SERENS: Self Regulating Network Slicing in 5G for Efficient Resource Utilization

Mohit Kumar Singh
Department of CSE
IIT Hyderabad, India
cs17mtech11015@iith.ac.in

Shwetha Vittal
Department of CSE
IIT Hyderabad, India
cs19resch01001@iith.ac.in

Antony Franklin A
Department of CSE
IIT Hyderabad, India
antony.franklin@iith.ac.in

***Abstract*—5G is designed to meet the requirements of various network services such as eMBB (enhanced Mobile Broadband), URLLC (Ultra Reliable Low Latency Communications), and mMTC (massive Machine Type Communication) by making use of the technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV). These services can be provided through isolated virtual networks bringing in the concept of network slicing in 5G which helps in adjusting the resources dynamically which in turn can maximize resource utilization across the services. The dynamic adjustment of resources can be achieved by monitoring slice instances in the Closed Loop Automation (CLA) to make quick decisions on slice scaling, selection, etc. In this paper, we propose a Self Regulating Network Slicing (SERENS) framework for slice monitoring and selection in 5G. We have developed a prototype of the proposed SERENS framework in a 5G test-bed system and shown that proper slice selection can avoid wastage of resources of slices by up to 60%. The proposed slice selection algorithm will help the operator to serve a higher number of users while making the efficient usage of the available resources.**

I. Introduction

5G is being evolved from the existing 4G/LTE radio access technology replacing the monolithic network entities running on proprietary hardware with the software modules called Network Functions (NFs). The NFs are capable of running on cloud servers/commodity hardware leaving behind the legacy network's hardware dependency by making use of Software Defined Networking (SDN) and Network Function Virtualisation (NFV) like technologies. 5G system provides various services categorised into three types of generic services namely enhanced Mobile Broadband (eMBB), Ultra Reliable Low Latency Communications (URLLC), and massive Machine Type Communication (mMTC). Providing different services having different Quality of Service (QoS) and Service Level Agreement (SLA) requirements imposes a huge challenge to the Service Provider (SP), requiring a flexible network deployment. Network slicing is the key feature adopted by 5G which helps the SP in setting up different virtual networks providing specific type of service running on the shared physical infrastructure, while providing the service level isolation. If the required resources for the deployed slices are configured statically, the SP won't be able to make the best usage of the available resources due to wastage of physical resources. The dynamic allocation of the resources to the slice is based on the current load to maximise the overall revenue for the SP.

978-1-7281-7299-6/20/$31.00 © 2020 IEEE

Owing to the demand of high availability and reliability on the services, the SP would deploy multiple slice instances of the same type [1] leading to challenging and complex situation of selecting a specific slice instance to serve the incoming User Service Request (USR). Hence, the network may have multiple candidate slices capable of serving the incoming USR. 3GPP [2] defines network slice selection as the process of selecting an instance of a network slice for the incoming USR in 5G Core Network (5GC). The new incoming USR has to be associated with a network slice instance to carry forward the user centric activities on the 5GC. Performing dynamic slicing and getting the candidate slices for selecting the target slice necessitates the requirement of performing the monitoring and analyzing of network slice life cycle and load. In this work, we propose a novel SERENS framework for performing slice monitoring, analytics, and selection to achieve Self Regulating Network Slicing (SERENS) in the 5GC.

The rest of the paper is organised as follows. Starting with related work and motivation in Section II, we detail our proposed SERENS framework of 5GC in Section III which uses slice monitoring, analytics, and slice selection algorithm for efficient resource utilization. In Section IV, we demonstrate the implementation of the proposed SERENS framework in a 5G system which would be used for our performance study in Section V and conclude our paper in Section VI with the future work.

