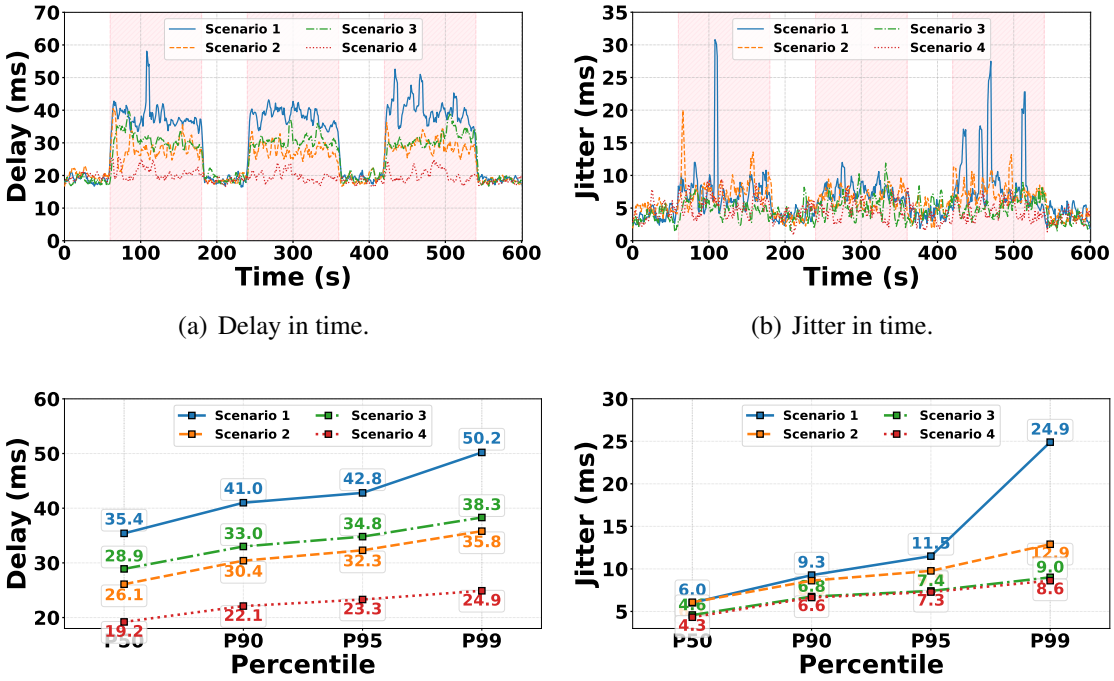
To further stress the system under realistic adverse conditions, we introduced two additional impairments only during intervals in which background traffic was active. First, we used **`stress-ng`** to impose a controlled computational load on UPF1, thereby emulating CPU overload conditions. Second, we configured an artificial one-way delay of 10 ms in the transport network using the Linux **`tc`** tool. Both impairments were deliberately synchronized with background traffic activity to reproduce realistic congestion and processing bottlenecks, while maintaining stable conditions during periods without background traffic.

4.2. Results

The results obtained for the four scenarios are presented in Figure 3, which illustrates the behavior of latency and jitter over time. All results shown in Figure 3(a) and Figure 3(b) are calculated using a moving average over the last 20 measurement points, which smooths short-term fluctuations and highlights the underlying performance trends of each strategy. Figures 3(c) and 3(d) depict the percentile values of the results obtained for delay and jitter, respectively.

In Scenario 1, the highest variability is observed, with an average latency of 30.81 ms and maximum values reaching 58.10 ms. This scenario also exhibits the highest average jitter (6.57 ms), highlighting the impact of background traffic, induced transport delay, and computational overload when no control mechanism is used. Scenario 2 demonstrates that dynamic UPF reassignment reduces the average latency to 24.72 ms, corresponding to a reduction of 19.77% compared to Scenario 1. However, the average jitter is only marginally reduced, reaching 6.16 ms (a 6.24% improvement). These results suggest that, although UPF reassignment effectively mitigates computational bottlenecks, it has only a marginal effect on transport-layer variability in the absence of any traffic-prioritization mechanism. RL-driven traffic prioritization leads to a significant reduction in jitter, as can be seen in Scenario 3. The average value decreases to 4.70 ms, a 28.46% improvement relative to the baseline. The average latency is reduced to 26.10 ms, but remains higher than in Scenario 2. This behavior suggests that transport-level control effectively stabilizes packet delivery, but cannot fully compensate for computational overload at the UPF when dynamic reassignment is not employed.

The best overall performance is achieved in Scenario 4, which combines both control mechanisms. In this scenario, the average latency is reduced to 19.60 ms and the



(a) Delay in time.

(b) Jitter in time.

(c) Delay percentiles ($P_{50}, P_{90}, P_{95}, and\ P_{99}$).

(d) Jitter percentiles ($P_{50}, P_{90}, P_{95}, and\ P_{99}$).

