From Concept to Reality: QoS-Based RAN Slicing in Next Generation O-RAN Networks

Amjed H. Mohammed[†], Syed Tariq Shah[*], Yusuf Sambo[†], and Muhammad Imran[†]
[†]James Watt School of Engineering, University of Glasgow, Glasgow, UK
Emails: {a.mohammed.3@research.gla.ac.uk, Yusuf.Sambo@glasgow.ac.uk, Muhammad.Imran@glasgow.ac.uk}
[*]School of Computer Science & Electronic Engineering, University of Essex, Colchester, UK
Email: Syed.Shah@essex.ac.uk

***Abstract*—Service-oriented networks are gaining significant attention recently, giving rise to concepts such as network slicing within the research community. Network slicing involves partitioning the network into distinct and logically isolated segments, each customised to cater to specific services or use cases. The concept of slicing became more feasible with the emergence of the Open Radio Access Network (O-RAN), which aims to reduce RAN complexity by facilitating programmability and employing an automated control system via the RAN Intelligent Controller (RIC). This paper presents a near real time application (xApp) that runs on the RIC, designed to implement a Quality of Service (QoS)-based dynamic RAN slicing module for real-world scenarios. This slicing module assigns a downlink target throughput to each slice. It then dynamically adjusts the share of Physical Resource Blocks allocated to each slice, ensuring efficient resource utilisation and meeting Service Level Agreement requirements. The obtained results indicate that the instantiated slices can readily meet their designated target throughput in accordance with their predetermined priority.**

***Index Terms*—O-RAN, Network Slicing, Near-RT-RIC, 5G and SLA**

I. Introduction

Network slicing offers a solution for provisioning specific types of services and addressing a wide range of Quality of Service (QoS) requirements across various sectors such as smart grids, the Internet of Things (IoT), healthcare, and automotive. It involves creating multiple virtual networks on a shared physical infrastructure, categorising them based on attributes [1] like performance (data rates, latency, mobility), functionality (priority, policy control), and user types (public safety, roamers, Multimedia Priority Service (MPS) users), enabling tailored services for diverse applications and user groups within the same network infrastructure. The efficient achievement of these attributes and prevention of possible violations are crucial, necessitating mechanisms to meet the essential requirements of slices' Service Level Agreements (SLAs). Challenges of managing the scarce network resources arise due to diverse service requirements and the dynamic behaviour of the Radio Access Network (RAN), including varying channel conditions and traffic demands of mobile users, requiring an adaptive resource provisioning scheme.

The Open Radio Access Network (O-RAN) has emerged as a viable solution in recent years to facilitate the efficient openness, intelligence, disaggregation, programmability, and visualisation, which can be flexibly configured to achieve various service requirements. This contrasts with the traditional monolithic RAN, which comprises vendor-specific hardware with embedded software components. That limits network operators from integrating new components from different vendors, and coordinating between RAN nodes for crucial joint control and optimisation of RAN components [2].

The management and optimization of network functions within the O-RAN architecture [3] have been facilitated by leveraging open-closed loops enabled by two types of logical controllers: Non Real-Time RIC (Non-RT RIC), operating on a timeframe greater than 1 second, and Near Real-Time RIC (Near-RT RIC), functioning within a 10 ms to 1-second timeframe. These RICs serve as a platform for deploying RAN optimization applications called rApps (for non-RT) and xApps (for near-RT), with rApps offering policy guidance to xApps, such as managing ML models. In this architecture, the E2 interface plays a crucial role as an open interface between the Near-RT RIC and the E2 Node. The E2 Node encompasses eNBs in 4G and O-RAN Distributed Units (O-DU), as well as O-RAN Central Units (O-CU) in 5G. It facilitates the configuration of E2 nodes based on policies established by the xApps.