II. Related Work and Motivation

In the literature, sincere efforts have been made in making the decision of admissibility of a new slice request and a new user request. In [3], the authors proposed the admission control algorithm to perform the slice selection for the incoming tenant request based on its required SLA and available slice resource by granting the request to the least loaded candidate slice. In the same context, authors in [4] calculate the bandwidth and End-to-End (E2E) delay of the provisioned slice(s) and compared against the SLA requirement to decide on the admissibility of the new slice request. Authors in [5] predict the resource requirement of the incoming slice request and admit the new slice request only if it does not result in

degradation of the already provisioned slice(s). Authors in [6] have proposed Machine Learning (ML) based model trained on the network Key Performance Indicators (KPIs) to predict the load on the network and selecting the slice type from one of the pre-defined slice categories (eMBB, URLLC, and mMTC). In [7], the authors have proposed a slice admission control framework which makes the decision on the admissibility of the new slice request on the basis of the available resource capacity.

The decision of admissibility of the new slice request and user request urges the close monitoring of slices in 5GC. Typically, Management and Orchestration (MANO) units have the provision of reporting the performance management metrics, mainly CPU and memory of individual NFs and network services. However, monitoring and reporting these network KPIs adds to the complexity of overall slice performance management, as orchestrator is transparent to 5G system and core network functions. Additionally, no attention has been placed in literature so far, on slice selection for an incoming user request at Network Slice Selection Function (NSSF) in 5GC, after identifying the candidate slices. Hence, we propose the tightly integrated framework of network slice monitoring and analytics, capturing both the network slice and network function specific metrics to achieve self optimization and regulation in 5GC slice selection. In our previous work [8], we have addressed the issue of controlling slice activation and deactivation in 5GC, to achieve dynamic slicing. In this paper, we focus on achieving self regulation of network slicing in the 5GC using slice monitoring, analytics, and selection.

The following are the key contributions of this work.

- A framework to facilitate the self regulation of network slices called SERENS using the Closed Loop Automation (CLA) for achieving the slice monitoring, slice analytics, and slice selection in the 5GC.
- Algorithms for slice monitoring, slice analytics, and slice selection in order to study the proposed SERENS framework to optimize the usage of underlying resources.
- Implementation of the proposed SERENS framework in a 5G test-bed system as a proof of concept and evaluate the effectiveness of the proposed solutions.

III. Self Regulating Network Slicing (SERENS) Framework

In order to manage the 5G system which will have number of slices operating on the same network infrastructure, we need slice monitoring, slice analytics and slice selection in a closed loop manner. So, we propose a self regulation of network slicing in 5GC based on the CLA mechanism shown in Fig. 1. The figure depicts the close loop working of the slice monitoring, slice analytics, and slice selection functions. The monitoring includes various network and 5G KPIs such as number of users, latency, reliability, and throughput. The measurement of the slice level KPIs will be done at the Network Slice Management Function (NSMF) in 5GC. These measured KPIs are fed to the slice analytics module at NSSF of 5GC and updated with the instantaneous load information of all the available slice instances. The main functionalities of all the three components of the CLA mechanism of self regulatory network slicing are described below.

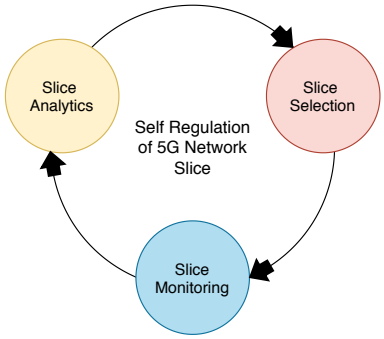


Fig. 1: Proposed SERENS Framework using the Closed Loop Automation.

A. Slice Monitoring at NSMF

The SERENS framework has the slice monitoring functionality at NSMF entity of the 5GC, responsible for monitoring all the deployed slice instances as discussed below.

- **Slice LCM Handler:** The Slice Life Cycle Management (LCM) Handler module performs the communication with the orchestrator entity over its North Bound Interface (NBI) and controls the life cycle of a network slice.
- **Slice Monitoring Module:** The Slice Monitoring module fetches the required KPIs of all the available slices such as number of registered UEs, total number of registration requests from different UEs, active UEs, and de-registered UEs at every slice instance, along with data plane throughput of the slice instance.

