SDN-Based Slicing Testbed for 5G Networks

Pablo Bertrand
Universidad de la República
Montevideo, Uruguay
pbertrand@fing.edu.uy

Claudina Rattaro
Universidad de la República
Montevideo, Uruguay
crattaro@fing.edu.uy

Antonio Hidalgo
Universidad de Extremadura
Cáceres, Spain
ahidalgof@unex.es

David Cortés-Polo
Universidad de Extremadura
Cáceres, Spain
dcorpol@unex.es

Jesús Calle-Cancho
Universidad de Extremadura
Cáceres, Spain
jesuscalle@unex.es

***Abstract*—The evolution of mobile networks to 5G has introduced a key concept such as network slicing which allows network providers to create multiple logical networks over a physical infrastructure to satisfy the growing user demand on the network. In this work, we present an open-source, SDN-driven testbed for 5G standalone network slicing that enables dynamic throughput management via both reactive and predictive controllers. Hosted on Proxmox VE, our architecture combines Open5GS for the 5G core, UERANSIM and srsRAN for RAN emulation and real-world SDR connectivity, OpenDaylight as the SDN controller, and Open vSwitch for data-plane slicing. Two adaptation strategies—a threshold-based allocator and an LSTM-based predictor trained on VoD traffic—are evaluated through throughput measurements. Our results demonstrate that both schemes enforce slice isolation and guarantee prioritized bandwidth under variable loads, with the LSTM approach offering smoother performance during traffic surges. This modular platform opens a reproducible path for rapid prototyping of slice orchestration policies and paves the way for advanced AI/ML-driven slicing.**

***Index Terms*—Network Slicing, 5G, Testbed, SDN, LSTM**

I. Introduction

The evolution of 5G networks has introduced network slicing, a paradigm that enables the creation of multiple virtual network instances over a shared physical infrastructure, each tailored to specific application or service requirements. This approach is essential for addressing the diverse demands of modern wireless applications, particularly the three main use cases defined by the International Telecommunication Union (ITU): enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and Massive Machine-Type Communications (mMTC) [1].

According to the 3rd Generation Partnership Project (3GPP), a network slice is defined as a logical network that provides specific network capabilities and characteristics, supporting various service properties for network slice customers. The 5G system is designed to offer optimized support for a wide variety of communication services, different traffic loads, and diverse end-user communities. For instance, services utilizing network slicing may include Vehicle-to-Everything (V2X) communications, seamless eMBB services with Fixed Mobile Convergence (FMC), and massive IoT connections. Moreover, it enables emerging applications such as real-time cloud gaming, 8K video streaming, immersive augmented and virtual reality (AR/VR) experiences, and live broadcasting over mobile networks—use cases that demand high bandwidth, ultra-low latency, and consistent quality of service [2].

The management and orchestration of these network slices are crucial for operators to effectively support their communication services. In addition, they enable new business models by allowing network resources to be allocated and monetized on-demand, tailored to the specific needs of vertical industries and third-party service providers [3]. Key technical aspects of network slice management include:

- Service Level Agreement (SLA) requirements: defining and ensuring compliance with the performance metrics agreed upon with network slice customers.
- Performance measurements and Key Performance Indicators (KPIs): monitoring network slices to ensure they meet the desired performance standards.
- Management data analytics: utilizing data to optimize network slice performance and resource allocation.
- Closed-Loop Communication Services Assurance: Implementing automated feedback mechanisms to maintain and improve service quality.
- Management of non-public networks: handling private networks that utilize network slicing for specialized services.
- Network slice security: ensuring robust security measures are in place to protect each network slice.

To implement network slicing efficiently, Software-Defined Networking (SDN) provides a programmable control mechanism that abstracts and centralizes network management. Together with Network Function Virtualization (NFV), SDN enables the decoupling of the control plane from the data plane, allowing for highly flexible and dynamic resource allocation [4]. Each network slice can be independently configured to meet specific performance requirements—such as latency, bandwidth, reliability, and security—making it suitable for a wide range of use cases [5]. This capability is particularly crucial in scenarios where multiple services coexist and real-time adaptation of network resources is required to maintain quality of service (QoS). By dynamically adjusting bandwidth and priority levels across slices, SDN enhances the responsiveness and efficiency of the network infrastructure.

