IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 10, 2021, accepted May 23, 2021, date of publication May 28, 2021, date of current version June 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3084834

Network Slice Lifecycle Management for 5G Mobile Networks: An Intent-Based Networking Approach

KHIZAR ABBAS, TALHA AHMED KHAN, MUHAMMAD AFAQ, AND WANG-CHEOL SONG
Department of Computer Engineering, Jeju National University, Jeju-si 63243, South Korea

Corresponding author: Wang-Cheol Song (philo@jejunu.ac.kr)

Khizar Abbas and Talha Ahmed Khan contributed equally to this work.

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) Support Program Supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-2017-0-01633, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) through the Ministry of Education under Grant NRF-2016R1D1A1B01016322.

ABSTRACT Network slicing in 5G is a solution to accommodate a wide range of services. It also enables the network operators to establish multiple end-to-end (e2e) logically isolated and customized networks with shared or dedicated resources over the same infrastructure. Although, many tools and platforms have been developed to accomplish the management and orchestration (MANO) of e2e network slicing automatically, it is still challenging. Each of these platforms requires expertise and manual effort to define the requirements for the provisioning of the resources. The other issue is the generation of well-defined network slice configurations with lifecycle parameters. To this end, this paper proposes an efficient solution that automates the configuration process and performs the management and orchestration of network slices. This solution contains a one-touch Intent-based Networking (IBN) platform that effectively orchestrates and manages the lifecycle of multi-domain slice resources. IBN automates the process of slice configuration generation, service provisioning, service update, and service assurance by eliminating experts and manual effort. Furthermore, it has an intelligent Deep Learning (DL) based resource update and assurance mechanism which handles the run-time resource scalability and assurance.

INDEX TERMS SDN, NFV, e2e network slicing, IBN, 5G networks, E2e orchestration, slice LCM.

I. INTRODUCTION

The future 5G network will support a vast range of innovative services for customers, enterprises, various industrial verticals, government level demands, and virtual mobile network operators (MVNOs). According to their different quality of service (QoS) requirements, these services are categorized as massive machine-type communications (mMTC), enhanced mobile broadband (eMBB), and ultra-reliable low latency (URLLC) [1], [2]. Therefore, 5G network fulfills these service requirements through a core technology called network slicing. It is a central pillar of the 5G network, managed with end-to-end (e2e) service orchestration. Network slicing is not a new concept. Previously, it has also been supported in 4G networks. Still, its isolation level among multiple services is limited over the common physical infrastructure, such as multi-operator support for core network, multi-domain support, and dedicated core network resources. On the other side, network slicing in 5G will allow the network operators to create isolated data pipelines for each service type and guarantee the QoS requirements. Network slicing will also ensure data transmission quality for mission-critical, autonomous driving cars, time-sensitive, and low latency services [3].

Network slicing creates multiple logical isolated virtual networks over the shared common physical infrastructure that fulfill QoS requirements requested by the different tenants, consumers, or network service providers. The network function virtualization (NFV), cloud technologies, and software-defined networking (SDN) enable future networks to create flexible, reliable, and programmable virtual network instances as per QoS requirements. NFV allows the small vendors or organizations to develop virtual network functions (VNFs) to run over general-purpose or Whitebox hardware devices. The core concept of network slicing is intimately relevant to the cloud computing infrastructure as a service (IAAS). IAAS enables the cloud providers to share

The associate editor coordinating the review of this manuscript and approving it for publication was Amin Zehtabian.

the network, storage, and computing resources between different tenants and creates fully-featured virtual networks in collaboration with NFV and SDN technologies. However, SDN decouples the control plane from the data plane and enables the network programmability. Each virtual network created in a network slice, provides complete e2e functionalities through radio access network (RAN) and core network domains. Due to these aspects, the network slice is a combination of various VNFs that provides scalable, programmable, and flexible network services. It reduces the operational expenditures (OPEX) and capital expenditures (CAPEX) significantly by managing and orchestrating the VNFs efficiently [4], [5].

There is a need for a well-designed tool or platform to automatically manage the lifecycle of virtual and physical network slice resources. Network slice lifecycle management (LCM) is considered as a fundamental solution to managing slice instantiation, activation, runtime monitoring, updating, and deactivation [6], [7]. The process of LCM follows the following step. It starts with the slice instantiation phase, which defines the slice requirements into policy catalogs/slice templates. Once the slice policy is defined, the slice creation phase starts to reserve the resources and requests the orchestrator to create the slice. After that, the slice instantiation process instantiates the slice components, and the activation process activates the slice and makes it available for use. At the activation phase, the network slice is working to keep the slice or service in a demanded state the monitoring system plays its role. The monitoring phase provides feedback to update the slice as per the services demand. Once a slice has reached its life span, it is required to be deactivated in the deactivation phase. Hence LCM performs complete management and orchestration of network slice instances.

Network slicing faces several challenges regarding e2e management, isolation, elasticity, and customization while allocating the resources. Especially, sharing and isolation of slice resources are not so simple because of fluctuating communication network environment—for example, isolation and customization of the RAN resources. The customization of network slice resources enables the network operators (NOs) to maximize the utilization of slicing resources to accommodate the strict services requirements. The critical point is the conversion of specified service requirements into desirable network resources. This conversion requires deliberation at various levels such as data plane level, control plane level, and network-wide level. Due to the variable nature of the network, the deployment and fulfillment of Service Level Agreement (SLA) is challenging [8]–[10]. For example, it is a demanding task to deploy the total number of VMs with required storage, compute, and network resources in real-time.

A. PROBLEM STATEMENT

The 5G network supports a wide range of services that have distinct requirements. Therefore, it is a time-consuming and error-prone process to generate configurations for each service manually. It also required a lot of manual effort and expertise. Moreover, the manual resource allocation for network slicing is not an optimal approach. An automated and optimal allocation of resources according to service demand is mandatory. The automatic allocation of e2e resources in a multi-domain environment is a challenging task. Moreover, the existing systems cannot accommodate the drastic changes in network configurations because of intricate traffic patterns over the network [2], [11], [12]. Hence, a well-designed platform is required that can automate the network configurations generation process dynamically by considering diverse service requirements. This manuscript proposes an automated solution that enables the NOs to generate network configurations for a multi-domain environment and control and update the network state based on run-time requirements. More importantly, it supports network slice LCM by providing an automated way to instantiate, update, and delete the network slices. It follows a closed-loop mechanism which ensures the system in a stable state while fulfilling the user demands.

B. RESEARCH SOLUTION

Intent-based networking (IBN) is a promising solution developed based on Internet Engineering Task Force (IETF) standards [13]. It can manage and control the network configurations, modifications, and operation processes in an automated fashion. In the IBN approach, a network should have several features such as self-organizing, self-assurance, self-healing, and self-configuration. We have proposed a novel Intent-based networking platform for e2e network slicing that automatically orchestrates network resources in a multi-domain environment. It additionally has a deep learning module that can proactively update the system on runtime based on varying service requirements. Our Intent-based e2e network slicing system consists of an IBN platform, NFV orchestrator OSM, RAN controller FlexRAN, and deep learning module. IBN is an orchestration platform that automates the network configuration generation process for underlying multi-domain infrastructure. It takes the higher-level QoS requirements from the operators for a slice and translates them into specific slice template/policy configurations acceptable by the underlying orchestrators. Additionally, the orchestration platform keeps the network in the desired state by proper monitoring and updating mechanism. This system performs step-by-step network slice LCM while creating, monitoring, updating, and deleting the core and RAN domain's network slices.

C. MAJOR CONTRIBUTION

The main contributions of our proposed network slice LCM solution are summarized as follows:

- A closed-loop intent-based networking platform is designed and developed to automate the slice configuration and resource orchestration for multi-domain.

- A one-touch system that eliminates the manual effort for creating, deleting, and updating the network slice instances. The users provide QoS requirements from the IBN platform Graphical User Interface (GUI). After requirement submission, it translates these requirements into slice template/policy configurations to deploy resources on the underlying infrastructure.
- It follows LCM-IETF standards [7] while managing the lifecycle of network slice instances automatically.
- Continuous monitoring of slice resources is essential for lifecycle management; IBN integrates open-source monitoring tools for core and RAN resources.
- The IBN standards [13], [14] include intelligence for multi-domains of the networks. The intelligent intent management and update mechanism is achieved through the deep learning module. It has the capability to scale up/down the slice instance resources when the requirement changes.

The rest of the manuscript is organized as follows. Background literature related to the standardization and specifications of IBN networking, e2e network slicing, open-source solutions, open-source NFV orchestrators has been presented in Section II. Section III explains the 3GPP management and orchestration technologies. Besides, Section IV contains the design and architecture of the proposed system. Section V describes the experimental test-bed setup. Section VI explains the experimental results generated with the proposed approach. The conclusion and future direction of the presented work have been discussed in the final Section.

II. RELATED WORK

A. NETWORK SLICING PROGRESS AND STANDARDS

The e2e network slicing is a promising solution to accommodate a wide range of services supported by 5G networks. Many standardized bodies of communication networks have developed the specifications for e2e network slicing. Third Generation Partnership Project (3GPP) is one of the leading organizations, other research organizations such as ETSI, Fifth Generation Partnership Project (5GPPP), Next Generation Mobile Network (NGMN), International Telecommunication Union (ITU), and RAN3 have also proposed their network slicing standards. Some popular industrial organizations like Cisco, Samsung, Huawei, HP, Nokia, and Ericsson are also contributors to design 5G specifications. They divide the network slicing into three broad categories: xMBB, mMTC, and URLLC. The xMBB type of services requires high bandwidth, reliable broadband communication, and low latency in highly dense areas. In addition, the mMTC needs seamless wireless connectivity for the immense number of devices. On the other side, the URLLC type services require an ultralow latency connection of 10ms. Each of these services needs isolated resources to satisfy the requirements [6], [11], [13], [15]–[19].

In the service-oriented 5G network, network service orchestration and management are crucial and challenging tasks [15]. European Telecommunication Standard Institute (ETSI) has proposed an NFV management and orchestration (NFV-MANO) framework that can manage and orchestrate the virtual network instances (VNFs) efficiently. It comprises NFVO, virtual infrastructure managers (VIMs), and VNF managers (VNFMs). The first and most crucial component is NFVO. It is responsible for orchestrating the network services with the cooperation of operation and business sup-port system (OSS/BSS). In the MANO framework, the users provide the network configurations to the NFV orchestrator to orchestrate the network resources through VIMs. These configurations are vital because it defines network behavior. Therefore, Open-Source MANO (OSM) is the finest example of ETSI NFV-MANO standards implementation that is the most popular platform among the development community. Figure 1 illustrates the components of NFV-MANO framework. Multiple orchestration platforms follow either ETSI or 3GPP standard architecture for orchestration of next generation network services using high-level configurations. Each orchestrator accepts predefined standard configurations from the top and applies them to the different layers of infrastructures. Based on platform configuration and orchestrator supports different VIM and SDN controller can be integrated and controlled using the predefined top-level configurations [20], [21].

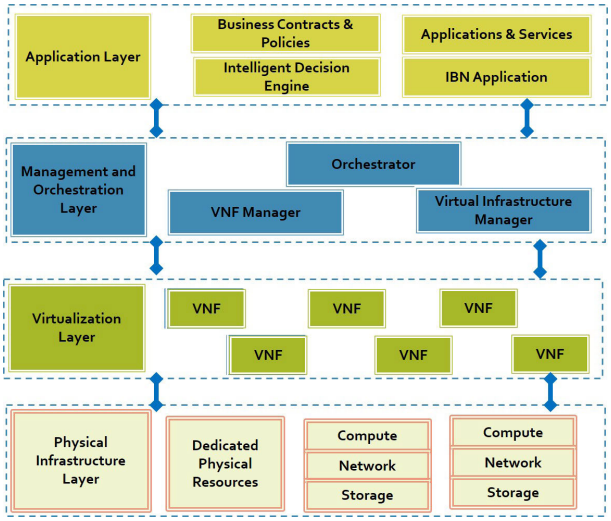


FIGURE 1. Management and network orchestration (MANO) architecture.