B. Slice Analytics at NSSF

The slice analytics functionality of SERENS framework resides in the Slice Analytics module of the NSSF entity. At a given point of time, this module performs data analytics functionality on the number of USRs received for better handling the future requests on the candidate slices. For this it continuously analyses the slice status and load information it receives from NSMF, to ensure controlling the slice activation and deactivation of the required slices on a need and timely basis. Hence, the slice information maintained by this module helps in getting the set of candidate slices for the incoming USR and effectively helps to achieve dynamic slicing.

C. Slice Selection at NSSF

In the SERENS framework, NSSF fetches the status and infrastructure KPIs such as CPU and memory usage along with 5GC KPIs of every available slice instance, at specific periodicity, from NSMF. NSSF performs the slice selection for the incoming USR, using the received slice specific information from NSMF. Thus, NSSF along with NSMF take part in deciding the appropriate slice instance for the incoming USR.

The NSSF entity of the proposed framework has four main functional modules as described below.

Algorithm 1: The proposed slice selection algorithm.

Input: Incoming Request(R_i) at time i
Result: Target Slice

$InstanceLoad[\,],\ RequestsServed[\,];$
$NumberInstances[\,],\ ExcessInstances[\,];$
$ActiveUsers[\,],\ AptSlices[\,];$
$InstanceId \leftarrow 1;$
if $i > StayDuration$ **then**
  $removeUSR(i - StayDuration);$
for $t \leftarrow i$ **to** i+TTL **by** 1 **do**
  $ActiveUsers[t] \leftarrow ActiveUsers[t] + 1;$
while $R_i > 0$ **do**
  $S \leftarrow InstanceIds\ in\ decreasing\ order\ of\ Load;$
  while $s\ in\ S$ **do**
    $t \leftarrow min(R_i, SliceCapacity - InstanceLoad[s]);$
    $R_i \leftarrow R_i - t;\ RequestsServed[i] \leftarrow (s, t);$
  if $R_i > 0$ **then**
    $InstanceLoad[\,] \leftarrow (InstanceId + 1, 0);$
    $InstanceId \leftarrow InstanceId + 1;$
$AptSlices[i] \leftarrow ActiveUsers[i] \div SliceCapacity;$
$NumberInstances[i] \leftarrow Size(InstanceLoad[i]);$
$ExcessInstances[i] \leftarrow NumberInstances[i] - AptSlices[i];$

- **Slice Request Handler:** The Slice Request Handler module of the NSSF, collects the concurrent USRs arriving to the network and assigns them to the Slice Selection Algorithm module for selecting a suitable network slice instance.
- **Slice Profile Registry:** This registry of NSSF maintains the slice instance information it obtains from NSMF, along with slice SLA and user's record of slices in a database, available to other modules of NSSF to use this stored slice information.
- **Slice LCM Controller:** This functional module of NSSF helps in managing the network slice life cycle and put the received information from Slice Selection Algorithm module into effect by informing the NSMF to trigger the corresponding change in the network slice life cycle.
- **Slice Selection Algorithm**: The Slice Selection Algorithm module of NSSF implements the slice selection scheme for making the decision of selecting the best candidate slice. It informs the Slice LCM Controller module for triggering the activation/deactivation of a new/existing slice instance dynamically, achieving dynamic slicing, while making use of the Big Data Analytics (BDA) analysis of the incoming requests.

The proposed slice selection algorithm is shown in the Algorithm 1. All the provisioned slice instances are capable of serving maximum of *SliceCapacity* number of users with the required SLA. The proposed Most Loaded Slice Selection (MLSS) scheme maps the incoming requests to the slice instances in decreasing order of their active users. With the aim of having the minimum number of instances running to serve the current traffic, the proposed algorithm ensures more users leave the network from the *least loaded instance*, leading to de-commissioning of an active slice instance with no load.