While network slicing provides the architectural foundation for service differentiation, the dynamic provisioning of slices—especially in response to fluctuating demand and heterogeneous service requirements—remains an open research challenge. Efficient resource allocation mechanisms must account for varying traffic patterns, latency constraints, and priority levels in real time. In this context, artificial intelligence (AI) and machine learning (ML) have emerged as a promising approach to enhance decision-making, automate orchestration, and improve the scalability of slice management.

In this work, we present a modular and fully open-source 5G testbed that leverages SDN-based network slicing to dynamically allocate resources. The testbed is built on Proxmox as a virtualization environment and integrates Open5GS as the 5G Core Network,

OpenDaylight (ODL) as the SDN controller, Open vSwitch (OVS) for managing traffic flows, UERANSIM to simulate the Radio Access Network (RAN), and srsRAN to support external gNB connectivity with real-world traffic and commercial User Equipment (UEs).

A key innovation of this testbed is its adaptive slice throughput management, controlled via SDN. Two adaptation strategies are implemented:

- Threshold-based approach: adjusting slice bandwidth based on predefined utilization limits.
- Machine learning-based approach: employing Long Short-Term Memory (LSTM) neural networks to predict traffic patterns and proactively adjust resource allocation.

These approaches ensure efficient resource utilization and maintain performance isolation, guaranteeing that high-priority slices receive sufficient bandwidth even under network congestion. The open and extensible nature of this platform makes it a valuable tool for advancing research in AI-driven network orchestration and reproducible 5G experimentation.

This paper aims to demonstrate how SDN-based control combined with open-source 5G components can enable dynamic, intelligent, and reproducible slice management. By comparing reactive and predictive allocation strategies, we assess their performance in terms of resource efficiency and service differentiation.

The remainder of this paper is organized as follows: Section II discusses related work on network slicing and SDN-based resource allocation. Section III details the testbed architecture, including hardware and software components. Section IV explains the slice adaptation mechanisms. Section V presents conclusions and future directions.

II. Related Work

Recent years have seen significant progress in the development of 5G slicing testbeds and SDN-based slicing mechanisms. Several research groups have demonstrated the feasibility of dynamic resource allocation and QoS assurance using SDN/NFV paradigms. Below, we highlight some of the most relevant and comparable initiatives to our own work.

Pineda *et al.* [6] present one of the first comprehensive open-source deployments of a 5G Standalone (SA) testbed with SDN capabilities. Their platform integrates Open5GS for the 5G Core, UERANSIM and srsRAN for the RAN and UE, OpenDaylight (ODL) as the SDN controller, and Open vSwitch (OVS) for data-plane forwarding. Two scenarios are demonstrated—one fully virtualized with UERANSIM and another hybrid setup using srsRAN over SDR hardware—showing how network slicing configuration and SDN policies can be applied to shape throughput and latency. Our architecture is directly inspired by this work, but has some key differences: we adopt Proxmox VE for virtualization to simplify VLAN and bridge management, we split control- and user-plane functions into separate VMs for finer isolation, and we augment the basic slice configuration with dynamic threshold- and machine-learning-based adaptation mechanisms.

Esmaeily *et al.* [7] introduce 5GIIK, a cloud-native SDN/NFV testbed that orchestrates E2E slices across RAN, transport, and core domains. Leveraging ETSI NFV MANO components (OSM as the orchestrator and OpenStack as the VIM), 5GIIK emphasizes multi-tenancy, real-time VNF onboarding, and dynamic slice provisioning. While 5GIIK excels at automated slice lifecycle management and supports multiple access technologies, it does not integrate a full 5G SA core or physical SDR nodes. In contrast, our testbed sacrifices NFV MANO complexity in favor of a light, SDN-centric control plane (OpenDaylight) and a fully SA stack (Open5GS + UERANSIM/srsRAN), enabling rapid prototyping of slice adaptation logic.

Ye *et al.* [8] propose a theoretical framework for E2E QoS provisioning via network slicing, introducing dynamic radio resource slicing in the wireless domain and bottleneck-resource generalized processor sharing (BR-GPS) in the wired NFV domain. Their work formulates optimization problems to allocate bi-resources (CPU and bandwidth) among service function chains, guaranteeing isolation among eMBB, URLLC, and mMTC. Although highly insightful, this approach remains analytic, without an accompanying prototype. Our contribution complements their framework by delivering an open-source testbed where these resource-slicing concepts can be experimentally validated under realistic traffic conditions.