Many open-source solutions enable the NOs to perform e2e network slicing developed by different organizations, including ONAP, OSM, OpenBaton, JOX, Cloudify, Tacker, 5G NORMA, M-CORD, 5G TANGO, SONATA, 5G-PAGODA, OpenNFV, T-NOVA, OPNFV [4], [18], [20], [22]–[29].

Table 1 presents the comparative analysis of the developed system with existing approaches in the literature. The proposed system has advantages over all the considered approaches because it ensures QoS, an e2e slice LCM support, a DL-based intelligent resource update mechanism, and OSM and FlexRAN controllers for controlling core and RAN

TABLE 1. Detail comparison of the proposed system with existing approaches.

Ref#	Objective	Slice LCM support	AI-based intelligence	QoS assurance	Orchestration platforms
[9]	This work presents a plug-in-based mechanism where the vendors request network resources, and the system coordinates with orchestration entities to ensure resource availability.	✗	✗	✗	OSM
[10]	This work illustrates the e2e network slicing system in the core, RAN, and transport domains. It proposed an orchestrator which controls each domain controller and handles the core and RAN mapping to enable e2e connectivity.	✓	✗	✗	Orchestrator
[11]	The presented work is a pioneer and practical mechanism for slicing the LTE network. This work is based on the OAI LTE protocol stack, JUJU charms, and JUJU VNFM.	✗	✓	✗	JUJU framework
[12]	It proposed a deep Q-learning (DQL) based radio resource allocation mechanism in cloud-RAN.	✗	DQL	✗	✗
[18]	Presents a novel framework for handling network slice orchestration and management for a multi-domain environment. This work enhances the 3GPP MANO architecture, consisting of network slice manager, descriptor design function, multi-domain deployment executor, and network slice lifecycle management function modules.	✓	✗	✗	✗
[27]	This work presents an AI-based network slicing mechanism for future generation cellular networks. It includes an efficient RAN slicing approach with AI techniques. It also provides a detailed survey of how AI and reinforcement learning (RL) techniques are helpful in wireless network domains.	✗	DNN and RL	✓	✗
[30]	Illustrates an orchestration platform based on an open-source solution named as Open Network Automation Platform (ONAP) for slicing the core, RAN, and transport domain.	✓	✗	✗	ONAP
[31]	This work presents an automated mechanism for service-oriented LCM in 5G cellular networks. It manages the lifecycle of supported services using the knowledge management technique.	✓	✗	✓	Cloudify, OpenStack Heat
Proposed work	The proposed work is a closed-loop orchestration mechanism based on IBN-enabled e2e slice lifecycle management where the user inputs QoS requirements, and the system, in return, deploys the e2e network slices.	✓	LSTM-based resource forecasts and update mechanism	✓	OSM, M-CORD, FlexRAN

network domains, respectively. Moreover, it has an IBN platform for handling the underlying OSM and FlexRAN.

B. INTENT-BASED NETWORKING STANDARDS

Intent-based networking (IBN) is an innovation toward the management and orchestration of the networks. The IBN system's primary goal is to develop a closed-loop platform where users need to input higher-level service requirements. Afterwards, the system translates them into low-level configurations and automatically performs the resource orchestration. It automates the process to create, deploy, configure and update the network resources according to the NOs intentions. This system can manage the lifecycle of the network resources by proper monitoring and control. IBN provides intelligence to the networks by using machine learning approaches. This system enables the future communication networks to be self-organized, self-planned, self-healing, and self-assured [32]–[34].

With the collaboration of Cisco, Huawei, and APSTRA, IETF has developed the IBN standards to automate and manage future generation networks [14], [35], [36]. Cisco developed its first closed-loop IBN system in 2017. It has three main modules such as intent-translation, intent-activation, and intent- assurance shown in Figure 2. Cisco IBN is a one-touch system where the user provides their intention, and the system performs all operations automatically to orchestrate the resources over the infrastructure. The IBN intent translation module enables the network operators (NOs) to express their service requirements/intentions in a declarative fashion. The NOs should deliberate business perspectives and not worry about the resources' configuration to achieve the desired target. Besides, the intent activation module is responsible for deploying the received intents over the underlying network resources. It activates the resources with domain NFVOs. The last intent assurance module continuously monitors the activated resources and performs auto-recovery in case of failure [14]. It also has machine learning models to assure and update the resources on demand.

Huawei has developed an intent-driven Networking (IDN) platform, which is very similar to Cisco IBN [35]. It supports closed-loop, multi-services, failure and recovery, e2e slicing, intelligent management and operation, and lifecycle management functionalities for future generation networks. It is an efficient system to orchestrate the business intent and ensure a better quality of experience (QoE). It has an intelligent control and management module named Network Cloud Engine (NCE). NCE is responsible for performing the intent translation and activation functionalities. It contains

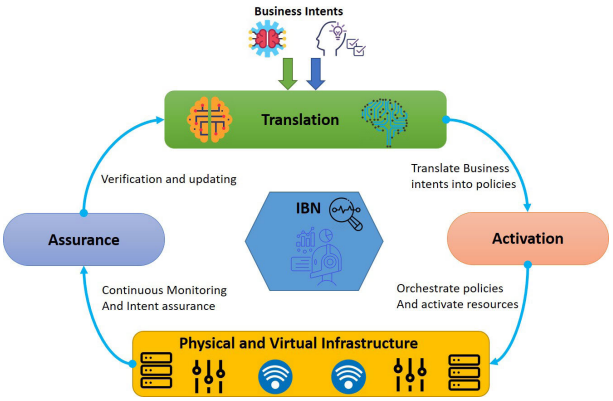


FIGURE 2. Design and architecture of IBN system.

big data and Artificial Intelligence (AI) modules to support network intelligence. Moreover, the APSTRA operating system (AOS) is another solution to manage and control the cloud infrastructure and enables edge cloud connectivity [36]. Since all the existing approaches cannot perceive the integration among different domain orchestrators, they require complex configurations to control and manage the network slice resources. Hence, the primary goal of this paper is to develop a unique one-touch IBN platform based on the IETF standard approach for automatic orchestration and control of network infrastructure regardless of underlying platforms. To this end, an IBN-based end-to-end slice orchestration and lifecycle management framework is proposed in this manuscript.

C. CYBERSECURITY CHALLENGES IN NETWORK SLICING

One essential feature of network slicing is the allocation of dedicated and isolated resources for multiple network operators that alleviates the security attacks in 5G networks. For example, the authors proposed an attack mitigation mechanism for network slicing. The proposed system mitigates the Distributed Denial of Services (DDoS) attacks in the core network functions by the slice isolation feature [37]. Thereby, multiple security-related network functions can be deployed within the slices to achieve different security levels as per NOs demand. Hence, the attacks executed against one slice do not affect the other slice resources due to slice isolation. With the slice isolation, the slice-specific security NFs will also be required to prevent these attacks. Further, MANO in network slicing can also provide the support to implement the security NFs dynamically. The third-party security organizations are also accepted to join and implement the network slicing security mechanism to secure 5G networks. It can also reduce the management tasks of the NOs and spread the duties to other tenants.

On the other side, a secure network slicing system should guarantee authentication, availability, authorization, confidentiality, and integrity security principles [38]. In the proposed system, only slice owner, NOs, infrastructure providers, and administrators interact with the network slicing system using their credentials through the GUI portal of the IBN system. Besides, the IBN system is available for the customers to deploy the slices and ensure QoS. For the authorization case, the slice owners only control and manage their own slice's resource or NFs. Our mechanism provides the dedicated and isolated core or RAN resources for each slice to ensure confidentiality. So, the packets of each slice use slice own dedicated core and RAN NFs. Besides, to ensure integrity, the data of each slice is secure and not tampered. Only the slice owner has permission to change the application data and flow path.

III. 3GPP MANAGEMENT AND ORCHESTRATION OF NETWORK SLICE

3GPP has designed a crucial architecture for management and orchestration (MO) of the 5G networks. This MO architecture contains the following network management functions: communication service management function (CSMF), network slice management function (NSMF), and network slice subnet management function (NSSMF). Each subnet slice domain has its own NSSMF function, such as RAN, core and Transport domains. The network slice requirements definition and its deployment with lifecycle parameters are fundamental issues in the network slicing domain.

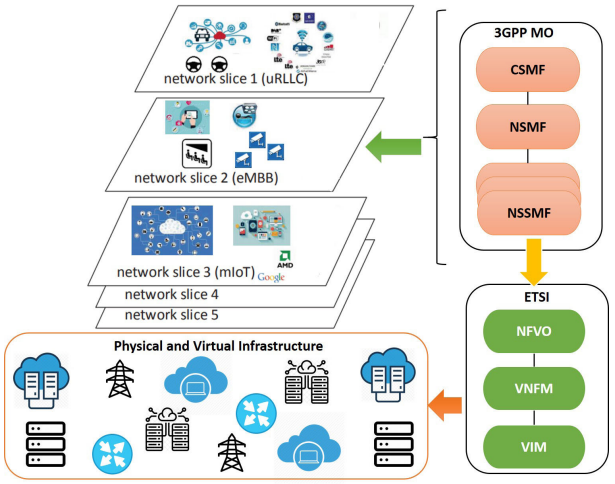


FIGURE 3. 3GPP and ETSI integration to manage e2e network slicing.

Specifically, the MO architecture is designed to eliminate this issue. The user inserts the service requirements, and CSMF converts them into network slice requirements. These slice requirements are forwarded to NSMF for further processing. Then, NSMF decomposes the slice requirements into different subnet domain formats. After that, those requirements are given to NSSMF for the deployment of the resources. NSSMFs of various subnet domains manage and orchestrate the slice resources over the corresponding infrastructure domains [5]–[7], [39]. For instance, whenever a slice request is received from NSMF, the NSSMF of the core network domain estimates the total number of VNFs (VMs) resources with storage, computing, and network capabilities. The required VNFs are deployed with the help of NFVO of the MANO platform. The other responsibility of NSSMF

is to adjust scale up/down the resources dynamically with the coordination of MANO NFVO. Figure 3 presents the integration of 3GPP MO and ETSI components to achieve e2e network slice lifecycle management.

A. 3GPP LIFECYCLE MANAGEMENT OF NETWORK SLICE

Many research studies have shown many results related to e2e network slicing in the literature, but managing the network slice lifecycle is still challenging [16]. Figure 4 shows the slice LCM phases and their working. According to 3GPP standard [6] of the network slicing LCM (lifecycle management) consists of four major modules: commissioning, activation, runtime monitoring and decommissioning.

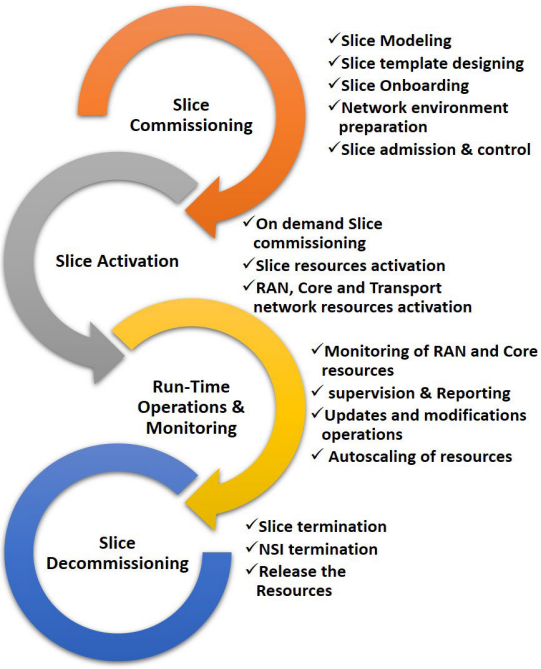


FIGURE 4. Network slice lifecycle management.