IV. Implementation of SERENS Framework in 5GC

In order to study the performance of the proposed solution, we have implemented the proposed SERENS framework in the NSMF and NSSF network entities of the 5GC, performing slice monitoring, slice analytics, and slice selection functionalities of various deployed slices. The Fig. 2 shows the deployed architecture for realising this proposed SERENS framework having the NSSF and NSMF performing self regulation of network slices. The deployed architecture consists of NSMF

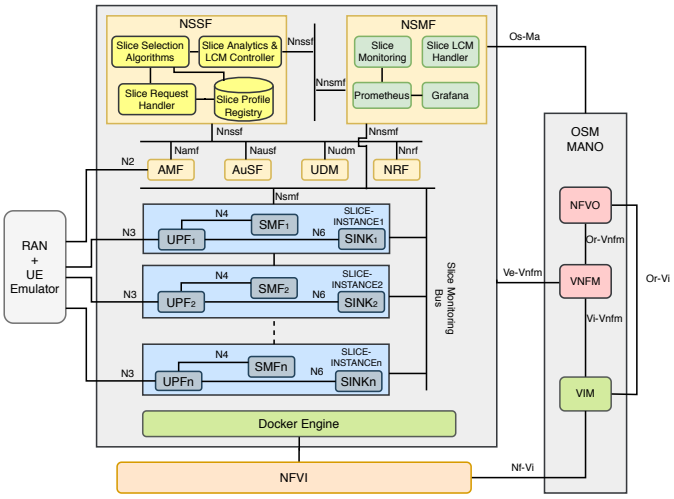
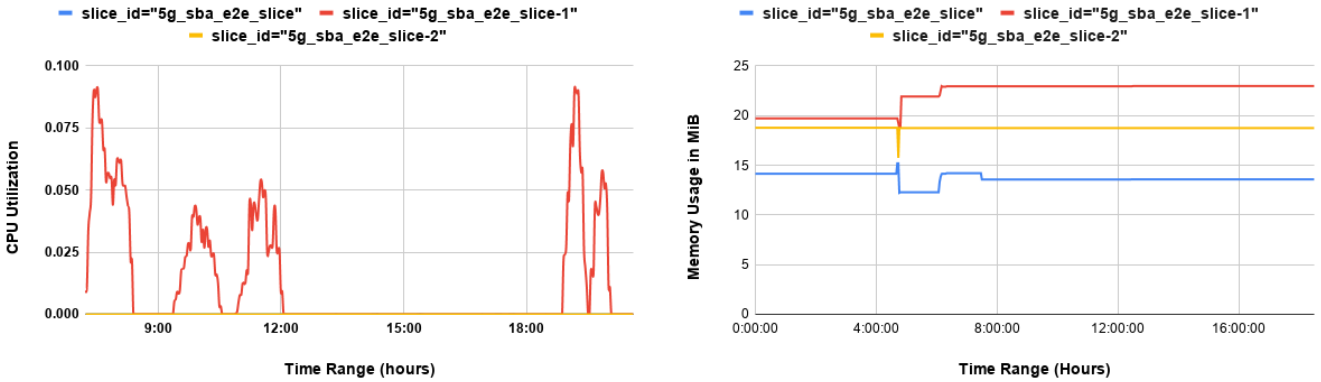


Fig. 2: Proof of concept system realising the proposed SERENS framework.

and 5GC Network Function (NF)s orchestrated using the NFV MANO functions provided by OSM [9] Rel.5. Here OSM provides the NFV Orchestration (NFVO) and Virtual Network Function Management (VNFM) functionalities that supports communicating with different Virtual Infrastructure Management (VIM)s. We have picked a light weight VIM-Emulator [10] which emulates the Openstack [11] functions for VIM named as vim-emu. Vim-emu allows the execution of real NFs packaged as docker [12] containers in an emulated network topology. NSMF utilizes OSM's North Bound Interface (NBI) taking the role of OSS/BSS, responsible for creating, deploying one or more slice instances of various types like eMBB, URLLC, and mMTC.