Fanibhare *et al.* [9] present TINetS3, an SDN-driven slicing mechanism tailored to the Tactile Internet. Using OVS under OpenDaylight, they demonstrate three scenario-based algorithms (topology, service, and emergency slicing) on an emulated network, and validate performance via throughput and RTT measurements. While TINetS3 highlights the versatility of SDN for tailored slice creation, it does not incorporate a 5G SA core or end-to-end mobile traffic. Our testbed bridges this gap by integrating a real 5G core and RAN, thus enabling scenario-based SDN slicing approaches to be evaluated in a genuine 5G context.

These works collectively underline the potential of SDN-based approaches to manage 5G slicing effectively, while also identifying open challenges. Unlike existing platforms, our architecture uniquely combines a fully open-source 5G SA stack, Proxmox-based virtualization, ODL-driven SDN, and two adaptive slice control algorithms (threshold and LSTM prediction), providing a flexible and reproducible environment for 5G slice research.

III. Testbed Architecture

To evaluate the dynamic control of network slices in a 5G environment, we built an experimental testbed using open-source software components and SDN-based traffic steering mechanisms. This testbed enables the configuration, monitoring, and adaptive reallocation of bandwidth between slices, allowing researchers to prototype resource management strategies in controlled conditions. The system was designed to implement a full end-to-end network with slicing capabilities at both the core and transport layers.

The testbed integrates a mix of virtualized and physical network elements deployed over a Proxmox VE hypervisor version 8.3.5. Its architecture aligns with the 3GPP-defined slicing framework [10], supporting isolated paths for prioritized and best-effort traffic flows. The slicing decisions and throughput adjustments are made in real-time using SDN techniques and RESTful APIs, which allow external logic to enforce dynamic policies based on traffic demand and service-level intent.

A. Hardware and Software Components

The entire testbed is hosted on a single Proxmox VE node, selected for its lightweight hypervisor capabilities, native VLAN-aware bridge support, and efficient resource scheduling. The environment hosts nine virtual machines (VMs), each assigned a specific network function role in accordance with the 5G system architecture.

Open5GS [11] version 2.7.2 is used as the 5G Core (5GC) and is split into three VMs: one hosting the control plane (AMF and SMFs) and two others acting as user plane functions (UPFs), one for each network slice. This setup is fully extensible to support n slices by instantiating additional UPFs and associating them with

their respective slice-specific configurations. This separation allows us to emulate per-slice user data handling.

UERANSIM [12] version 3.2.7 provides the simulated gNB and UEs, ensuring full compatibility with Open5GS and exposing complete NAS signaling. In parallel, the testbed optionally supports a real radio access setup using srsRAN [13] and an Ettus SDR, allowing over-the-air communication with a commercial mobile phone configured with a custom SIM.

The SDN controller is OpenDaylight (ODL) version 0.8.4, running within a dedicated VM. It was installed with OVSDB support and integrated with Apache Karaf for module management [14]. The ODL controller communicates with an Open vSwitch (OVS) version 2.13.8 instance hosted in another VM, which serves as the data-plane switch. OVS [15] handles all L2 forwarding and enforces per-slice throughput control using queues and QoS rules based on the Linux Hierarchical Token Bucket (HTB) scheduler.

Each VM runs Ubuntu Server 20.04, with a custom network configuration including multiple virtual NICs connected to VLAN-tagged Linux bridges. This enables fine-grained control over traffic paths and interconnection between nodes. All VMs were configured with static IPs and persistent NAT rules, and SSH was enabled to allow centralized orchestration. Proxmox's interface is used to manage MAC address mapping, ensuring consistent network interface assignment across reboots.

Special attention is given to the integration of components: OVS ports are dynamically bound to specific queues, and flow rules are added to associate ingress traffic with the appropriate QoS policy. The REST APIs exposed by ODL are validated using Postman and then consumed via custom Python scripts for real-time control. Two adaptation schemes are implemented: one uses static thresholds to trigger bitrate changes, while the other relies on an LSTM model trained on YouTube VoD traffic to forecast demand and proactively reassign bandwidth.