In the slice commissioning phase, the preparation and design for a slice is handled, e.g., a slice template is created in the first phase. This slice template contains information about the user requirements and resources to be deployed. After that, the designed slice template is deployed on the underlying infrastructure with NFVO and network controllers; the specified resource should be assigned to a slice and activated. The activated slice resources are monitored continuously through the monitoring tools, and in case of any failure, it will be notified to the NFVO orchestrator to resolve the error. In the decommissioning phase, the created slice would be deleted automatically at the specified time [9], [30], [31].

B. IBN-BASED NETWORK SLICE LIFECYCLE MANAGEMENT

Our IBN system [40]–[43] manages and orchestrates the LCM of e2e network slices automatically. It works quite similarly to the 3GPP MO platform discussed above. It can also scale up/ down the resources on-demand with the help of domain NFVOs and controllers. IBN is a one-touch system where the user inputs service requirements and the system automatically converts these requirements into slice template and forwards to domain NFVO and Controllers to activate the resources. Additionally, NFVO OSM and FlexRAN controller are responsible for controlling and managing the core and RAN subnet domains. Moreover, IBN automatically performs the slice commissioning, activation, decommissioning, and run-time monitoring of RAN and core network resources.

IV. RESEARCH METHODOLOGY

A. OVERVIEW OF PROPOSED SLICE LIFECYCLE MANAGEMENT THROUGH IBN PLATFORM

The detailed architecture of the proposed IBN-based e2e slice LCM is depicted in Figure 5. It is divided into four major categories: commissioning, activation, runtime monitoring, and decommissioning of network slices. This system has an IBN platform that is responsible for handling the e2e network slice lifecycle management. The IBN system consists of a web-based GUI, IBN manager, knowledge base or policy store, Policy configurators/slice template generators, and intelligent update engine. The IBN GUI is used to design the slice where users input slice requirements in the form of user intents. The IBN manager acts as a mediator for all the components of the IBN. The policy store is a database containing VNFs, instance images, versions, IP addresses, and infrastructure information. It also stores the information related to the deployed intents. Besides, the IBN manager prepares a VNF forwarding graph (VNFFG) by mapping the intent requirements into the required resources. It sends the created forwarding graphs to the slice template generator module, which has policy configurators for translating higher-level QoS requirements into policy configurations. Moreover, policy configurators forward the slice templates to OSM and FlexRAN controller to deploy the core and RAN resources. In addition, the IBN system has an intelligent update and assurance engine that automatically updates the resources at runtime in failure or overload cases.

B. RESEARCH PHASES OF SLICE LCM MECHANISM

The step-by-step explanation of the IBN-based slice LCM mechanism is presented below:

1) SLICE COMMISSIONING

As aforementioned, we have an IBN platform for the slice design and preparation that is an essential and unique component. The IBN system has a web-based portal GUI, intent manager, knowledge base or catalog repository, slice template generators, and intelligent update engine. The web portal provides the users (customers, mobile network operators, resource vendors) an easy way to input slice QoS requirements in the form of user intents. The user intents are higher-level abstract requirements (SLAs) for a slice,

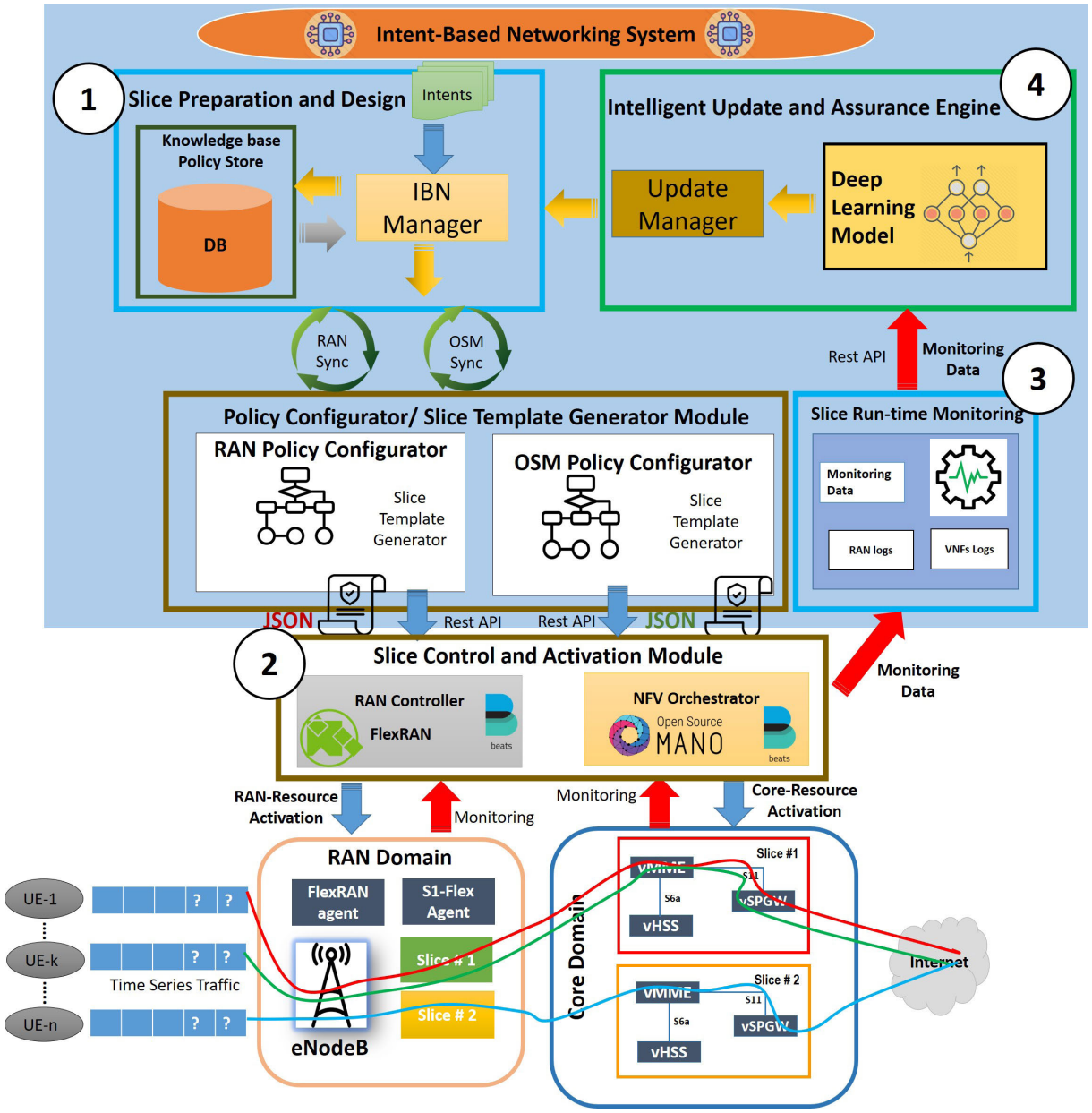


FIGURE 5. Detailed architecture of IBN-based e2e slice orchestration and lifecycle management.

e.g., slice information (sliceID/S-NSSAI), QoS requirements (up-rate, down-rate), service type (eMBB, URLLC, IoT), slice instantiation time, and deletion time-period. The intent manager is the communication entity between all the modules. It is a coordinator that is responsible for forwarding messages among all components of the IBN directly. It also interacts with the network orchestrators through the REST API. Whenever a user intent comes from the users, it fetches the resource and architecture information from the knowledge base database, maps them with the available resources, and creates a forwarding graph. The knowledge base or catalog repository is a rich database management system that can store all the information regarding the underlying resources, VNFs, PNFs, IP addresses scheme, instance images, and deployed intents. The generated forwarding graph is forwarded to the slice template generators/policy configurators module for further processing. We have multiple slice template generators for multi orchestrators, such as NFVO OSM, SDN RAN controller FlexRAN and ONOS controller. These slice template generators receive the user intent forwarding graph from the intent manager and convert them into the underlying NFVO acceptable format. The OSM slice template generator generates the slice template into JSON string format because OSM NFVO accepts the template in this form. The Slice template generator for RAN generates the JSON file format template because the FlexRAN controller agrees with the slice template in this format. The M-CORD policy configurators convert the user requirements in TOSCA

format. In this slice instantiation module, we designed and prepared the slice template according to the underlying NFVOs.

2) SLICE ACTIVATION

We have multiple NFVOs and network controllers for the network slice activation in a multi-domain environment, such as OSM MANO, M-CORD, ONOS controller, and FlexRAN controller. How a slice would be activated through these orchestrators are explained below.

- Network Orchestrator Open-Source MANO (OSM): Open-source MANO is a network orchestrator developed by ETSI that can enable mobile network operators to deploy e2e network services in an automated fashion [20]. It is designed on NFV specifications that consist of multiple modules such as NFVO(NFV orchestrator), VIM (virtual infrastructure manager), and VNFM (virtual network function manager). It has a GUI portal for managing, monitoring, and deploying the network resources automatically. OSM facilitates the user to deploy the e2e network slices and manages the slice's lifecycle. Our e2e network slicing test-bed has used NFVO OSM to deploy Core network VNFs such as EPC VNFs (vMME, vHSS, vSPGW). The OSM slice template generator/ OSM policy configurator of the IBN system generates the slice configurations for OSM NFVO in the JSON string format. The complete design of our policy configurators are well explained in our papers [41], [42], [44]. Moreover, the slice template is forwarded to OSM through the REST API, and then the OSM deploys the core network VNFs with the help of VIM (OpenStack) over the infrastructure automatically. In this way, the core network VNFs are automatically deployed using the OSM. The core VNFs are also deployed in our test-bed with the M-CORD platform by providing the slice template in TOSCA format.
- RAN domain controller: We have used the FlexRAN controller for slicing the RAN resources as per user requirements. FlexRAN is an SDN-enabled RAN controller that provides a highly programmable environment and decouples the control plane from the data plane. Its programmability support enables the users to develop the applications on top of it. It can also control and manage multiple base stations efficiently. It has two main components: FlexRAN control plane and agent API. The FlexRAN master controller resides inside the FlexRAN control plane, which interacts and controls the FlexRAN agents [45]. On the other side, the FlexRAN agent acts as a local controller when interacting with the other agents and the master controller. It can control one eNodeB at a time. The FlexRAN agent API is a southbound interface that can decouple the control plane of FlexRAN from the eNodeB data plane. The FlexRAN protocols are used for the communications between a FlexRAN master-controller and agents. Control and monitoring applications are developed on top of the RAN controller and communicate with the master controller through the northbound interface. These applications are used for the automatic management, modifications, and control of the RAN resources. FlexRAN controller in our network slicing system is responsible for slicing the RAN resources. Due to this, our RAN slice template generator converts the higher level slice configurations into the JSON format. Moreover, it receives the RAN slice template generated by the RAN slice template generator and deploys the slice over the eNB as per requirements specified in the slice template, e.g., 20 MB/s bandwidth. We made some modifications by developing management and control applications on top of controller to deploy QoS aware network slices. We have developed a radio resource management application for accommodating QoS aware RAN slicing. Besides, it can also monitor and control the RAN infrastructure.

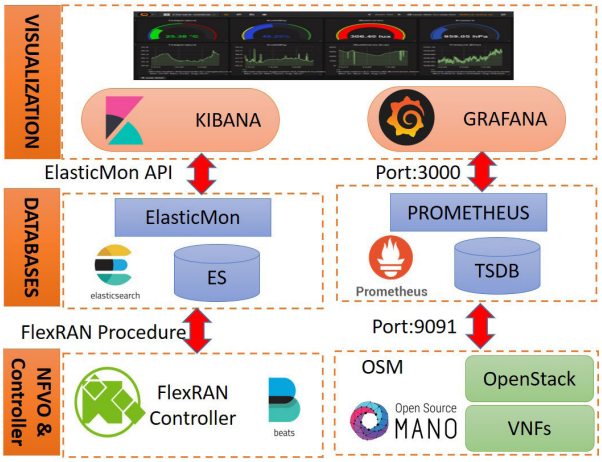


FIGURE 6. Slice resources run-time monitioring mechanism.