We have used our own 5GC prototype developed in house based on 3GPP Release 15 which comprises of 5GC control plane Network Function (NF)s listed as Access and Mobility Management Function, Network Repository Function (NRF), Authentication Server Function (AUSF), Unified Data Management (UDM), and Session Management Function (SMF) with Service Based Interaction (SBI) between them. SBI is implemented with REST APIs using HTTP/2 library from



(a) (b)

Fig. 3: Slice level resource utilisation a) CPU usage of slices and b) Memory usage of slices.

nghttp2 [13]. NRF provides service registration and discovery services to other NFs in the framework. UPF in the 5GC participates in data plane path with SINK, enabling GPRS Tunneling Protocol User plane (GTP-U) on $N3$ with RAN and $N6$ interface with SINK.

We have considered that a slice instance includes SMF, UPF and SINK NFs. AMF, NSSF, NRF, AUSF and UDM are shared among the slice instances and hence form a common slice subnet. All the NFs including RAN+UE emulator are developed as virtualized docker containers each intended to provide functionalities as micro services such as UE registration, de-registration, and end-to-end uplink and downlink data exchange over different network slices.

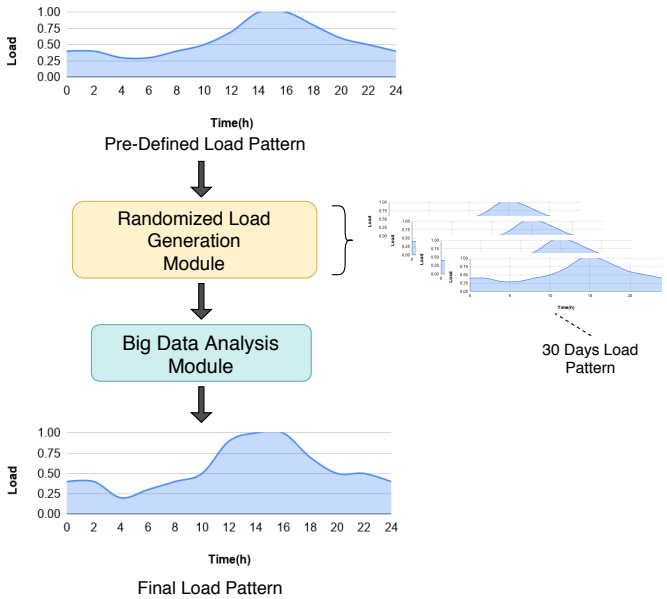


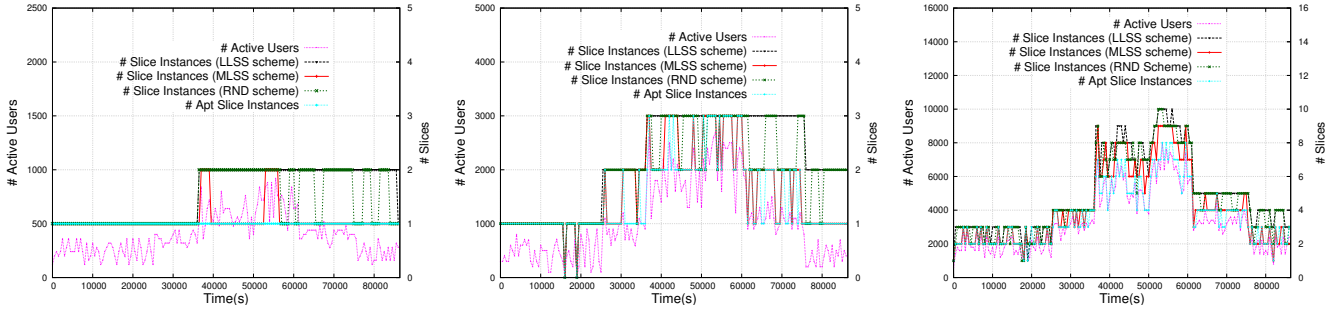
Fig. 4: Synthetic Traffic Data Generation Model.