B. Architecture

The architecture diagram (see Fig. 1) illustrates the logical separation of slices, traffic paths, and control elements. All UEs are connected to either the simulated or real gNB, and their traffic is routed through OVS, where it is tagged and classified. Two slices are defined:

- Slice 1: A high-priority slice initialized with a 30 Mbps maximum bitrate.
- Slice 2: A best-effort slice with an initial 10 Mbps cap.

Each slice's user-plane traffic is directed to a dedicated UPF, allowing per-slice enforcement of QoS policies. The OVS switch is the convergence point for all data traffic and is configured with multiple Linux-HTB queues. ODL manages these queues and their associated max-rate settings via the OVSDB protocol.

The configuration of the adaptation mechanisms required significant coordination between the data and control planes. Queue and flow rule creation was carefully orchestrated to ensure that in-port identifiers, MAC bindings, and VLAN tags aligned across Proxmox, OVS, and OpenDaylight. Adaptation logic running externally communicates with the controller's REST API to measure throughput on a per-flow basis and to dynamically adjust queue settings.

The inclusion of a real gNB setup using srsRAN version 24.20.1 added further complexity, as it required hardware synchronization, USB SDR configuration, and interface bridging across physical and virtual environments. This hybrid model allowed us to validate the slicing functionality with real radio traffic while maintaining the flexibility of simulation.

The result is a modular, reproducible testbed that can be used to explore dynamic slice allocation strategies, QoS enforcement, multi-slice coexistence, and intelligent traffic adaptation under realistic load conditions.

IV. Slice Adaptation

A key objective of this work is to demonstrate that throughput control in a 5G slicing environment can be achieved dynamically using SDN techniques. This capability is particularly important for video-on-demand services, where maintaining consistent bitrate and minimizing buffering are critical to ensuring a high-quality user experience. To that end, the proposed testbed includes a control mechanism capable of monitoring real-time traffic flows and adapting the allocated bandwidth for each slice accordingly. The core of this mechanism is implemented in Python, leveraging the REST APIs exposed by the OpenDaylight (ODL) SDN controller.

The adaptation logic operates on the data-plane elements configured in Open vSwitch (OVS). Each slice is associated with a dedicated queue managed by OVS using Linux HTB-based Quality of Service (QoS). These queues are defined with maximum throughput values (max-rate) that determine how much bandwidth a slice can use at any given time. By adjusting these max-rates dynamically, the testbed emulates the enforcement of service-level agreements (SLAs) and prioritization policies.

The adaptation scripts interact with the SDN controller in two fundamental ways:

1) Throughput Monitoring: The script periodically issues HTTP GET requests to ODL's RESTCONF interface to retrieve per-flow and per-port statistics, including the number of bytes transmitted through each switch port. By measuring the difference in bytes over successive intervals, the script estimates the average throughput in Mbps for each slice.
2) Rate Adjustment: When a decision is made to adjust a slice's resources, the script sends an HTTP PUT request to the REST API, modifying the max-rate attribute of the queue associated with that slice. Queue identifiers and their associated ports are discovered and validated via OVS and the ODL API.

The following subsections describe two approaches implemented for slice adaptation: one based on static thresholds, and another using a deep learning model to forecast traffic demand.

A. Threshold Approach

The threshold-based method serves as a simple but effective baseline for dynamic adaptation. It relies on a predefined traffic threshold to determine whether a slice should receive more or less bandwidth.

The process begins with the Python script monitoring the byte counters of the switch port associated with a particular UPF or gNB queue. These values are collected every five seconds—a period carefully chosen to align with the refresh rate of the controller's internal statistics. At each interval, the script calculates the current throughput for the slice using:

$$\text{Throughput (Mbps)} = \frac{(B_t - B_{t-1}) \times 8}{T \times 10^6} \tag{1}$$

where B_t is the byte count at time t, and T is the sampling period (5 seconds).