3) RUN-TIME MONITORING OF SLICE RESOURCES

The run-time monitoring of slice resources is essential for efficient resource management. Figure 6 illustrates the monitoring module that works in two ways: RAN domain monitoring and Core VNFs resources monitoring. The eNodeB resources are monitored using the FlexRAN controller and elasticMon tool [46], [47]. The elasticMon is a monitoring tool specially designed for 5G mobile networks to monitor the enormous real-time data traffic to control and manage it. FlexRAN controller and elasticMon tool can monitor and store the statistics of the eNodeB. Kibana and Grafana tools are used on the top of the FlexRAN controller and elasticMon to visualize the data traffic on runtime. On the other side, core network VNFs are monitored by the Openstack telemetry service or ceilometer. On top of that, MON OSM monitoring service collects the VNFs data logs that are further forwarded to Prometheus, which is an open-source tool for monitoring the real-time data traffic and stored these matrices into a time-series database. So, Prometheus kept the VNFs data logs into the real-time database repository. After that Grafana tool is integrated with the Prometheus database to visualize the

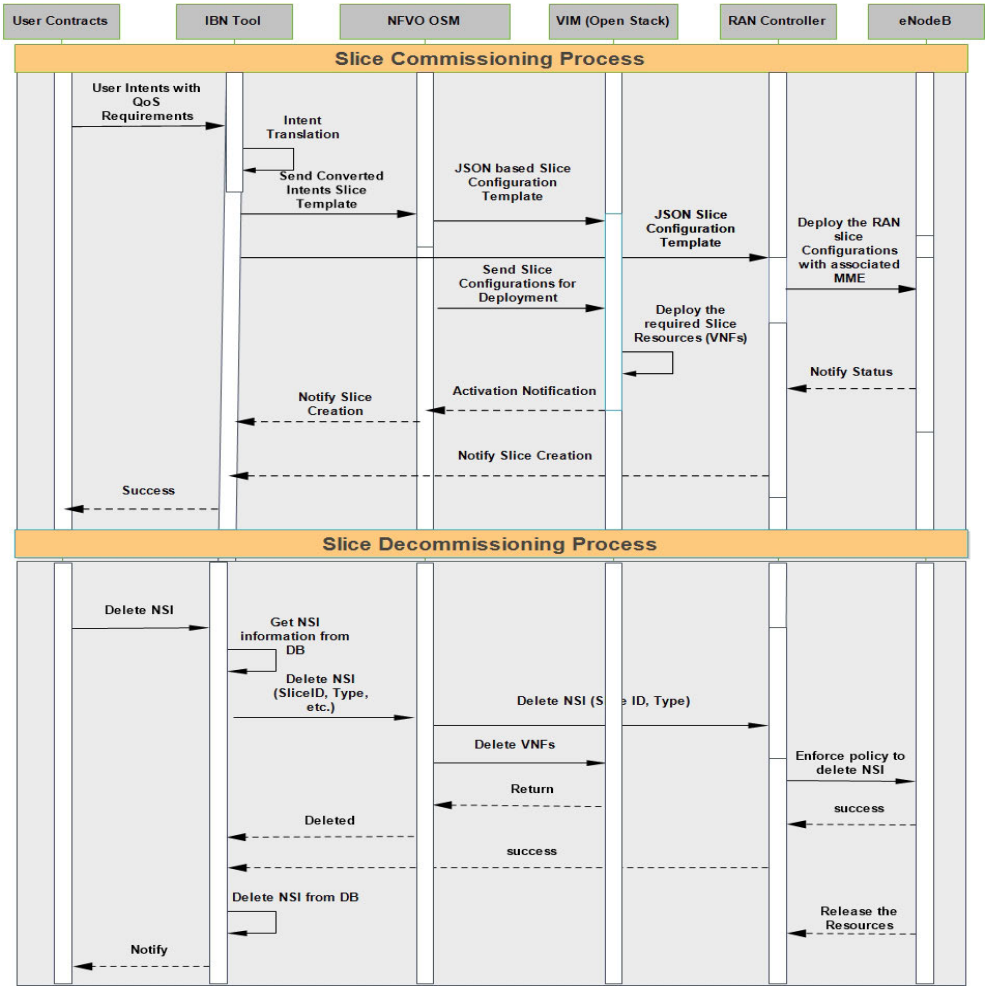


FIGURE 7. Slice instantiation and deletion workflow.

VNFs resource stats such as CPU usage, RAM usage, etc. Using these open-source monitoring tools, we can monitor and collect the network slice resource usage on runtime. In case of failure, our system can reconfigure and update the slice resources dynamically.

4) SLICE DECOMMISSIONING

There are two folds for slice deletion using our IBN network slicing system. The first way is that the slice termination re-quest is generated through the IBN system web portal, which contains the slice SNSSAI (single network slice selection assistance information), SliceID. The slice deletion configurations are rendered the same as the slice creation process. The slice deletion template is forwarded to the NFVOs such as OSM and FlexRAN controller to delete the resources. The OSM and RAN controller delete the specified resources and release the resources for future usage. The second way is that slice starting and ending time should be inserted during the slice creation phase. When the slice ending time comes, the IBN system automatically generates the slice deletion request and automatically performs the same process. After that, deletion confirmation notification is sent back to the operator through the IBN platform.

Figure 7 illustrates the step-by-step slice activation and deletion process through the IBN system. In the first step, the users define slice requirements in the form of intent by inputting QoS (uplink, downlink) requirements, sliceID, slice category information through the GUI of the IBN system. After that, the IBN policy configurators translates these higher-level requirements into policy configurations for each domain and forwards them to OSM and FlexRAN to deploy resources. Furthermore, OSM prepares the core network EPC VNFs resources for the requested slice using OpenStack. On the other hand, the FlexRAN controller deploys the slice configurations at the RAN domain. These slice configurations include information about dedicated vMME, slice SNSSAI, and QoS requirements used to stitch RAN slice with core VNFs. Afterward, IBN notifies the user of successful slice activation. For the slice decommissioning process, the user needs to input the network sliceID information through the IBN portal. Then IBN manager checks the slice information from the catalog repository of the IBN system. IBN again translates the slice information into policy

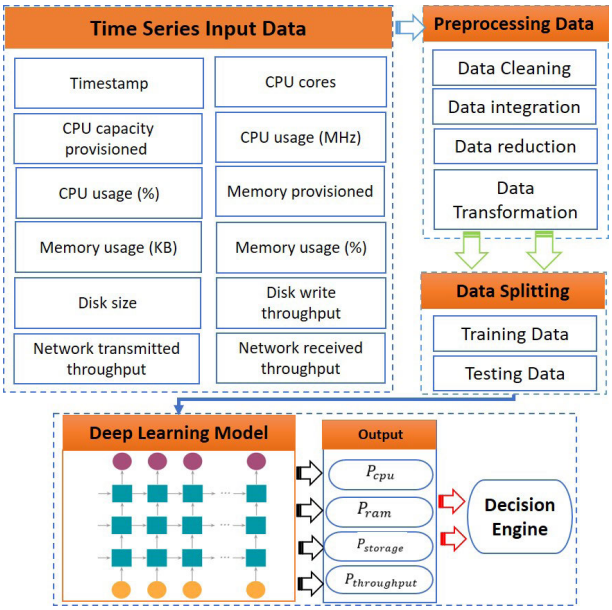


FIGURE 8. Illustration of proposed intelligent update and assurance engine with DL model.

configurations and forwards them to OSM and FlexRAN controller to delete core and RAN domain resources associated with that slice. Subsequently, OSM with OpenStack deletes the core EPC VNFs, and FlexRAN deletes the slice at eNodeB. Finally, OSM and FlexRAN notify the IBN platform about slice deletion. IBN reports the user for successful slice deletion. In this way, the IBN system can automatically design, activate, and delete the slices where the user only provides higher-level slice requirements, eliminating manual effort.

C. INTELLIGENT UPDATE AND ASSURANCE ENGINE

The most crucial aspect of the IBN-based closed-loop system is the self-assurance and update mechanism. To this end, the IBN system performs monitoring and utilizes an intelligence-driven approach to update the system autonomously. Hence, this section explains the service assurance, update, failure detection, and auto-recovery mechanism followed in this test-bed. The goal of network resource monitoring is to assure smooth service provisioning according to the system state. We have used a proactive DL-based model that continuously receives the resource status and updates the system according to predicted future needs.

The proposed intelligent update and assurance engine's working is illustrated in Figure 8, which consists of the following phases: real-time cloud resource time-series dataset, data processing, data splitting, DL model, and decision engine. We have used the open-source dataset and performed preprocessing operations such as data cleaning, feature extraction, and feature reduction techniques to transform the data for the DL model. After that, the preprocessed dataset is divided into training and testing the DL model. As mentioned above, due to time series data, we have used the Recurrent

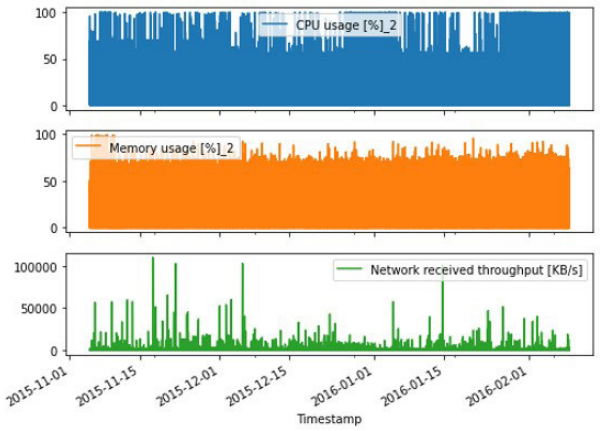


FIGURE 9. A sample of preprocessed dataset.

Neural Network (RNN) long-short time memory (LSTM) model to predict future resource utilization in the cloud environment. Further, the predicted resource utilization (CPU, RAM, storage, throughput) from the LSTM model is used by the decision engine to decide either the update required or not. The step-by-step working of each component is presented below.

1) DATASET

The real-time dataset is required for applying the DL model. However, we have used one open-source cloud data center dataset collected by the GWA-T-13 MATERNA, a German-based cloud provider company. This dataset contains the performance metrics of 527, 547, and 529 VMs. The total resources used for creating these VMs are 49 hosts with 69 CPU cores and 6780 GB RAM. This dataset was collected in three different traces within three months, each trace per month. MATERNA dataset contains 12 resource utilization attributes: CPU utilization, memory utilization, throughput, and disk size, etc. More details related to the dataset is provided here [48]. Besides, we have performed some preprocessing operations on the datase and extracted the relevant feature required for DL model training.

The MATERNA dataset's visualization is presented in Figure 9, which shows the memory and CPU usage in percentage, and KB/s for two months time-period. Visualization illustrates the depth of collected data with the peak utilization of CPU in percentage up to 100 percent similarly, the utilization of RAM also reaches peaks. The DL model learns the important patterns from the dataset and predicts the future resource utilization used for scaling the cloud resources. Figure 10 presents the heat map plot that shows correlation among different features of the prepossessed data set. The correlation graph shows how each parameter affects the other parameters and illustrates a significant relation among CPU utilization and RAM utilization. Similar behavior can be observed in Figure 9, where the RAM, Memory and storage are getting affected at peak utilization hours hence allowing DL models to map different utilization patterns to predict the exact utilization for future timestamps.