3GPP introduced the concept of network slicing in Release 15 [1] to manage different service requirements effectively. Building on this foundation, network slicing presents an opportunity for mobile operators to

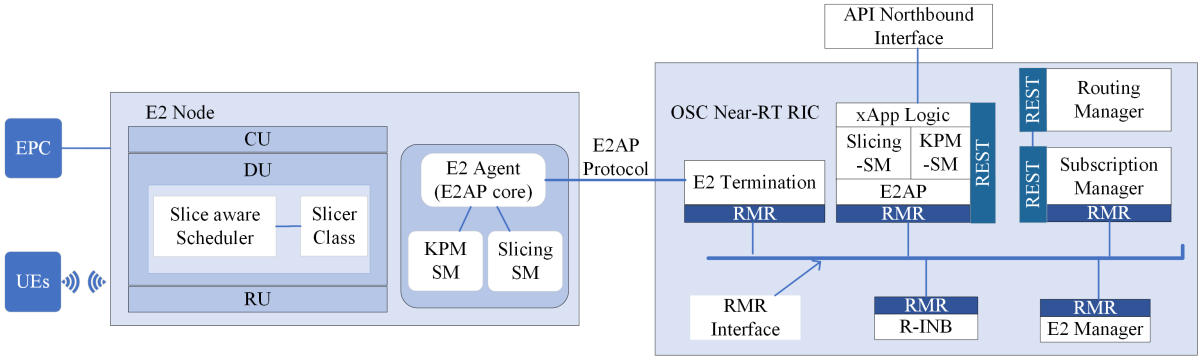

Fig. 1. A schematic representation of the experimental O-RAN framework, highlighting the main components of the near-RT RIC, along with the E2 Node, which is created by integrating the E2 Agent with srsLTE

generate additional revenue by offering tailored networks to application customers based on their specific needs. For instance, Polese et al. [3] implemented AI/ML xApps to manage RAN slicing and schedulers, utilising the Open RAN Gym platform. In [4], FlexRIC includes a slicing control xApp responsible for managing per-slice user scheduling and allocating inter-slice resources. This module sets adapted share value for each slice, ensuring efficient Physical Resource Blocks (PRBs) utilisation among slices. The SD-RAN project [5] introduced its slicing xApp, capable of assigning intra-slice schedulers and static shares of time frame rates for each created slice. Additionally, the OAIC framework [6] incorporated the NexRAN slicing xApp [7], enabling implementation on a single computing machine, offering dynamic allocation policies like Balanced Slice Throughput and Slice Throttling. These initiatives collectively represent valuable resources for the O-RAN research community, providing practical insights into the O-RAN ecosystem, xApp design, and implementation. While O-RAN research is still in its early stages, the designed xApps serve as use cases to validate the O-RAN architecture.

In this paper, we develop an xApp that can be tailored for real-world scenarios, expanding beyond exploratory purposes. In this scenario, the mobile operator, acting as the Network Slice Provider (NSP), has established an SLA with a Network Slice Customer (NSC). This SLA requires continuously available downlink capacity to meet the customer requirements, such as emergency services that demand guaranteed QoS, represented by downlink throughput per slice. Meanwhile, the NSP must ensure the efficient usage of resources while avoiding SLA violations. The xApp is validated on an experimental platform and the results show that it efficiently meets the QoS requirements, represented by the target throughput, and ensures efficient usage of PRBs. The rest of this paper is organised as follows: Section II discusses the adopted system model; Section III presents the proposed RAN slicing algorithm implemented in the O-RAN xApp; Section IV is dedicated to result analysis; and Section V provides the conclusion.

II. System Model

The system model presented herein is anchored on the Open AI Cellular (OAIC) framework [8]. Within this framework, the OAIC has integrated the RAN slicing xApp (NexRAN xApp) into its testbed, as mentioned earlier in Section I. Our model extends the functionality of this xApp to implement the proposed dynamic RAN slicing module in the downlink direction. All framework components are deployed on a single computing machine operating with Ubuntu 20.04 LTS. As illustrated in Fig. 1, the system design comprises two primary components:

A. Near-RT RIC

Based on the O-RAN Software Community (OSC)[1] near-RT RIC compliant with the E-release, the RIC is developed on Kubernetes, an open-source system for automating the deployment and management of containerised applications across various hosts. In this setup, Pods serve as the deployment units, encapsulating the RIC components that support xApp functionality. These components include the Routing Manager, Subscription Manager, Shared Data Layer (SDL), E2 Manager, and the Radio-Network Information Base (R-NIB) database. The mentioned components are responsible for a range of functions such as internal message routing, subscription management, data storage, registration and monitoring of E2 nodes, and the maintenance and monitoring of information like

[1]https://wiki.o-ran-sc.org/display/ORAN

the list of connected E2 nodes and User Equipment (UEs) within the R-NIB database. Moreover, the RIC architecture incorporates an internal messaging infrastructure supported by the RMR library.

- Slicing xApp: E2 nodes reveal their supported RAN functions to the Near-RT RIC via integrated Service Models (SMs), including report, insert, control, and policy services, with the "query" service as a recent addition [9]. These SMs, transmitted via E2 application protocol (E2AP) messages, are vital for effective RAN analysis and control. The slicing xApp is based on E2AP-V1.0, E2SM-KPM v1.0, and the RMR library from OSC, implements an extended control SM (Slicing-SM). Additionally, it is equipped with a RESTful Northbound Interface API, catering to multiple tasks such as the creation, binding, updating, and deletion of slices. The E2SM-KPM facilitates the reception of periodic E2 indication messages from the E2 Node through the report service, encompassing downlink and uplink measurements for each slice and UE within each report timeframe, including the amount of transmitted traffic data and PRB usage, among others. We have expanded this SM to include a new metric: the count of allocated subframes for each report timeframe. Originating from [7], Slicing-SM features a control service that interprets instructions from the API northbound interface and dynamically adjusts slice allocation shares following the proposed slicing policy.

B. E2 Node

The platform implements 4G RAN using srsLTE with an integrated E2 agent, establishing an E2 node. Virtual radios utilise the ZeroMQ library, and the system connects to a 4G core network. The E2 agent manages E2AP callback messages and is equipped with KPM-SM and Slicing-SM, which enable interaction with available RAN functions and the creation of KPM indication messages to be sent to the xApp.

- Extension of Packet Scheduler: The Medium Access Control (MAC) layer of E2 Node hosts a slice-aware scheduler, which includes an extended class for interfacing with the E2 agent and receiving real-time updates of slice configurations. Upon receiving the xApp control message via Slicing-SM, containing the share values for each slice indicating the percentage of subframe per allocation round, the class determines the allocation round by computing the Greatest Common Divisor (GCD) of the slices' shares. For instance, with two slices having shares of (60:40), the GCD(60:40) = 20, which leads to an efficient distribution across a round of 5 subframes, adjusting the shares to 3:2. Subsequently, the extended class interacts with the E2 Node to retrieve the bounded UE Radio Network Temporary Identifier (RNTI) of each slice. Finally, the class report the list of RNTIs to be serviced in each subframe to the downlink packet scheduler.
- Downlink Packet Scheduler: The E2 Node MAC scheduler adopts the Round Robin (RR) scheduler algorithm to allocate Physical Resource Blocks (PRBs) to the UEs in each Transmission Time Interval (TTI), which also represents one subframe duration. During each TTI, the scheduler estimates the number of Physical Resource Groups (PRGs) required to be allocated to each user based on factors such as available bandwidth, signal power, Modulation Coding Scheme (MCS), and Transport Block Size (TBS) [10]. By enabling RAN slicing, the scheduler determines which UEs to schedule in each TTI based on the reported Radio Network Temporary Identifiers (RNTIs) from the extended slicing class.