In the deployed architecture, the NSMF first instantiates the common core network slice subnet with AMF, AUSF, UDM, and NSSF. It then instantiates and activates a set of required slice instances of various slice types. Once the common slice subnet and slice instances are activated, each of the NF in the common slice subnet and slices registers itself at NRF. Each NF is now ready to serve the traffic of 5GC control plane and data plane. Once the RAN+UE emulator function gets active, it emulates the UE activities by requesting for a specific slice service. AMF upon receiving the user request contacts the NSSF to find an appropriate target slice instance and then creates a service session for the UE on the selected target slice instance.

V. Performance Evaluation

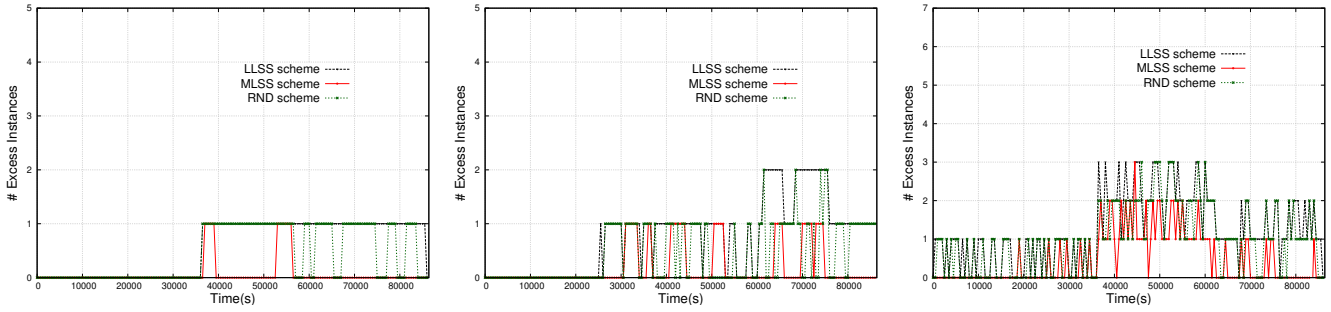
A. Synthetic Traffic Data Generation

We have used a BDA enabled simulation model for generating the data set, simulating the actual load pattern of slice requests coming to the network. Since it is difficult to get the real data set from the service providers we have used the BDA based model for generating an incoming requests load pattern as shown in Fig. 4. The Randomized Traffic Generator module of the model generates the load pattern of 30 days by using the input traffic profile. The BDA module of the model generates the traffic load pattern profile by performing the average value based analysis on the provided 30 days load profile. The predicted load (in range of 0-1) from the model is scaled with a constant factor to get the numeric value of the incoming requests and observed at a time interval of every 8 minutes. Each of the incoming slice request has a Time-To-Live (TTL) field [6] which specifies the time duration for which a request stays in the network. The TTL field value is specific to the type of service being requested. We have considered the TTL value for eMBB, URLLC, and mMTC services in the range of 160-300 Secs, 150-200 Secs, and 60-100 Secs, respectively. The observed incoming requests pattern with their TTL value are used for generating the number of active users for each of the generic slice types (eMBB, URLLC, and mMTC) as shown in the first plot of Fig. 5a. The performance of the slice selection schemes has been studied on the aforementioned slice specific load patterns. For this performed study we have considered the



(a) (b) (c)

Fig. 5: Number of slice instances in a) mMTC Slice, b) URLLC Slice, and c) eMBB Slice.



(a) (b) (c)

Fig. 6: Number of excess slice instances in a) mMTC slice, b) URLLC slice, and c) eMBB slice.

slice capacity in terms of number of users (in this case, 1000 users).