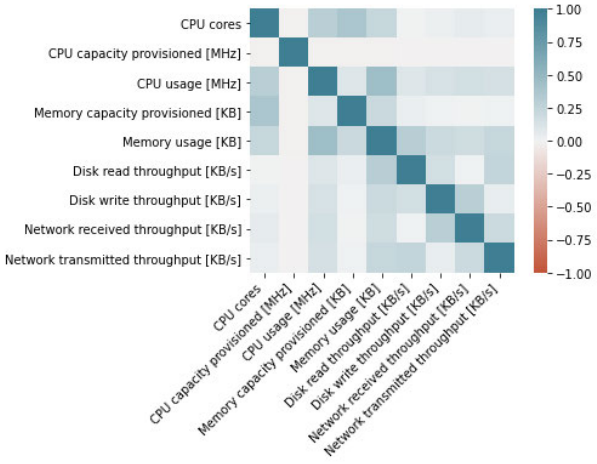


FIGURE 10. Heatplot depicts the correlation between different resource usage metrics.

2) DEEP LEARNING MODEL

Deep learning (DL) is one of the most simplistic categories of machine learning (ML). It is widely adopted because of self-learning abilities for solving complex problems, e.g., prediction, classification, detection, etc. The DL capabilities are achieved using the deep neural networks (DNN) that process the data in a feed-forward way from the input layer to several hidden layers and the final output layer. Each node or neuron in the DNN approach is directly connected with other nodes of the next layer. The first input layer has multiple input neurons that are further connected with multiple weight factors according to the design of the model and dataset features.

We have used the long-short time memory (LSTM) RNN model to predict future resource utilization, for example, VNFs resource usage RAM, CPU, etc. LSTM is a well-known DL model mostly used for natural language processing (NLP) operations, machine translation, speech and handwriting recognition, grammar learning, time-series prediction, stock price changes, etc. The LSTM model is best for time-series prediction because it keeps track of the previous data points by storing history [49], [50]. It learns the long-term dependencies and avoids the gradient explosion problem. Each hidden-node in the LSTM model has three main memory gates for storing, outputting, and forgetting the data dependencies. These memory gates are input gate i_t, output gate o_t, and forget gate f_t [51]. The sigmoid is used as an activation function in each memory gate. Hence, our VNFs resource utilization is in the time-series pattern we have used the LSTM model in our test-bed.

$$i_t = \sigma(u_i x_t + w_i h_{t-1} + b_i) \quad (1)$$

$$a_t = tanh(u_c x_t + w_c h_{t-1} + b_c) \quad (2)$$

$$f_t = \sigma(u_f x_t + w_f h_{t-1} + b_f) \quad (3)$$

$$o_t = \sigma(u_o x_t + w_o h_{t-1} + v_o c_{t-1} + b_o) \quad (4)$$

$$c_t = a_t * i_t + c_{t-1} * f_t \quad (5)$$

$$h_t = tanh(c_t) * o_t \quad (6)$$

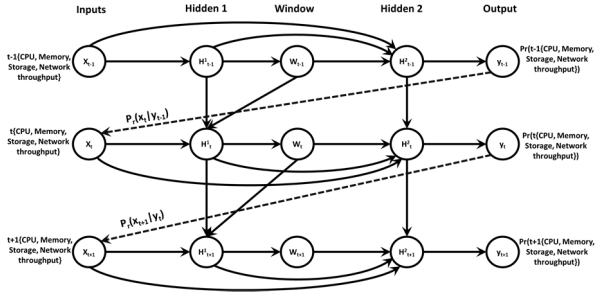


FIGURE 11. Illustrates LSTM-RNN architecture where circles show layers, solid lines show connections, and dashed lines show predictions. Inputs are the timestamped dataset, and output is predicted resources P_r at the next timestamp.

Equation 1 presents the input gate function in the LSTM model where w_i, u_i and bi, are the input weights and bias value respectively. The function for the intermediate state is depicted in equation 2, in which w_c and u_c are the input weight parameters, and b_c is bias value. On the other side, w_f and u_f and are the weighting factors and b_f is the bias value of the forget gate function gate f_t in equation 3. Equation 4 illustrates the output gate function o_t in which u_o, w_o, v_o, b_o are the input weight parameters and bias value. The cell state information is presented in equation 5, whereas a_t, i_t, and f_t are the intermediate, input, and forget gates. The equation 6 presents the objective function of new hidden-state h_t. Furthermore, tanh and sigmoid are the activation functions, h_{t-1} is the current time-interval, and (*) is element-wise multiplication operator [52].

Figure 11 illustrates the architecture of the LSTM model with the input data and predicted output. The data is provided as three consecutive timestamp windows X_{t-1}, X_t, X_{t+1}, and Y_{t+1} as the model's output. In this architecture, each layer of the RNN feeds its output $P_r(y_{t-1})$ to the next layer and we introduced direct connections from the inputs to all hidden layers and from hidden layers to the outputs shown in Figure 11. P_r is the predicted output at the next timestamp. On runtime for each input timestamp LSTM model predicts the future utilization at timestamp $t+1$ as $P_r(t+1)$. Now the intent update engine invokes the DL model for n timestamps and store the predicted outputs in the form of P matrices as $P = p_{t+1}, p_{t+2}, p_{t+3}, \ldots\ldots, p_{t+n}$. The update engine divides the P matrices into three as P_{cpu}, P_{mem}, and $P_{storage}$.

We have implemented the LSTM model using opensource Keras and other supporting libraries in Python. The LSTM model structure is as follows: one input layer with ten neurons, three hidden layers with 8 neurons, and one output layer with four neurons for predicting the resource CPU, RAM, storage, and throughput usage. The 0.25 dropout is used in the LSTM unit to avoid the overfitting issue; The Adam optimizer is used as an optimization algorithm, sigmoid as an activation function while training the model. In addition, the mean square error (MSE) is used to calculate the loss value. Moreover, for efficient training and to prevent overfitting, we have used 80% dataset samples for training and 20% for testing purposes.

TABLE 2. Notations used in mathematical model.

Notations	Description
V_x	*Storage Allocated to VNF or VM*
V_y	*CPU Allocated to VNF or VM*
V_z	*Memory Allocated to VNF or VM*
S_t	*State of the VM machine*
$F(S_t)$	*Decision Function*
1, 0, -1	*(Overload, Normal, Under-Utilized)*
T_r^h	*High threshold for overload state*
T_r^l	*Low threshold for under-utilized state*

3) DECISION ENGINE

The decision engine takes the predicted resource usage from the LSTM model and decides whether the resources' updates are needed. It works based on the mathematical model. The decision engine results are further forwarded to the IBN platform in the form of smart alerts or alarms.

- System Model: This system uses dynamic thresholding by applying K-means clustering with average inter-quartile range algorithm for deciding the updates [53], [54]. The K-mean clustering algorithm divides the predicted P dataset into n groups resulting in $G_r = G_1, G_2, G_3, \ldots\ldots, G_n$. After that, the update engine calculates each group's average utilization using equation 7, resulting in n G_r^{avg}. Equation 8 considers the calculation of Inter-Quartile range (IR) based on the averaged means calculated over n Groups. Using the IR update engine calculates upper and lower thresholds, which results in deciding the resource scaling. The elaboration of each notation used in our mathematical model is detailed in Table 2.

$$G_r^{avg} = \frac{n_{j+1}, n_{j+2}, n_{j+3}, \ldots, n_{j+m}}{T_i} \tag{7}$$

$$IR = Q_3 - Q_1 \tag{8}$$

$$T_r^h = 1 - R * IR \tag{9}$$

$$T_r^l = 0.5\,(1 - R * IR) \tag{10}$$

Equation 9, contains the calculation of the upper threshold value where R defines how aggressively the compute nodes VMs are allocated. The higher R means less optimal resource utilization but better SLA fulfillment. Hence, dependent upon the type of service, the value of is determined. Therefore, R will be higher for slices requiring higher SLA and service provisioning cannot be compromised. For regular slice types, the update engine can have a reduced value for R. Hence, this thresholding approach allows us to scale up VM resources in clouds while considering service type constraints. After that, based on the upper and lower thresholds, the update engine determines the predicted utilization states of the CPU, RAM, and storage as V_x, V_y, and V_z, respectively, using the equation 11, 12, and 13. After classifying each of the VM resource parameter statuses, we calculate the state of VM by using Equation 14. Based on the results of equation 14 the decision of scaling is performed using equation 15.

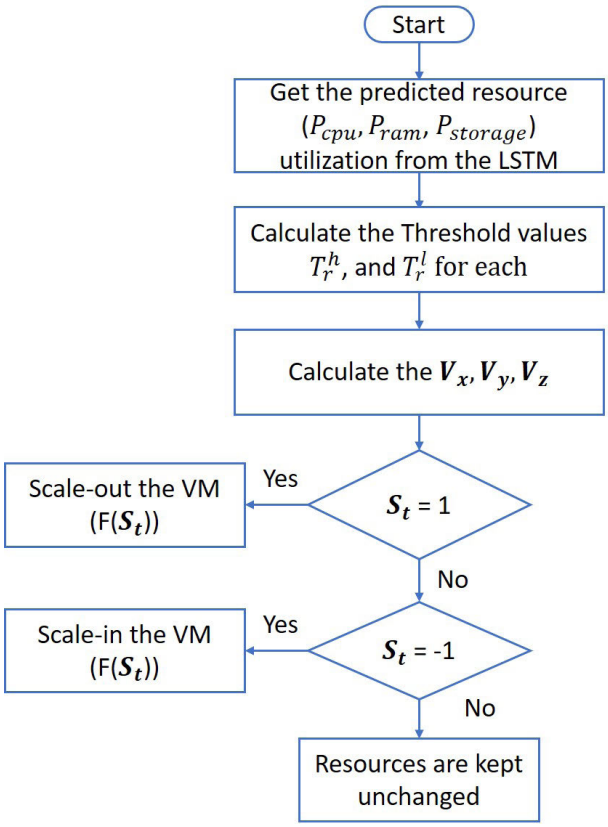


FIGURE 12. Illustrates the flowchart for deciding the scaling decision based on predicted resource utilization.

$$V_x = \begin{cases} 1, & \text{if } x \geq T_r^h \\ 0, & \text{if } x > T_r^l \text{ and } x < T_r^h \\ -1, & \text{if } x \leq T_r^l \end{cases} \tag{11}$$

$$V_y = \begin{cases} 1, & \text{if } y \geq T_r^h \\ 0, & \text{if } y > T_r^l \text{ and } y < T_r^h \\ -1, & \text{if } y \leq T_r^l \end{cases} \tag{12}$$

$$V_z = \begin{cases} 1, & \text{if } z \geq T_r^h \\ 0, & \text{if } z > T_r^l \text{ and } z < T_r^h \\ -1, & \text{if } z \leq T_r^l \end{cases} \tag{13}$$

$$S_t(V_x, V_y, V_z) = \begin{cases} 1, & \text{if}(V_x, V_y, V_z) \geq T_r^h \\ 0, & \text{if } (V_x, V_y, V_z) > T_r^l \\ & \text{and } (V_x, V_y, V_z) < T_r^h \\ -1, & \text{if } (V_x, V_y, V_z) \leq T_r^l \end{cases} \tag{14}$$

$$F(S_t) = \begin{cases} Scale - out, & \text{if } S_t = 1 \\ Normal, & \text{if } S_t = 0 \\ Scale - in, & \text{if } S_t = -1 \end{cases} \tag{15}$$

Figure 12 illustrates the flowchart for deciding resource scaling based on the predicted utilization of VM resources. It takes VM resource utilization as input and calculates the dynamic thresholding factor for

TABLE 3. Experimental setup specifications and configurations.

Component:	Specifications:
SDR-USRP B210	Channels: 2Rx * 2Tx Frequency-Range: 70(MHz) to 6(GHz)
NFVO-OSM	Operating-System: UBUNTU-18.04 Processor: 32 Cores 2.10 GHZ Memory: 252 GB OpenStack-Version: stein-version OSM-Version: V7
SDN-RAN Controller	Operating-System: UBUNTU-18.04 LTS Processor: Core i7 (3.0 GHz) RAM: 32GB
IBN tool	Operating-System: Window-10 Memory: 32GB Processor: Core I5 (3.0GHZ) Programming-languages: Python, JAVA

each resource utilization. After that, it determines the VM state be overloaded or underloaded. After determining VM state, the decision engine decides to scale out the VM, second scale in the VM, or finally leave the system without changing.