The adaptation logic compares this calculated throughput with a predefined threshold (e.g., 30 Mbps). If the usage exceeds this threshold, the script increases the max-rate of the high-priority slice (e.g., from 30 Mbps to 60 Mbps), while simultaneously reducing

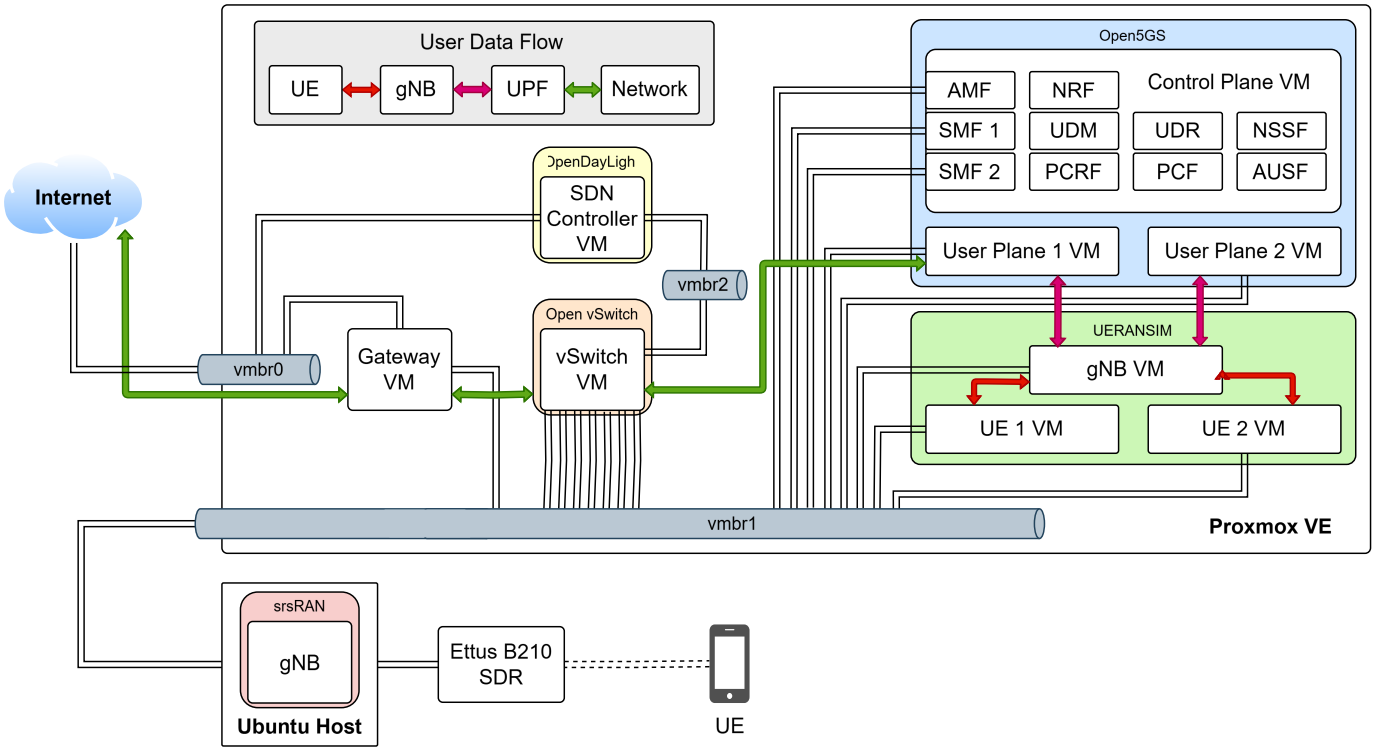

Fig. 1: Architecture diagram of the full testbed with the optional RF components.

the allocation for the best-effort slice to preserve total bandwidth constraints. If the usage is below the threshold, the rates revert to their default configuration.

This mechanism ensures that priority slices receive sufficient bandwidth during high-demand periods, while still allowing opportunistic use by lower-priority traffic when the network is underutilized. While straightforward, this approach is reactive and does not account for predicted traffic trends, making it potentially less efficient under rapidly fluctuating conditions.

B. Deep Learning Approach

To improve upon the reactivity and rigidity of the threshold model, a second mechanism was implemented using a Long Short-Term Memory (LSTM) neural network to predict the future traffic demand of each slice. This allows the testbed to proactively adjust slice bandwidth based on expected utilization rather than relying solely on instantaneous measurements.