B. 5G System KPIs using Slice Monitoring in SERENS Framework

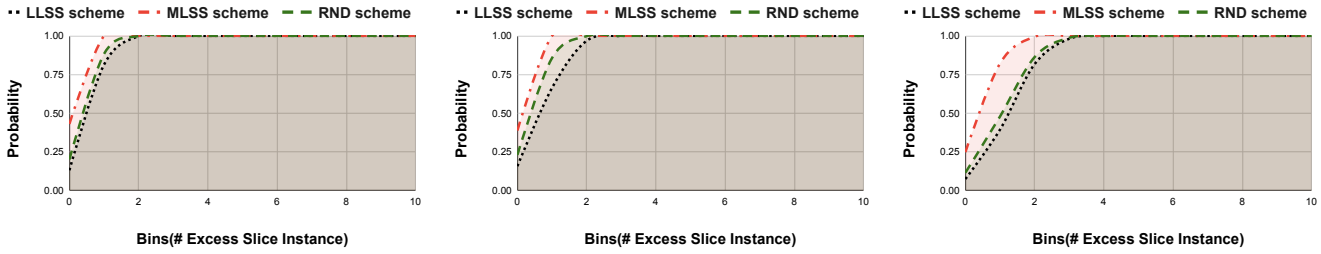
The slice monitoring mechanism of the proposed SERENS framework monitors the 5G systems KPIs with the deployed architecture depicted in the Fig. 3a and 3b. These figures show the CPU utilization and memory utilization captured for the active slice instances by the Slice Monitoring module using Prometheus [14] at regular intervals. Here, the slice1 represented by *5g_sba_e2e_slice-1* had higher number of users using it compared to other slices and hence is showing high CPU and memory consumption.

C. Performance of Slice Selection Algorithm in SERENS Framework

The slice analytics and selection functionalities of the SERENS framework is studied with the slice selection schemes running in the Slice Selection Algorithm module, while making use of the BDA based analytics performed at Slice Analytics Module of NSSF to achieve the dynamic slicing. The Slice Analytics module performs average value based analysis on the number of incoming USRs, from Slice Request Handler module, to predict the incoming USR pattern. The proposed Most Loaded Slice Selection (MLSS) scheme mentioned in the Algorithm 1 is studied and compared with Least Loaded Slice Selection (LLSS) scheme which maps the incoming user requests to the slice instances in the increasing order of the number of active users and Random Slice Selection (RND) scheme which picks a random candidate slice instance for serving the incoming USR. The schemes select a candidate slice in the resource optimised manner and hence we study their performance on three slice specific load patterns which are eMBB, URLLC, and mMTC, generated using the aforementioned simulation model (Section V-A). The schemes have been compared on the basis of metrics collected using the Algorithm 1. All three schemes make use of dynamic slicing to trigger the slice activation of the already provisioned slice instance(s) by making use of the predicted incoming USR pattern at Slice Analytics module.

Accordingly, the implemented slice selection schemes assume that the predicted USR pattern to be correct to achieve the dynamic slicing, while the errors in the prediction can result in running less number of slice instances than required and thus, the new USR might face a delay in getting the target network slice instance.

In Fig. 5, we record the number of active slice instances running to serve the active users. The plots of Fig. 5 shows the observed metric value for the slice specific load patterns. Due to higher TTL value for the incoming eMBB slice request, there are higher number of active users on the eMBB slice as compared to URLLC and mMTC slices at a given point of time. Presence of higher number of active users demands the requirement of running higher slice instances, in order to meet the requirements of the user traffic. We can observe



(a) (b) (c)

Fig. 7: CDFs of number of excessive slice instances in a) mMTC slice, b) URLLC slice, and c) eMBB slice.

that the proposed MLSS scheme has less number of active slice instances than the other two schemes (LLSS and RND) because MLSS scheme places the incoming USR in the resource optimised manner, allowing SPs to decommission an active slice instance (having no active user) and make better use of the available resources. On the other hand, LLSS scheme and RND scheme are observed to run more number of slice instances as they possess poor slice selection techniques from the active slice instances.