As discussed above, We have used the MATERNA dataset for the training of our LSTM model. The 80% dataset is used for training, and 20% is for testing purposes. After training the model, it works fine, and the prediction accuracy is excellent, almost 94% on the MATERNA dataset. For integrating the LSTM model with our IBN system, we have set-up an FTP server that collects the data of the VMs from monitoring tools, and the model can take that real-time data and perform prediction in each 10s. However, the IBN platform update engine can check the model predictions in each 10s, and it triggers an alarm whenever the CPU, storage and memory of the VMs reach a threshold. After getting the requirements update request from the update engine, IBN instantiates the configuration for activating the duplicate resources to guarantee the QoS. Besides, in case of failure, IBN also performs the recovery operations. Currently, we are updating the resources based on CPU, storage and RAM statistics of the virtual resources. Hence, our update engine makes IBN an intelligent platform that provides an automated way to update, assure, and recover the network virtual resources.

V. EXPERIMENTAL TEST-BED IMPLEMENTATION

Figure 13 shows the e2e slice orchestration and management test-bed that comprises the IBN platform, NFVO OSM, FlexRAN RAN controller, a monitoring system, OAI EPC, OAI eNodeB, and OAI UEs. The OAI (OpenAirInterface) is an open-source community that provides the implementation of EPC, eNB and UE [55]. The OSM is responsible for the orchestration and management of network services and deploys the requested resources through its communication with OpenStack at the physical and virtual infrastructure. Further, OpenStack provides virtualization support for the deployment of core and RAN network VNFs.

Our test-bed comprises OAI-based Core and RAN network functions. We have deployed the dedicated OAI EPC VNFs for each slice with the help of OSM. The OAI EPC contains the LTE VNFs: vMME (virtual mobility management entity), vHSS (virtual home subscription server), vSPGW (virtual serving and packet gateway). Nevertheless, we have used the Open-Source EPC solution named NextEPC, a modified OAI EPC compatible with the OSM orchestrator. Our IBN system creates the VNFFG/slice template in JSON string and sends it to the OSM through Rest API for preparing the requested service resources (VMs). The slice template has all the information regarding the VNFs instances, images, CPU, and memory. In this way, with OpenStack's help, OSM can design and orchestrate the core network functions for a slice.

On the other side, the FlexRAN controller had been deployed for slicing and controlling the RAN resources. We have used a separate PC equipped with 16GB RAM, i5 7400 CPU and Ubuntu 18.04 for running the FlexRAN controller. The OAI eNodeB has been used as a RAN in our test-bed with SDR (software-defined radio) USRP (universal software radio peripheral) B210. The USRP B210 is connected to eNodeB Radio Resource Control (RRC), and it provides the LTE radio signals for the attachment of simulated UEs. A PC is used with USB port 3.0, Ethernet 1GBit/s, 16 GB RAM, and a Core i5 CPU to install OAI eNodeB with USRP B210. Two eNodeBs with two USRP B210 have been deployed that form a shared pool of RAN resources. Moreover, the FlexRAN controller controls and manages the RAN domain, and creates and deletes the slice on this domain. The IBN generates a slice template in JSON format for the FlexRAN controller, and in return, the FlexRAN controller enforces the requested slice template on the eNodeB for slice creation. An associated vMME with dedicated core network functions is assigned to the RAN slice that establishes an e2e connection. The multiple simulated OAI UEs are used for testing the performance of the proposed mechanism. The IBN system automates the slice template generation process and dynamically manages the RAN and core resources. Moreover, the monitoring tools continuously monitor the core and RAN network slice resources on run-time. Table 3 presents the details of the experimental setup.

Figure 14 presents the web-based GUI portal of the IBN platform, where users can define their service requirements in the form of intents. Afterwards, these intents are translated by the policy configurators into slice template for underlying orchestrators. The generated slice template is forwarded to OSM and FlexRAN controller to deploy the slice resources over the core and RAN domains. The policy configurators of the IBN communicate with the OSM and RAN controller through the Rest-API. Figure 15 shows the deployment status of core network (EPC) VNFs deployed through the IBN portal using OSM and OpenStack.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

The slice template defines the resource requirements in terms of CPU, memory, bandwidth, instance images, repository, etc. The RAN slice template includes the QoS (downlink and uplink), slice-ID, service type, and Public Land Mobile

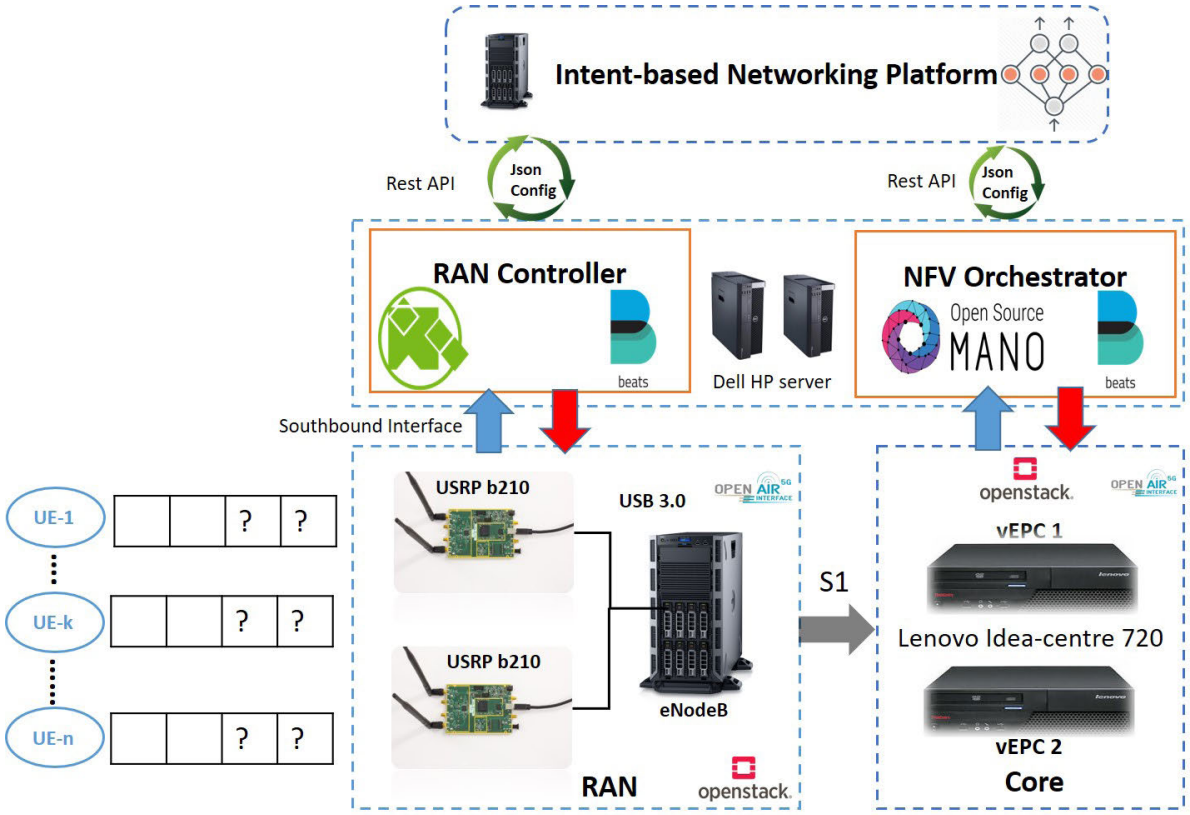


FIGURE 13. Experimental test-bed setup.

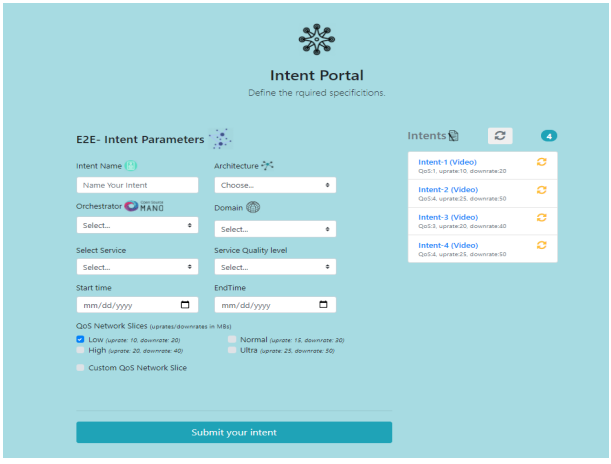


FIGURE 14. GUI of the IBN platform.

Network (PLMN)-ID for a slice; downlink and uplink requirements are further converted into required radio resource blocks (RBs). On the other side, we are also using the static approach for assigning the radio resources with percentages for creating the slice—for example, 50%, 30%, and 20% of the RAN resource. For the core network, the dedicated EPC VNF resources are allocated to each radio slice. The EPC components such as vMME, vHSS, and vSPGW configurations are added to the RAN slice template at the time of creation. In this way, our IBN performs the slicing for core and RAN domains.

We have tested the proposed system by creating multiple slices with different QoS of three categories: Guaranteed Bit Rate (GBR), non-GBR, and URLLC slice. The GBR slice is a fixed slice that provides the exact resources in terms of throughput specified in the contract. The second non-GBR is a type of best effort slice or eMBB slice that can dynamically allocate the slice resources. Whenever the intents with eMBB or non-GBR slice type are inserted, the IBN system calculates the resources and assigns the best resources for a slice; if enough resources are available, it gives the maximum resource. It allocates the minimum available resource in case of fewer resources being available at that time. Furthermore, in our system, we have used the highest priority for low latency slices to achieve strict requirements. So, the IBN system can also allocate the best RAN resources to the URLLC category.

The iPerf tests have been performed in order to check the stability of our test-bed. Figure 16 shows the downlink throughput results for multiple deployed slices with different QoS requirements. In Figure 16a, we deploy the two static GBR and non-GBR slices with 50% RAN resources for each. The results show an almost equal throughput of 28 MB/s for GBR and 27 MB/s for non-GBR slice due to the same resources. In Figure 16b, four slices are instantiated with different bandwidth requirements: GBR slice