III. The Proposed QoS-Based Slicing Policy

Our proposed slicing policy aims to allocate target throughput to all slices except for the General slice, which is akin to a primary bank account for radio resources, facilitating borrowing or lending resources, among other slices. During the slice creation process, each slice is identified by its name along with the specified target throughput, both of which are transmitted via the slicing xApp's northbound interface. This target throughput signifies the maximum data transfer per second designated for the slice. The xApp is responsible for ensuring an adequate supply of PRBs to fulfil this target whenever necessary. When a slice necessitates additional PRBs to achieve its target throughput, these resources are reallocated from the General slice. In contrast, if the demand within a slice falls short of its target throughput, the xApp evaluates the availability of free (unscheduled) PRBs and, if available, reallocates them to the General slice to enhance its capacity.

Due to a limited number of PRBs within the RAN, achieving targeted throughput for all slices may pose challenges. The overall capacity of the radio cell is finite, imposing constraints on the sum rate. While an NSP can allocate target throughputs to slices based on an understanding of cell capacity, fluctuations in channel conditions lead to variable spectral efficiency of the PRBs, which lead to variable cell sum rates over time, introducing uncertainty into this allocation

process. To tackle these challenges, we propose an adaptive solution, which is a flexible strategy that prioritises the distribution of required PRBs among slices. This approach involves assigning a percentage share of PRBs to each slice, customised to achieve the target throughput according to priority classification when slices compete for PRBs.

A. *Slice Handling Algorithm*

As the process of donating PRBs to the General slice has priority over the borrowing process. Upon receiving the KPM report that containing information about the created slices, denoted as $S = \{S^1, \ldots, S^I\}$, i represents the index of each created slice in the E2 Node, where i ranges from 1 to I, the xApp is triggered to categorize these slices into three distinct sets based on their current reported throughput ($\mathcal{T}^i_{\text{cur}}$) and the percentage of their free PRBs ($\mathcal{P}^i_{\text{free}}$), which can be obtained from (6) in Section B. The first set includes slices that have exceeded their target throughput ($\mathcal{T}^i_{\text{tar}}$) and donate PRBs to the General slice (S^g) so they can meet their target, this set is denoted by $\mathbb{S}_h = \{\mathcal{S}^1_h, \ldots, \mathcal{S}^J_h\}$, j represents the index of each slice that has been added to this set, where j ranges from 1 to J. The second set comprises slices that did not exceed the target throughput and possess more than 10% of their total allocated PRBs ($\mathcal{P}^i_{\text{total}}$) as free. This 10% margin contributes to preventing throttling in case more traffic demand is requested during the next report time frame, this set also donates PRBs to S^g, and it is denoted by $\mathbb{S}_f = \{\mathcal{S}^1_f, \ldots, \mathcal{S}^N_f\}$, n represents the index of each slice that has been added to this set, where n ranges from 1 to N. The third set includes slices that have free PRBs less than 10%, they borrow PRBs from S^g or other slices based on their priority values. This set is denoted by $\mathbb{S}_l = \{\mathcal{S}^1_l, \ldots, \mathcal{S}^M_l\}$, m represents the index of each slice that has been added to this set, where m ranges from 1 to M. Algorithm 1 describes the slice categorization process.

B. *Computation of Slice Share Values*

After slice categorisation, the next step involves deriving updated share values for each network slice and communicating them to the E2 Node via the Slicing-SM. These messages encapsulate the slice names and their corresponding adjusted share values.

In our adapted model, PRB allocation is subframe-based, with each subframe entirely allocated to a single slice, and each slice is assigned a percentage share value. The E2 Node establishes an allocation round based on these shared values, as described in Section II. This round is defined by the number of subframes, where the duration of the round is equal to the sum of these subframes, each lasting 1 ms. Consequently, slices receive a proportion of subframes corresponding to their allocated share values, representing their subframes allocation within the allocation round.