Fig. 6 shows the number of excessive slice instances running for each of the studied slice specific load pattern, as compared to the ground truth value, which represents the minimum slice instances needed to support the active users on the slice. We can observe that the proposed MLSS scheme is using excessive slice instances for less amount of time compared to other two schemes. LLSS scheme and RND scheme used an excessive slice instance for 45% and 30% of the time, respectively for mMTC and URLLC slices, and 60% and 50% of the time respectively for eMBB slices as compared to the proposed scheme. For the mMTC slices, we observe that the proposed MLSS scheme runs an excessive slice for just about 6% of the time as compared to the ground truth value, while the same value goes to 12% and 17% for URLLC and eMBB slices, respectively.

Fig. 7 shows the Cumulative Distribution Function (CDF) for excessive slice instances across different slices. We can observe that the proposed MLSS scheme shows high probability of having less excessive slice instances running across all the slice types as compared to LLSS and RND schemes.

VI. Conclusion and Future Work

In this paper, we have proposed a Self Regulating Network Slicing (SERENS) framework performing the Monitoring, Analytics and Selection of the slice instances in a closed loop automation to serve incoming user requests for efficient resource utilization. Our proposed SERENS framework helps the service provider to monitor the 5GC function's KPIs at the slice level and use it to select the candidate slices for the incoming user requests. Our study has shown that the proposed MLSS scheme outperforms the LLSS and RND schemes. This proposed MLSS scheme makes the best use of the available resources by handling the incoming slice traffic with minimum slice instances and thus, benefiting the service provider in terms of making higher revenue with the available resources. In future, we plan to introduce the Machine Learning (ML) based model(s) using a more realistic and noisy data set for the number of user requests to predict the number of slice instances required at a given point of time and evaluate the performance of the model(s) with our proposed MLSS scheme.

Acknowledgment

This work has been supported by the Department of Telecommunications, Ministry of Communications, India as part of the "Indigenous 5G Test Bed" project.

References

[1] "5G PAGODA End-to-end Network Slice", https://5g-pagoda.aalto.fi/assets/demo/attachement/delivrables/5G!Pagoda-D4.3-End-to-end%20Network%20slice_1.0.pdf.
[2] 3GPP, "System Architecture for the 5G System", Tech. Rep. TS 23.501, 3GPP, 2018.
[3] A. Kammoun, N. Tabbane, G. Diaz, and N. Achir, "Admission control algorithm for network slicing management in sdn-nfv environment", in *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, May 2018, pp. 1–6.
[4] T. V. K. Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, "Resource Allocation with Admission Control for GBR and Delay QoS in 5G Network Slices", in *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, Jan 2020, pp. 213–220.
[5] B. Han, A. DeDomenico, G. Dandachi, A. Drosou, D. Tzovaras, R. Querio, F. Moggio, O. Bulakci, and H. D. Schotten, "Admission and Congestion Control for 5G Network Slicing", in *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, Oct 2018, pp. 1–6.
[6] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "DeepSlice: A Deep Learning Approach towards an Efficient and Reliable Network Slicing in 5G Networks", in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, Oct 2019, pp. 0762–0767.
[7] M. R. Raza, A. Rostami, L. Wosinska, and P. Monti, "A Slice Admission Policy Based on Big Data Analytics for Multi-Tenant 5G Networks", *Journal of Lightwave Technology*, vol. 37, no. 7, pp. 1690–1697, April 2019.
[8] Shwetha Vittal, Mohit Kumar, and Antony Franklin A, "Adaptive Network Slicing with Multi-Site Deployment in 5G Core Networks (in press)", in *IEEE Conference on Network Softwarization (NETSOFT)*, 2020.
[9] "OSM", https://osm.etsi.org/wikipub/index.php/OSM_Release_FIVE.
[10] M. Peuster, H. Karl, and S. van Rossem, "MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments", in *2016 IEEE Conference on NFV-SDN*, Nov 2016, pp. 148–153.
[11] "Openstack", https://www.openstack.org.
[12] "Docker", https://www.docker.com.
[13] "Nghttp2: HTTP/2 C Library", https://nghttp2.org/.
[14] "Prometheus", http://prometheus.io.