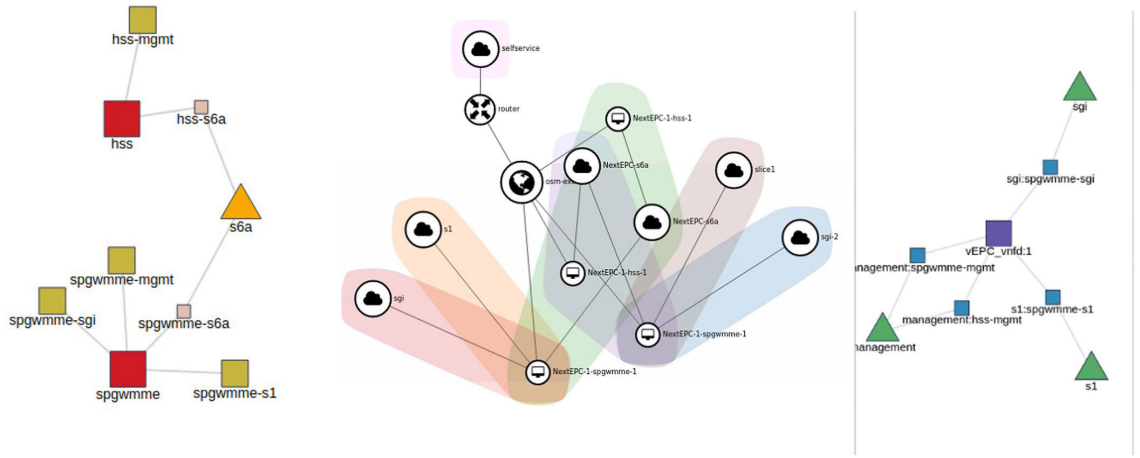
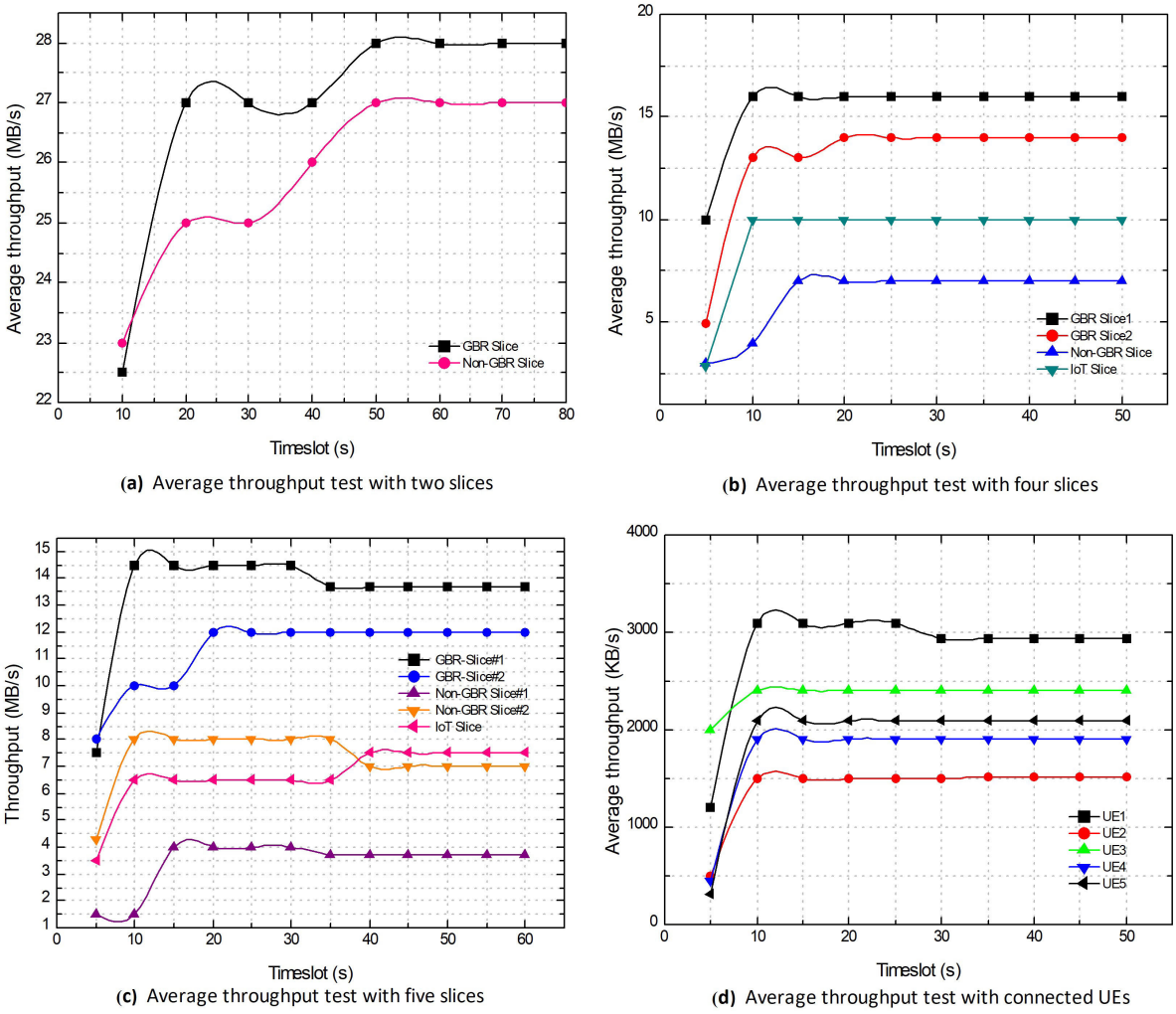


FIGURE 15. Deployment status of the core network EPC NFs through OSM and OpenStack.



(a) Average throughput test with two slices

(b) Average throughput test with four slices

(c) Average throughput test with five slices

(d) Average throughput test with connected UEs

FIGURE 16. Throughput results by creating multiple slices through IBN platform.

1 with 30%, GBR slice 2 with 30%, non-GBR slice with 20%, and IoT slice with 20% of the RAN resources. The maximum throughput of 16 MB/s and 14 MB/s have been recorded in multiple tests for GBR slice1 and slice2, respectively. After that, an IoT slice with 20% resources is deployed, and it achieves 10 MB/s throughput. Lastly, the non-GBR

slice is deployed with 20% resources which achieves 6 MB/s throughput. The 3GPP EPC service request initially follows a handshake procedure for allocating slices. For the initial registration procedure, user equipment requests the assignment of the resources. The specified slice resources are allocated to the requested UE. At the start, the UE does not stream the full capacity. After the resource reservation, the slice with specified PLMN and gateway is provided to the UE for streaming. Hence, original traffic starts streaming after the initial 3GPP handshake and registration procedure for every incoming UE. Our system schedules the resources on a priority basis: GBR slice category has top priority, IoT slice has a second top, and non-GBR is best effort slice. If the system has enough resources, it allocates full resources according to contract, but in fewer resources, the system allocates minimum available RAN resources to this category. On the other side, IoT slice and non-GBR slice are deployed with the same contract, but due to low slice priority, the system allocates fewer resources to non-GBR slice. However, we achieve satisfactory throughput for each of the deployed slices. The reason for varying results is because of USRP B210 based RAN, which is not a standardized solution.

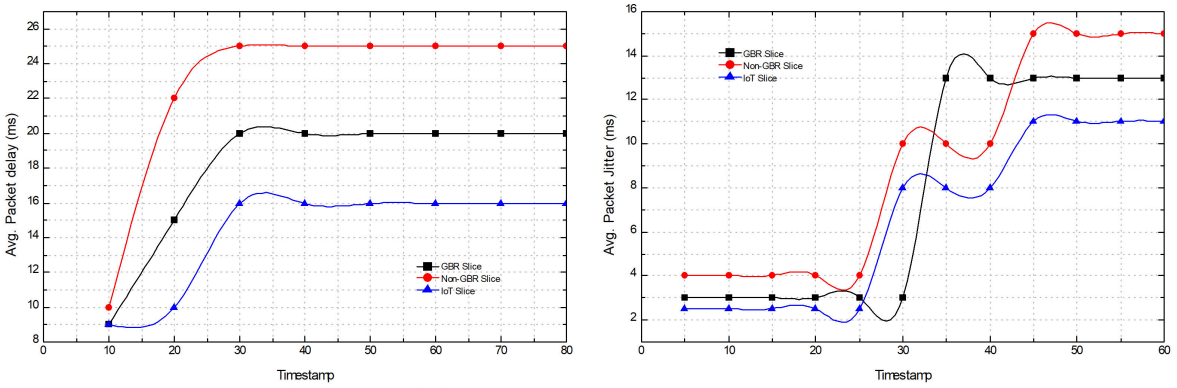
Figure 16c presents the results of five deployed slices through the IBN system with varying QoS requirements in terms of RAN resources, such as 30% resources for each GBR slice1 and slice2, and the recorded throughput was 15 MB/s and 12 MB/s, respectively. Two non-GBR slices with 10% and 15% resources are executed, and 4 MB/s and 7 MB/s throughput are achieved. Also, an IoT slice with 15% resources is implemented, which achieved maximum of 8.1 MB/s throughput. Besides, in Figure 16d, we use simulated OAI UEs to test slice performance with 20% of the RAN resources. The five UEs are connected with one deployed static slice, and they achieved a maximum of 3000 KB/s, 2200 KB/s, 2050 KB/s, 2000 KB/s, and 1500 KB/s throughput. The calculated average packet delay and average packet jitter for three GBR, non-GBR, and IoT slices are depicted in Figures 17a and 17b. The delay is the time taken for a packet to move between two endpoints, whereas jitter is the difference in delay between two received packets. As mentioned above, due to the priority-based scheduling mechanism, the packet delay for the non-GBR is more significant than GBR and IoT slices. In the IoT slice, the recorded packet delay is minimal, and the RAN scheduler always assigns more resources to guarantee QoS requirements. As shown in Figure 17b, similar to the average delay, the jitter is also calculated and recorded in the same manner, maximum for a non-GBR slice and minimum for an IoT low latency slice. The results show the promising performance of the system in terms of bandwidth, delay, and latency.

Moreover, if a slice is requested, the current users consume all the slice resources and cannot be served to additional users. The solution to this problem is our intelligent update and assurance module that can scale-out and scale-in the resources on demand. This module has a pre-trained LSTM model that predicts and forecasts the slice resources statistics to the IBN system. The model takes the collected data as input provided by the monitoring tools and triggers whenever resources are overloaded or under-utilized. We have performed the test by creating a slice and started streaming and sending connection requests to overload the slice resources. Our system can update the slice resource in case of overloaded or under-utilized conditions and allocate the duplicate resources for new slice users.

A. LSTM-BASED RESOURCE ASSURANCE AND UPDATE

The LSTM-based resource scaling test was performed on EPC while simultaneously instantiating GBR and Non-GBR slices, as shown in Figure 18. The results show the effectiveness of LSTM in scaling resources while comparing the increasing number of users and traffic on the x-axis with the utilization of resources on the y-axis. The threshold value for resource utilization of GBR and Non-GBR was set to 80% and 50%, respectively. The graph considers real-time resource utilization of each slice instance with the increasing number of users and traffic intensities. The LSTM-based scaling mechanism prevents the system from reaching the threshold value and scales the system accordingly. In the case of a NON-GBR slice, scaling decision was first implemented at interval 4, as LSTM predicted the over-utilization of GBR VNF resources. However, it did not reach the threshold value, so the increasing traffic trend made it evident to LSTM to scale the slice proactively. Similarly, at instance 9, a third slice was instantiated for the non-GBR slice to keep the traffic and resource utilization below the threshold value. Similarly, in GBR, two scale-out functions were performed by the LSTM-based update mechanism at instances 4 and 10 to keep the accumulative resource utilization of VNF for slices below the threshold value. In GBR slice A, from instance 7 to instance 10, the utilization goes down because the resource allocated to GBR slice A was being handled by GBR slice B due to the FlexRAN slice selection procedure. If the LSTM-based update procedure is not applied, one of the following two can occur between these two intervals: 1) instantiation of three slices beforehand and preventing the system from reaching the threshold value. 2) instantiation of one or two slices and allowing the system to reach the threshold value, eventually resulting in performance degradation. The performance degradation is against the key idea of service assurance of the IBN-based mechanism. Hence, using a proactive and dynamic update system will provide efficient assurance and optimize resource utilization.

With the advantages of the presented system, there are few limitations. Currently, we are slicing the RAN resource with a static resource allocation mechanism. So, an efficient algorithm for radio resource management would be needed for assuring dynamic resource allocation. The mobility management for the users is another limitation of the proposed system. Furthermore, another important issue is the unavailability of open-source 5G components, e.g., gNodeB and core NFs. Currently, all the academic researchers have been using LTE components for realizing the 5G networks.



(a) Average packet delay per created slice (b) Average packet jitter per created slice

FIGURE 17. Average packet delay and average jitter per packet for each slice category.