Algorithm 1 Slice Classification Algorithm

Input: Parameters of $\mathcal{S}^i$ ($\mathcal{T}^i_{tar}$, $\mathcal{T}^i_{cur}$, $\mathcal{P}^i_{free}$, $\mathcal{P}^i_{total}$)
Output: $\mathbb{S}_h$, $\mathbb{S}_f$, $\mathbb{S}_l$
Initialization:
Initialize empty sets $\mathbb{S}_h$, $\mathbb{S}_f$, $\mathbb{S}_l$
Loop Process: {Iterate over slices}
for i = 1 to I **do**
  if $\mathcal{T}^i_{\text{tar}} > \mathcal{T}^i_{\text{cur}}$ **then**
    Add $\mathcal{S}^i$ to $\mathbb{S}_h$
  end if
  if $\mathcal{T}^i_{\text{cur}} < \mathcal{T}^i_{\text{tar}}$ and $(\mathcal{P}^i_{\text{free}}/\mathcal{P}^i_{\text{total}}) > 10\%$ **then**
    Add $\mathcal{S}^i$ to $\mathbb{S}_f$
  end if
  if $\mathcal{T}^i_{\text{cur}} < \mathcal{T}^i_{\text{tar}}$ and $(\mathcal{P}^i_{\text{free}}/\mathcal{P}^i_{\text{total}}) < 10\%$ **then**
    Add $\mathcal{S}^i$ to $\mathbb{S}_l$
  end if
end for

It's important to maintain the total shares of all slices to be 100 at all times, distributed among the established slices. This constant total share is essential for several reasons. Firstly, it relates to the allocation round, where each slice receives a share of subframes. The slices will sequentially consume their allocated share of subframes within this round. With unlimited total share, slices could potentially receive an unlimited share value. This scenario would lead to a proportional increase in the number of subframes per allocation round, potentially prolonging the allocation round duration and introducing higher latency as slices wait longer to receive their share of subframes. For instance, if two slices are assigned shares of 511 each, as there is no GCD between them, the allocation round will extend to 1022, necessitating a wait of 511 ms for each slice to be served. On the other hand, an excessively small total share could result in fewer subframes being distributed among slices, potentially undermining precise throughput control due to the limited amount of shared values. Moreover, maintaining a constant total share simplifies monitoring, as percentage values are more straightforward to evaluate.

Upon segregating the slices into distinct sets, the xApp logic iterates over the created sets, starting with $\mathbb{S}_h$, followed by $\mathbb{S}_f$, and concluding with $\mathbb{S}_l$. This sequencing is deliberate so that the algorithm prioritizes the distribution of resources before any borrowing occurs. The iteration over the $\mathbb{S}^h$ focuses on adjusting the share values by introducing a Share Factor ($\mathcal{SF}$) mechanism while ensuring that the minimum share

is 5% to reserve resources for accommodating new UEs. The reduced resources from the adjusted shares are subsequently allocated to the S^g. In this context, the $\mathcal{SF}^j$ for each slice j of this set is computed. Subsequently, the new share value ($\mathcal{V}^j_{new}$) is derived based on the current share ($\mathcal{V}^j_{cur}$) :

$$\mathcal{SF}^j = \frac{\mathcal{T}^j_{tar} - \mathcal{T}^j_{cur}}{\mathcal{T}^j_{cur}}, \tag{1}$$

$$\mathcal{V}^j_{new} = \max(\mathcal{V}^j_{cur} + \mathcal{V}^j_{cur} \times \mathcal{SF}^j, 5). \tag{2}$$

Then, new share value ($\mathcal{V}^g_{new}$) for S^g is calculated based on changing of the share value of slice j, denoted as ($\mathcal{D}^j$):

$$\mathcal{D}^j = \mathcal{V}^j_{new} - \mathcal{V}^j_{cur}, \tag{3}$$

$$\mathcal{V}^g_{new} = \mathcal{V}^g_{cur} - \mathcal{D}^j. \tag{4}$$

The iteration over the $\mathbb{S}_f$ aims to optimize PRB allocation when it exceeds the slice's requirements by calculating the $\mathcal{SF}^n$ for each slice n based on the free available PRBs ($\mathcal{P}^n_{free}$).