Figure 3. Results for delay and jitter metrics over time in each scenario.

average jitter to 4.52 ms, corresponding to reductions of 36.37% and 31.20%, respectively, compared to the baseline (Scenario 1). These results demonstrate that coordinated control across the transport network and the 5G core enables more effective mitigation of both network congestion and computational bottlenecks. Table 2 summarizes the average delay and average jitter observed in each scenario, together with relative improvements relative to the baseline.

Table 2. Latency and jitter across all evaluated scenarios.

Scenario	Average Delay	Average Jitter
1	30.81 ms	6.57 ms
2	24.72 ms (-19.77%)	6.16 ms (-6.24%)
3	26.10 ms (-15.29%)	4.70 ms (-28.46%)
4	19.60 ms (-36.37%)	4.52 ms (-31.20%)

Across all percentiles, both delay and jitter increase as expected, but the different scenarios exhibit significant differences in their tail behavior. In terms of delay, Scenario 4 consistently exhibits the smallest values, increasing only slightly from 19.2 ms at P_{50} to 24.9 ms at P_{99}., whereas Scenario 1 shows the greatest overall latency and the most pronounced tail, with delay values rising from 35.4 ms at P_{50} to 50.2 ms at P_{99}. Scenarios 2 and 3 present intermediate results, with Scenario 2 slightly outperforming Scenario 3 in all percentiles (e.g., 35.8 ms vs. 38.3 ms in P_{99}). In terms of jitter, Scenario 4 exhibits the

highest stability, increasing only from 4.3 ms (P_{50}) to 8.6 ms (P_{99}), and Scenario 3 shows a comparably stable behavior (4.6 to 9.0 ms). Scenario 2 demonstrates a moderate level of jitter, rising from 5.6 ms at P_{50} to 12.9 ms at P_{99}. In contrast, Scenario 1 exhibits a high spike at the upper percentiles, with jitter increasing from 11.2 ms (P_{95}) to 24.9 ms (P_{99}), which reflects substantial variability under worst-case conditions.. Scenario 4 consistently has the lowest delay and jitter, while Scenario 1 shows the worst tail (especially at P_{99}).

To further illustrate the system behavior under the hybrid control strategy, Figure 4 presents the throughput observed at UPF1 and UPF2 in Scenario 4. The figure highlights the moment at which dynamic UPF reassignment is triggered. During periods of background traffic, when CPU overload is induced at UPF1, a clear reduction in throughput at UPF1 and a corresponding increase at UPF2 can be observed. This behavior confirms that the controller successfully detects the overload condition and redirects traffic to an alternative UPF instance via the NEF API, without interrupting the ongoing session. The smooth transition between UPFs indicates that the reassignment mechanism operates transparently and contributes directly to the improved latency and jitter results observed in Scenario 4.

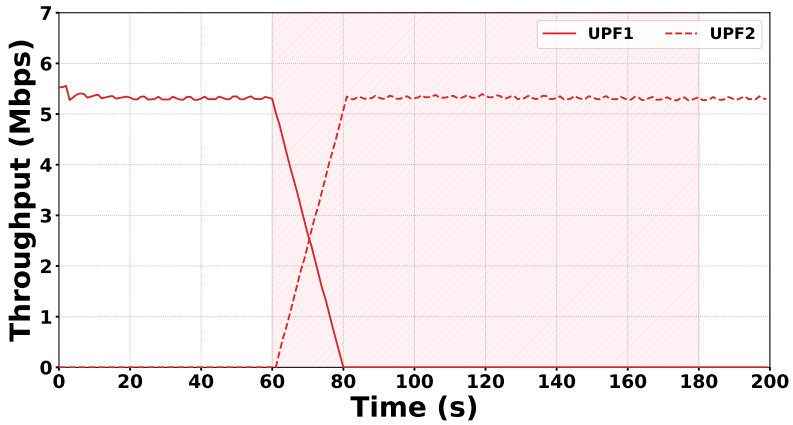


Figure 4. Throughput observed at UPF1 and UPF2 in Scenario 4.

In general, the results confirm that the evaluated control mechanisms are complementary. RL-driven traffic prioritization primarily improves transport-level stability, while dynamic UPF assignment addresses computational overload in the core. Their combination results in the lowest latency and jitter values, reinforcing the importance of a closed-loop management approach that jointly considers multiple domains of private 5G network infrastructures.

4.3. Discussion

The evaluation carried out in this work considered four scenarios designed to compare increasingly capable closed-loop control strategies under dynamic stress (cyclic background traffic, with induced CPU overload and delay in the transport network during periods with background traffic). In the static baseline scenario, without using any control mechanism, the results presented the worst performance (30.81 ms average delay, 6.57 ms average jitter and high variability). By adding eBPF-based UPF CPU monitoring and NEF-triggered UPF assignment when utilization exceeded 60%, the average delay

improved to 24.72 ms (-19.77%), but only slightly reduced the jitter to 6.16 ms (-6.24%), indicating limited stabilization of the transport-network. We also applied RL to prioritize primary traffic in the transport network, resulting in a significant reduction in jitter to 4.70 ms (-28.46%) but a reduced delay improvement to 26.10 ms (-15.29%), reflecting the sensitivity to core-side overload without dynamic UPF assignment. Lastly, we combined both mechanisms, achieving the best overall results with an average delay of 19.60 ms (-36.37%) and an average jitter of 4.52 ms (-31.20%), while the throughput traces confirmed a seamless traffic migration from UPF1 to UPF2 during overload, demonstrating that coordinated transport-and-core control mitigates both congestion and computational bottlenecks most effectively.