The LSTM model was trained offline using real-world traffic traces collected during previous testbed executions. The chosen traffic type for training was Video on Demand (VoD) content from YouTube, which exhibits a semi-periodic burst pattern: sharp peaks during buffering phases, followed by sustained low-rate intervals during playback. These patterns are ideal for LSTM-based modeling, which excels at capturing temporal dependencies and long-term trends in time-series data.

Once trained, the model is integrated into the same Python adaptation script. Instead of acting on current throughput values, the script inputs a window of recent measurements into the LSTM, which returns a predicted throughput value for the next interval. Based on this prediction, the script applies an adaptive policy similar to the threshold method, but with the benefit of foresight.

For example, if the LSTM predicts that Slice 1 is about to experience a traffic surge, the system may preemptively raise its max-rate from 30 Mbps to 50 Mbps, ensuring smooth service continuity. This is particularly beneficial for services with stringent latency or buffering requirements.

The model is updated periodically with new observations, allowing it to remain relevant as traffic patterns evolve. Additionally, by shifting from reactive to predictive control, this approach reduces the likelihood of transient congestion and improves resource efficiency across slices.

Both adaptation methods are modular and extensible. They validate the ability of SDN controllers such as ODL to serve as closed-loop orchestrators in 5G slicing environments, managing traffic dynamically without disrupting existing flows.

V. Results

In this section, we validate the effectiveness of our SDN-driven slice adaptation mechanisms under realistic traffic conditions, by corroborating the testbed's capability to enforce slice isolation and to dynamically allocate resources both reactively and proactively.

All traffic traces were derived from a YouTube VoD stream, selected for its ubiquity and the clear, buffer-based nature of its traffic profile. This choice simplifies our evaluation: the threshold allocator only needs to respond to buffer-induced throughput peaks, while the LSTM predictors can focus on learning the characteristic patterns of these peaks to anticipate traffic changes.

A. Slice Isolation

To verify slice isolation, we generated both real and simulated uplink and downlink traffic on UE 1 (Slice 1) and UE 2 (Slice 2). Packet captures were collected at the two UPF VMs and at all OVS ports. Analysis of IP 5-tuple and VLAN tags confirms that:

- Slice 1 traffic traversed exclusively through UPF1 and the designated OVS interfaces, with no packets observed at UPF2 or the respective interfaces.
- Slice 2 traffic followed the symmetrical path through UPF2 and the respective interfaces, with zero leakage into the Slice 1 datapath.
- Attempts to reach the other slice's IP subnet were dropped at the OVS bridge in accordance with the installed flow rules.

These observations confirm strict per-slice isolation as enforced by our OVSDB-configured queues and flow rules (Section III.A), ensuring that no cross-slice communication is possible under either light or heavy load.

B. Threshold-based Slicing

We next evaluated the reactive threshold allocator using a step increase policy with a 30 Mbps utilization threshold. Starting from default caps of 30 Mbps (Slice 1) and 70 Mbps (Slice 2), we generated a variable traffic load on Slice 1 only. As shown in Fig. 2, each time the measured throughput for Slice 1 exceeded 25 Mbps over a 5 second interval, our Python script issued an OVSDB RESTCONF PUT to raise the Slice 1 max-rate to 60 Mbps while reducing Slice 2 to 40 Mbps; when throughput fell below the threshold, rates reverted to defaults.

The resulting rate profile exhibits clear step-shaped transitions: plateau segments at 30 Mbps followed by abrupt increases to 60 Mbps, aligned precisely with threshold crossings. These sharp adaptations demonstrate the reactivity of the threshold approach but also highlight its limitation in handling rapid load fluctuations, as transient spikes can trigger late or oscillatory adjustments.

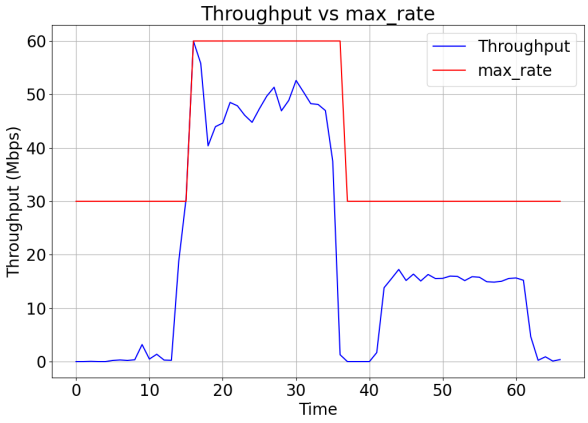

Fig. 2: Throughput over time for Slice 1 (blue) with threshold line (red). Rate adjustments at 5 second sampling intervals produce characteristic step shapes.