$$\mathcal{P}^n_{Total} = \frac{\mathcal{V}^n_{cur}}{\mathcal{V}_{total}} \times 2 \times U \times \mathcal{P}_{ava}, \tag{5}$$

$$\mathcal{P}^n_{free} = P^n_{total} - P^n_{used}, \tag{6}$$

$$\mathcal{SF}_n = -\frac{\mathcal{P}^n_{free}}{P^n_{total}}, \tag{7}$$

where $\mathcal{P}^n_{total}$ represents the total number of PRBs allocated to slice n during the report time frame based on its allocated share value. $\mathcal{V}_{total}$ represents the sum of shares of all slices $\sum_{i=1}^{I} \mathcal{V}^i = 100$. $\mathcal{U}$ denotes the number of subframes per report time frame, while P_{ava} signifies the number of available PRBs in each LTE slot. The value '2' indicates that there are two slots per subframe in the LTE configuration. $\mathcal{P}^n_{used}$ represents the number of PRBs used by slice n. Finally, the new share of slice n is calculated as in (2), while the change in share value $\mathcal{D}^n$ and the share value for the S^g are calculated as outlined in (3) and (4), respectively.

The iteration over $\mathbb{S}_l$ in Algorithm 2 addresses allocating PRBs to meet a slice's throughput demand when it's below the assigned target. This is crucial when slices don't exceed target throughput and consume most allocated PRBs (less than 10% are free). Determining throughput demand and allocating PRBs is challenging without traffic prediction. In such cases, the algorithm ensures efficient resource utilization by providing a tailored PRBs share to meet only required throughput. The algorithm adjusts slice share periodically by borrowing 1% from S^g while monitoring PRB utilization and throughput. If no shares are available in S^g, it iterates over other slices, starting from the least priority slice and moving up, ensuring prioritization based on slice priority levels.

Algorithm 2 Iteration over the $\mathbb{S}_l$

Input: $\mathbb{S}_l$ ($\mathcal{V}^m_{cur}$, $\mathcal{V}^g_{cur}$, $slicePriority$)
Output: $\mathcal{V}^m_{new}$

```
for m = 1 to M do
    if V^g_cur > 5% then
        V^m_new ← V^m_cur + 1
        V^g_new ← V^g_cur − 1
    else if V^g_cur = 5% then
        Arrange the slices in ascending order of their priority
        for i = 1 to I do
            if S^m ≠ S^i and V^i_cur > 5% then
                V^m_new ← V^m_cur + 1
                V^i_cur ← V^i_cur − 1
            end if
        end for
    end if
end for
```

IV. Result

The framework setup includes one E2 Node, two UEs, and an EPC core network. Each UE is associated with a particular slice. The E2 Node operates on a 10 MHz of spectrum (ie, 50 PRBs). Two slices have been established. The first slice, named Health, is tailored for an NSC that requires a guaranteed amount of throughput at all times whenever it is required, fitting its critical requirements. This is particularly vital in the health sector, which often includes emergency services that need to be prioritized on the network. The second slice, named General as previousely denoted by S^g, is designed to cover general services that do not require such prioritization, offering a standard level of service. The Health slice is targeted to achieve a 10 Mbps throughput. As the trigger for the KPM indication message is set to occur every 5.12 seconds, the time steps are multiples of this interval. The downlink data is generated using Iperf. For the S^g, Iperf is always run without a set throughput limit. However, for the Health slice, Iperf is run with various maximum throughput limits to test the slice with different traffic profiles.

Initially, the Iperf test of the Health slice is set to be unlimited. Fig. 2 illustrates the slicing policy process. When the xApp detects traffic exceeding 10 Mbps, it enforces a reduction by decreasing its share value, appending the slice to the $\mathbb{S}_h$, and applying its rules. The xApp continues to monitor the traffic, and once it falls below 10 Mbps, it reassigns the slice to the $\mathbb{S}_l$ (as there are no free PRBs due to the unlimited downlink data), increases its share by 1%, and repeats