5. Conclusion

The main contributions of this work are threefold. Firstly, we design and implement a closed-loop management architecture for private 5G networks that jointly considers transport-network telemetry and observability of core network functions. The proposed system integrates ONT telemetry collected via P4 with UPF-level CPU metrics extracted using eBPF, enabling a unified and fine-grained view of both network and computational resources. Secondly, we formulate the resulting control problem as an RL task, in which heterogeneous telemetry metrics define the environment state and control actions dynamically steer traffic across the transport network and multiple UPF instances to minimize end-to-end latency. Finally, we experimentally evaluate the proposed approach in a private 5G testbed, demonstrating that combining network and UPF-level observability enables more effective traffic balancing and latency reduction compared to baseline and single-domain control strategies.

Future work should scale and stress-test the RL controller in more realistic private 5G settings (slicing, mobility, bursty traffic), expand observability and control knobs beyond UPF CPU and queue occupancy, explore safer and more sample-efficient RL methods with service level objective (SLO) guarantees, and validate deployability through overhead, security, and production-like testbed evaluations. Moreover, we plan to address some limitations of the current study. In particular, to provide a more detailed characterization of the RL model, including the definition of states, actions, and reward design, and to extend the experimental evaluation to larger and more heterogeneous scenarios. We also intend to include comparisons with existing approaches and further analyze the overhead and integration aspects of the proposed architecture within standard 5G environments.

Acknowledgments

This work is supported by EMBRAPII (BFA 2301.0001), Cisco, Prysmian, and MPT Cable. The authors also thank CNPq (307108/2025-2, 404509/2025-8), CPQD, Inatel, Taggen, Data Machina, IFPB Innovation Hub, and DIPPED IFPB Campus João Pessoa.

References

Abranches, M., Michel, O., Keller, E., and Schmid, S. (2021). Efficient Network Monitoring Applications in the Kernel with eBPF and XDP. In *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 28–34.

Eswaran, S. and Honnavalli, P. (2022). Private 5G networks: a survey on enabling technologies, deployment models, use cases and research directions: Private 5G networks: a survey on enabling technologies, deployment models, use cases and research directions. *Telecommun. Syst.*, 82(1):3–26.

ETSI (2026). 5G; System architecture for the 5G System (5GS). Technical Specification ETSI TS 123 501, European Telecommunications Standards Institute (ETSI). 3GPP TS 23.501 version 19.6.0 Release 19. Accessed: 2026-01-31.

Goshi, E., Jarschel, M., Pries, R., He, M., and Kellerer, W. (2021). Investigating Inter-NF Dependencies in Cloud-Native 5G Core Networks. In *2021 17th International Conference on Network and Service Management*, pages 370–374.

Hoa, L. C., Dang, T. C., and Vo, V. M. N. (2025). An integrated model of threshold-based scaling and fractional admission controlling to improve resource utilization efficiency in 5G core networks. *PLOS ONE*, 20(8):1–21.

Hyun, J., Van Tu, N., and Hong, J. W.-K. (2018). Towards knowledge-defined networking using in-band network telemetry. In *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7.

Pizzutti, M. and Schaeffer-Filho, A. E. (2019). Adaptive Multipath Routing based on Hybrid Data and Control Plane Operation. In *IEEE Conference on Computer Communications*, pages 730–738.

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*, pages 1–6.

Puttlitz, C. and Schaeffer-Filho, A. (2025). P4eBalancer: Leveraging P4 and eBPF for Optimized Load Balancing with Network and Host Insights. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 392–405. SBC.

Simão, V. S., Monteiro, L. V., Gomes, R. D., de Almeida, L. C., and Maciel, P. D. (2025). Closed-Loop Network Control for Industrial Edge Computing: A Telemetry-Driven and AI-Based Approach for Latency-Critical Applications in Private 5G. *IEEE Networking Letters*, pages 1–1.

Tajbakhsh, H., Parizotto, R., Schaeffer-Filho, A., and Haque, I. (2026). Reinforcement Learning-Based In-Network Load Balancing. *IEEE Transactions on Network and Service Management*, 23:1100–1111.

Wu, D., Li, J., Ferini, A., Xu, Y. T., Jenkin, M., Jang, S., Liu, X., and Dudek, G. (2023). Reinforcement learning for communication load balancing: approaches and challenges. *Frontiers in Computer Science*, Volume 5 - 2023.

Zheng, C., Rienecker, B., and Zilberman, N. (2023). QCMP: Load Balancing via In-Network Reinforcement Learning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, FIRA '23, page 35–40, New York, NY, USA. Association for Computing Machinery.