Moreover, this mechanism can be enhanced by defining a finer-grained hierarchy of thresholds, allowing the allocator to react more precisely to diverse traffic profiles and use cases.

C. LSTM-based Slicing

Our exploration of predictive slice adaptation began by leveraging the ODL RESTCONF API, which allowed us to query per-flow byte counts at 5 second intervals. We created a dataset using different types of multimedia contents to fit with the majority of the YouTube's contents. We then trained an LSTM model with the created data set, and evaluated its ability to forecast upcoming throughput.

The model consists of three LSTM layers of 100, 50 and 50 hidden units respectively, all of which are followed by a dropout layer, the two first ones of 0.2 and the last one of 0.5, and finally one dense output layer. An activation function (ReLU) is employed to enhance non-linear representation learning at the output of the dense layer.

As shown in Fig. 3, while the model anticipated the timing of most of the major traffic peaks, it systematically under- and over-estimated their magnitudes, reaching a root-mean-square error (RMSE) of 0.26 when using normalized data. A closer examination revealed that the 5 second sampling interval smoothed out short-lived bursts, depriving the LSTM of the fine-grained patterns essential for accurate magnitude prediction.

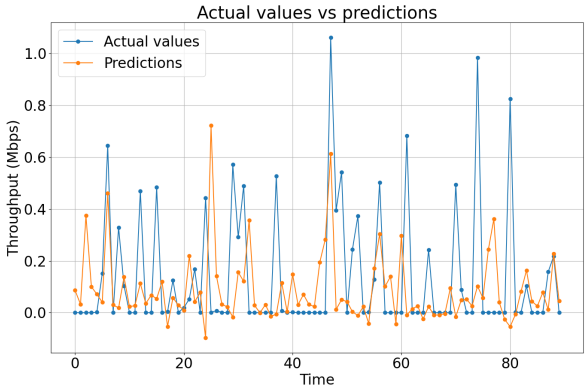

Fig. 3: Predictive performance using ODL API sampling (5 seconds); peak timing is arguably captured, but magnitude errors are pronounced.

Confronted with these limitations, we adopted a higher-resolution measurement approach using the *nload* tool, configured to sample throughput every 0.5 seconds. Incorporating these richer measurements into our training pipeline led to a neural network that exhibited markedly improved accuracy: it preserved peak timing and closely matched observed magnitudes (as illustrated in Fig. 4), reaching an RMSE of 2.6 with non-normalized data, which represents 5.25% of the data range. This practical pivot —necessary due to the constraints of the ODL API— demonstrates that sub-second sampling frequencies are crucial for constructing a predictive model capable of truly proactive and fine-grained slice resource allocation.

Collectively, these experiments demonstrate that our testbed enforces strict slice isolation, threshold-based adaptation provides a simple reactive baseline, and LSTM-based predictive control—particularly when fed with high-frequency measurements—yields smoother and more accurate resource allocations under dynamic traffic conditions.

VI. Conclusions and Future Work

In this paper, we have presented the design and implementation of an open-source SDN-based network slicing testbed for 5G Standalone environments. Built on Proxmox VE virtualization, our platform integrates Open5GS as the 5G core, UERANSIM and srsRAN for RAN simulation and real-world gNB connectivity, OpenDaylight as the SDN controller, and Open vSwitch for per-slice traffic enforcement. We demonstrated two closed-loop slice adaptation mechanisms: a reactive threshold-based controller and a proactive LSTM-driven predictor. Experimental results show that both methods effectively enforce slice bandwidth caps, guarantee priority for high-value traffic,

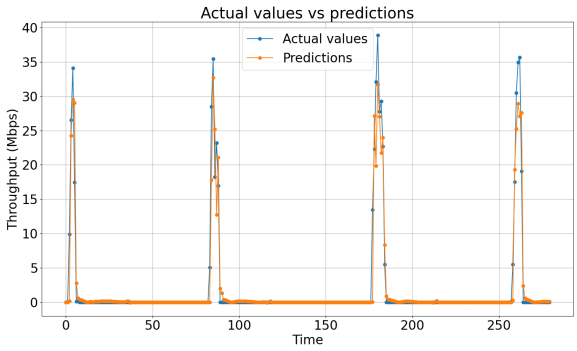

Fig. 4: Predictive performance using nload sampling (0.5 seconds); both timing and magnitude of peaks are closely tracked.

and react to dynamic load conditions with minimal disruption to ongoing flows.