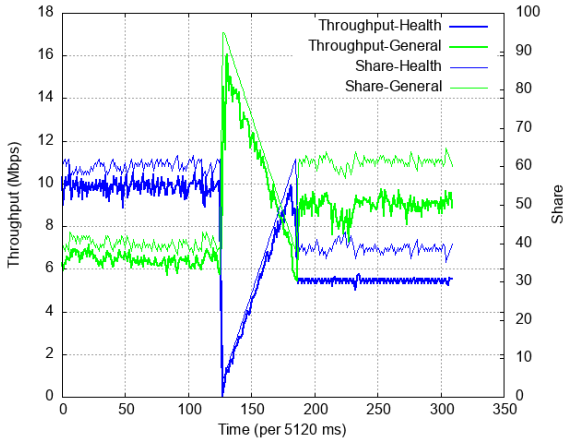

Fig. 2. Two slices with dynamic share allocation

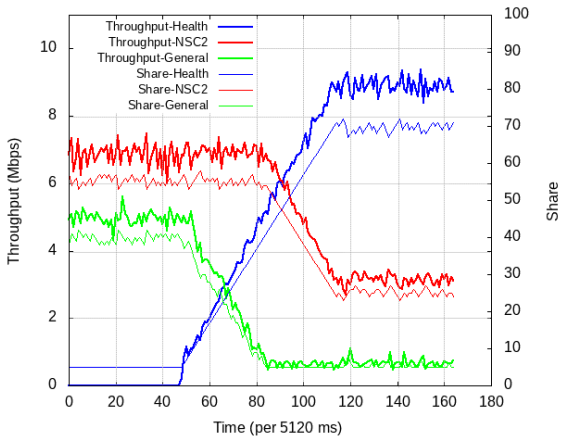

Fig. 3. Three slices with different priority

this process as needed. During this process, the share of the S^g varies inversely. Whenever the Health slice requires resources, they are taken from the S^g, and vice versa. Next, when the Iperf test of the Health slice is stopped in time step 125, its share drops down to the minimum, specifically 5%. In this case, the xApp binds the slice to the $\mathbb{S}_f$, as the allocation of downlink PRBs is almost zero due to the absence of data to be sent, resulting in a high availability of free PRBs. The rules of the $\mathbb{S}_f$ are then applied to reduce the share of the Health slice, while the S^g receives a larger share of resources, thereby increasing its throughput accordingly. This procedure is crucial to prevent wastage of resources without violating the SLA. Later, at approximately the same time, the Iperf test of the Health slice is set to 5 Mbps, which is less than the target throughput. As the slice consumes all of its available resources to reach the required throughput, it is bound to the $\mathbb{S}_l$. The share continuously increases by 1% as the xApp monitors the throughput. Whenever there are free PRBs available, indicating that the slice is receiving more share than required, the xApp will bind it to the $\mathbb{S}_f$ to reduce its share according to the availability of free PRBs, and so on.

The period between time steps 160 and 180 witnesses a throughput increase beyond the set 5 Mbps limit, is due to the Iperf test sometimes needing time to settle to its determined throughput limit. This variation in traffic is actually beneficial for the test, as it allows for consideration of different traffic patterns. During this period, we can observe that the share of resources allocated to the slice is increasing as it tries to reach its desired throughput since it has not yet reached its target throughput. Overall, the results demonstrate that the SLA is consistently met, as the slice receives all the required PRBs limited by its target throughput whenever it needs them. Additionally, when the slice consumes less traffic than the target value, its share is reduced accordingly.

Additionally, Fig. 3 illustrates the behaviour of the slicing policy when there are no available resources in the S^g. A new slice was created and added through the xApp northbound interface (NSC2) with target throughput equal to 7 Mbps and priority 2, while Health slice was set with target throughput 9 Mbps and priority 1. The Iperf of both NSC2 and S^g were set with no bandwidth boundary, allowing NSC2 to obtain its required share to achieve its target throughput. At the time, in step 50, the Iperf of Health slice is run with unlimited bandwidth. As observed, it initially starts consuming the share of the S^g gradually without affecting the share of NSC2. Until time step 85, when the share of S^g reaches its minimum, it starts consuming the share of the less priority slice (NSC2) until it reaches its target throughput in time step 120.