Our testbed proved capable of:

- Emulating full end-to-end 5G SA slice flows over both simulated and physical RAN links.
- Dynamically reallocating bandwidth between a high-priority slice and a best-effort slice via ODL's OVSDB interface.
- Validating that LSTM prediction reduces congestion and improves overall resource utilization compared to a purely reactive scheme.

This work provides a reproducible, modular framework for researchers to prototype and compare slice control policies under realistic 5G traffic patterns.

Future work will focus on two main directions: first, evaluating the performance of the slicing control strategies under diverse traffic types, including both video and non-video applications such as IoT and cloud gaming; and second, optimizing the LSTM prediction model to reduce computational load and inference time, making it more suitable for real-time integration with the control loop. Additionally, future extensions may include integration with NFV-MANO for automated VNF lifecycle management and support for more complex multi-slice orchestration scenarios.

VII. Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Innovation under grant PID2023-151462OB-I00, by 2025 Research and Transfer Plan of University of Extremadura under grant AV-05, and by CSIC R&D project: 5/6G Optical Network Convergence: an holistic view.

References

[1] 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.

[2] 3rd Generation Partnership Project (3GPP), "5g; management and orchestration; concepts, use cases and requirements," ETSI, ETSI Technical Specification TS 128 530 V16.3.0, October 2020, 3GPP TS 28.530 version 16.3.0 Release 16. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/16.03.00_60/ts_128530v160300p.pdf

[3] M. Moussaoui, E. Bertin, and N. Crespi, "Telecom business models for beyond 5g and 6g networks: Towards disaggregation?" in *2022 1st International Conference on 6G Networking (6GNet)*, Paris, France, 2022, pp. 1–8. [Online]. Available: https://hal.science/hal-04005533

[4] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "Nfv and sdn—key technology enablers for 5g networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.

[5] H. O. Otieno, B. Malila, and J. Mwangama, "Deployment and management of intelligent end-to-end network slicing in 5g and beyond 5g networks: A systematic review," *IEEE Access*, vol. 12, pp. 190 411–190 433, 2024.

[6] D. Pineda, R. Harrilal-Parchment, K. Akkaya, A. Ibrahim, and A. Perez-Pons, "Design and analysis of an open-source sdn-based 5g standalone testbed," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

[7] A. Esmaeily, K. Kralevska, and D. Gligoroski, "A cloud-based sdn/nfv testbed for end-to-end network slicing in 4g/5g," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 29–35.

[8] Q. Ye, J. Li, K. Qu, W. Zhuang, X. S. Shen, and X. Li, "End-to-end quality of service in 5g networks: Examining the effectiveness of a network slicing framework," *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 65–74, 2018.

[9] V. Fanibhare, N. I. Sarkar, and A. Al-Anbuky, "Tinets3: Sdn-driven network slicing enabling scenario-based applications in tactile internet," *IEEE Transactions on Network and Service Management*, vol. 21, no. 4, pp. 4639–4654, 2024.

[10] 3GPP, "3gpp network slice management," https://www.3gpp.org/technologies/slice-management, accessed: 2025-04-20.

[11] S. Lee, "Open5gs documentation," https://open5gs.org/open5gs/docs/, accessed: 2025-04-20.

[12] A. Gungor, "Ueransim repository," https://github.com/aligungr/UERANSIM, accessed: 2025-04-20.

[13] S. R. Systems, "srsran documentation," https://docs.srsran.com/, accessed: 2025-04-20.

[14] O. Project, "Opendaylight documentation," https://docs.opendaylight.org/, accessed: 2025-04-20.

[15] L. Foundation, "Open vswitch project," https://docs.openvswitch.org/, accessed: 2025-04-20.