The observed fluctuations in traffic, as depicted in Fig. 2 and Fig. 3, result from the unreliable processing of base station data, which lacks temporal constraints or synchronization. This issue leads to a degradation in system throughput, especially noticeable when accommodating additional users, as processing times increase, resulting in a decrease in the number of processed subframes per report time frame. Conversely, Fig. 4 illustrates variations in the number of processed subframes per report time frame, influenced by the speed of handling the MAC packet scheduler when two slices are created. These variations cause fluctuations in cell-level traffic. The optimal number of subframes during the report time frame is determined to be 5120, as each subframe ideally requires 1 ms for processing within a report time frame of 5120 ms.

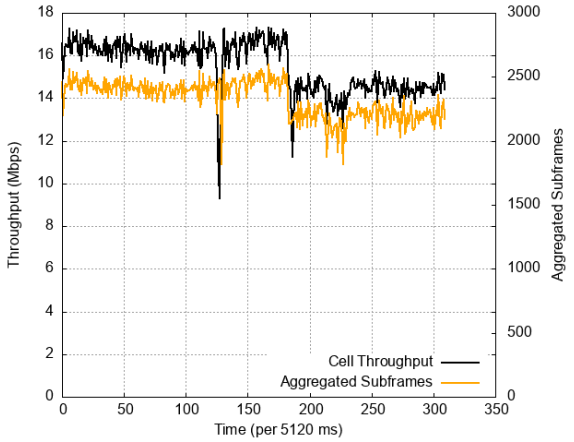

Fig. 4. The evaluation of base station performance reveals variations in the processing time of subframes over time

V. Conclusion

We have proposed a network slicing algorithm within the O-RAN ecosystem and then validated it experimentally, demonstrating its effectiveness in achieving the SLA objectives while optimizing resource utilisation. By dynamically adjusting resource allocation based on real-time traffic conditions and target throughput, the slicing policy ensures efficient utilisation of PRBs and prevents violations of SLAs. The closed control loop facilitated by the E2 interface enables seamless integration of the xApp slicing logic, allowing for dynamic control over throughput and PRBs allocation. On another hand, implementing the policy with a 5G E2 Node is potentially feasible, as the share could be mapped to the slicing-enabled MAC layer of 5G network.

References

[1] 3GPP, "Technical specification group services and system aspects; service requirements for the 5g system; stage 1 (release 15)," *Technical Specification (TS) 22.261*, https://www.3gpp.org/specifications.

[2] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[3] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, 2022.

[4] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: an SDK for next-generation SD-RANs," in *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 411–425.

[5] O. Sunay, S. Ansari, S. Condon, J. Halterman, W. Kim, R. Milkey, G. Parulkar, L. Peterson, A. Rastegarnia, and T. Vachuska, "ONF's software-defined RAN platform consistent with the O-RAN architecture," *OpenNetworkingFoundation, Whitepaper*, August 2020.

[6] P. S. Upadhyaya, A. S. Abdalla, V. Marojevic, J. H. Reed, and V. K. Shah, "Prototyping next-generation O-RAN research testbeds with SDRs," *arXiv preprint arXiv:2205.13178*, 2022.

[7] D. Johnson, D. Maas, and J. Van Der Merwe, "NexRAN: Closed-loop RAN slicing in POWDER-A top-to-bottom open-source open-RAN use case," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization*, 2022, pp. 17–23.

[8] OpenAI Cellular, "OpenAI Cellular Documentation," https://openaicellular.github.io/oaic, accessed: March 14, 2024.

[9] O-RAN Working Group 3, "O-RAN E2 Service Model (E2SM), RAN Control 3.0 ," *O-RAN.WG3.E2SM-RC-R003-v03.00, Technical Specification*, June 2023.

[10] A. Maghrabi, W. Hardjawana, P. L. Yeoh, and B. Vucetic, "An Experimental Inter-Slice RAN Controller for 4G/5G Cellular Networks," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 2021, pp. 1–6.