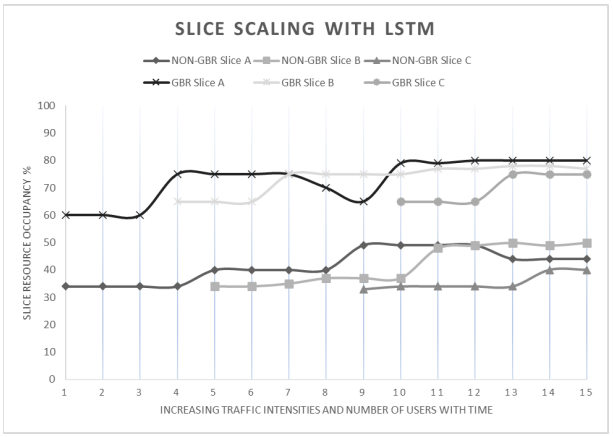


FIGURE 18. LSTM-based slice resource scaling with time while considering slice VNFs resource occupancy threshold percentage.

Furthermore, in our test-bed, we use the SDR USRP B210 devices as a physical radio to realize the RAN capabilities that is not a standardized industrial solution. The aim of the test results is not to test the bandwidth with the industrial standards but to test whether the proposed IBN system can orchestrate and deploy the slice resources according to the intents. The performance results show the consistency of our test-bed. It is a step towards the zero-touch system. Hence, our system can automate the slice creation and deletion process by taking higher-level QoS requirements for a slice. It can also manage the lifecycle of the network slice instance automatically.

VII. CONCLUSION AND FUTURE WORK

This paper presents an IBN platform that automates and manages the network slice lifecycle for the multi-domain environment. It is a one-touch system where the user needs to define higher-level service requirements through GUI. IBN translates the received QoS requirements for a slice into policy configurations/slice templates through domain policy configurators and forwards them to underlying orchestrators to automatically activate resources over the infrastructure. The OSM and FlexRAN controller are responsible for handling the core network and RAN slices. This system can perform the e2e network slicing in a customizable and flexible fashion. Moreover, the DL-based intent update engine makes IBN an intelligent platform that can update the run-time resources on demand. The update engine receives the future resource usage prediction results from the LSTM model, and it instantiates the update request to IBN whenever needed. Several tests have been performed by creating multiple slices of core and RAN network domains in our test-bed, which shows promising results in terms of resource provisioning, resource customization, resource isolation, and resource assurance. In the future, we aim to make IBN more intelligent by introducing DL-based intent translation mechanisms. In addition, a reinforcement learning-based mechanism will be developed for slicing and allocating the RAN resources according to the QoS requirements.

REFERENCES

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[3] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi, "5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view," *IEEE Access*, vol. 6, pp. 55765–55779, 2018.

[4] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Comput. Netw.*, vol. 167, Feb. 2020, Art. no. 106984.

[5] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network slicing-based customization of 5G mobile services," *IEEE Netw.*, vol. 33, no. 5, pp. 134–141, Sep. 2019.

[6] T. S. 3GPP. *5G; Management and Orchestration; Concepts, Use Cases and Requirements*. Accessed: Mar. 4, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128530/15.00.00_60/t%s_128530v150000p.pdf

[7] S. IETF. *Network Slicing Management and Orchestration*. Accessed: Mar. 4, 2021. [Online]. Available: https://tools.ietf.org/id/draft-flinck-slicing-management-00.xml#rfc.se%ction.3

[8] R. Su, D. Zhang, R. Venkatesan, Z. Gong, C. Li, F. Ding, F. Jiang, and Z. Zhu, "Resource allocation for network slicing in 5G telecommunication networks: A survey of principles and models," *IEEE Netw.*, vol. 33, no. 6, pp. 172–179, Nov. 2019.

[9] F. Meneses, M. Fernandes, D. Corujo, and R. L. Aguiar, "SliMANO: An expandable framework for the management and orchestration of end-to-end network slices," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–6.

[10] X. Li, R. Ni, J. Chen, Y. Lyu, Z. Rong, and R. Du, "End-to-end network slicing in radio access network, transport network and core network domains," *IEEE Access*, vol. 8, pp. 29525–29537, 2020.

[11] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network slices toward 5G communications: Slicing the LTE network," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 146–154, Aug. 2017.

[12] H. D. R. Albonda and J. Perez-Romero, "An efficient RAN slicing strategy for a heterogeneous network with eMBB and V2X services," *IEEE Access*, vol. 7, pp. 44771–44782, 2019.

[13] IETF. *Intent-Based Networking*. Accessed: Dec. 26, 2020. [Online]. Available: https://tools.ietf.org/html/draft-irtf-nmrg-ibn-concepts-definitions-01%

[14] Cisco. *Intent-Based Networking*. Accessed: Dec. 26, 2020. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-netwo%rks/digital-network-architecture/nb-09-intent-networking-wp-cte-en.pdf

[15] 5GPPP. *View on 5G Architecture: White Paper*. Accessed: Feb. 1, 2021. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-Whi%te-Paper-Jan-2018-v2.0.pdf

[16] H. Packard. *Network Slice Configuration and Service Slice Lifecycle Management*. Accessed: Feb. 1, 2021. [Online]. Available: https://h20195.www2.hpe.com/v2/Getdocument.aspx?docname=a00065972enw&skiphtml=1&&&

[17] Samsung. *Techn Ical White Paper: Network Slcing*. Accessed: Feb. 2, 2021. [Online]. Available: https://images.samsung.com/is/content/samsung/p5/global/business/networ%ks/insights/white-paper/network-slicing/200420_Samsung_Network_Slicing_Final.p%df

[18] A. Devlic, A. Hamidian, D. Liang, M. Eriksson, A. Consoli, and J. Lundstedt, "NESMO: Network slicing management and orchestration framework," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 1202–1208.

[19] S. Miladić-Tešić, G. Marković, D. Peraković, and I. Cvitić, "A review of optical networking technologies supporting 5G communication infrastructure," *Wireless Netw.*, pp. 1–9, Mar. 2021, doi: 10.1007/s11276-021-02582-6.

[20] OSM. *Open Source Mano*. Accessed: Feb. 2, 2021. [Online]. Available: https://osm-download.etsi.org/ftp/Documentation/201902-osm-scope-white-%paper/#!02-osm-scope-and-functionality.md

[21] ETSI. *Network Transformation; (Orchestration, Network and Service Management Framework)*. Accessed: Jan. 21, 2021. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/ETSI_White_Paper_Netw%ork_Transformation_2019_N32.pdf

[22] ONAP. *Onap: Open Networking Automation Platform*. Accessed: Feb. 3, 2021. [Online]. Available: https://www.onap.org/

[23] OpenBaton. *Openbaton: Nfv Mano-Based Framework*. Accessed: Feb. 3, 2021 [Online]. Available: https://openbaton.github.io/

[24] K. Katsalis, N. Nikaein, and A. Huang, "JOX: An event-driven orchestrator for 5G network slicing," in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2018, pp. 1–9.

[25] Cloudify. *Cloudify: A Open Source Network Orchestrator*. Accessed: Feb. 3, 2021. [Online]. Available: https://cloudify.co/

[26] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1311–1324, Jun. 2020.

[27] X. Shen, J. Gao, W. Wu, K. Lyu, M. Li, W. Zhuang, X. Li, and J. Rao, "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, 2020.

[28] P. W. Khan, K. Abbas, H. Shaiba, A. Muthanna, A. Abuarqoub, and M. Khayyat, "Energy efficient computation offloading mechanism in multi-server mobile edge computing—An integer linear optimization approach," *Electronics*, vol. 9, no. 6, p. 1010, Jun. 2020.

[29] M. Afaq, J. Iqbal, T. Ahmed, I. Ul Islam, M. Khan, and M. S. Khan, "Towards 5G network slicing for vehicular ad-hoc networks: An end-to-end approach," *Comput. Commun.*, vol. 149, pp. 252–258, Jan. 2020.

[30] V. Q. Rodriguez, F. Guillemin, and A. Boubendir, "5G E2E network slicing management with ONAP," in *Proc. 23rd Conf. Innov. Clouds, Internet Netw. Workshops (ICIN)*, Feb. 2020, pp. 87–94.

[31] R. Inam, A. Karapantelakis, K. Vandikas, L. Mokrushin, A. Vulgarakis Feljan, and E. Fersman, "Towards automated service-oriented lifecycle management for 5G networks," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2015, pp. 1–8.

[32] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani, "A survey on intent-driven networks," *IEEE Access*, vol. 8, pp. 22862–22873, 2020.

[33] Y. Wei, M. Peng, and Y. Liu, "Intent-based networks for 6G: Insights and challenges," *Digit. Commun. Netw.*, vol. 6, no. 3, pp. 270–280, Aug. 2020.

[34] E. Zeydan and Y. Turk, "Recent advances in intent-based networking: A survey," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC-Spring)*, May 2020, pp. 1–5.

[35] Huawei. *Intent-Driven Network*. Accessed: Feb. 25, 2021. [Online]. Available: https://carrier.huawei.com/~/media/CNBG/Downloads/Spotlight/all-cloud-n%etwork-towards-5g/idn-en.pdf

[36] Apstra. *Intent-Based Networking: A Next-Gen Vision for the Next-Gen Network*. Accessed: Feb. 25, 2021. [Online]. Available: https://go.apstra.com/white-paper-apstra-intent-based-networking

[37] D. Sattar and A. Matrawy, "Towards secure slicing: Using slice isolation to mitigate DDoS attacks on 5G core network slices," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Jun. 2019, pp. 82–90.

[38] V. A. Cunha, E. da Silva, M. B. de Carvalho, D. Corujo, J. P. Barraca, D. Gomes, L. Z. Granville, and R. L. Aguiar, "Network slicing security: Challenges and directions," *Internet Technol. Lett.*, vol. 2, no. 5, p. e125, Sep. 2019.

[39] 3GPP. *Telecommunication Management; Study on Management and Orchestration of Network Slicing for Next Generation Network*. Accessed: Feb. 25, 2021. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.a%spx?specificationId=3091

[40] T. A. Khan, M. Afaq, K. Abbas, A. Rafiq, A. Mehmood, and W.-C. Song, "Generic intent-based networking approach for end-to-end slice orchestration and lifecycle management," *J. Korean Inst. Commun. Sci.*, pp. 468–469, Feb. 2020.

[41] K. Abbas, M. Afaq, T. A. Khan, A. Rafiq, and W.-C. Song, "Slicing the core network and radio access network domains through intent-based networking for 5G networks," *Electronics*, vol. 9, no. 10, p. 1710, Oct. 2020.

[42] A. Rafiq, A. Mehmood, T. A. Khan, K. Abbas, M. Afaq, and W.-C. Song, "Intent-based end-to-end network service orchestration system for multi-platforms," *Sustainability*, vol. 12, no. 7, p. 2782, Apr. 2020.

[43] K. Abbas, M. Afaq, T. A. Khan, A. Mehmood, and W.-C. Song, "IBNSlicing: Intent-based network slicing framework for 5G networks using deep learning," in *Proc. 21st Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2020, pp. 19–24.

[44] T. A. Khan, A. Mehmood, J. J. Diaz Ravera, A. Muhammad, K. Abbas, and W.-C. Song, "Intent-based orchestration of network slices and resource assurance using machine learning," in *Proc. NOMS IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2020, pp. 1–2.

[45] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2016, pp. 427–441.

[46] O. S. Mano. *Osm Performance Management*. Accessed: Feb. 2, 2021. [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Performance_Management

[47] Gitlab:MOSAIC5G. *Elasticmon Manual*. Accessed: Nov. 7, 2020. [Online]. Available: https://gitlab.eurecom.fr/mosaic5g/mosaic5g/-/wikis/tutorials/elasticmo%n-manual

[48] Materna. *Dataset*. Accessed: Jan. 3, 2021. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna/

[49] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," *Phys. D, Nonlinear Phenomena*, vol. 404, Mar. 2020, Art. no. 132306.

[50] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," Google, New York, NY, USA, Tech. Rep., 2014. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43905.pdf

[51] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[52] S. P. Sone, J. J. Lehtomaki, and Z. Khan, "Wireless traffic usage forecasting using real enterprise network data: Analysis and methods," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 777–797, 2020.

[53] Z. Zhou, Z. Hu, and K. Li, "Virtual machine placement algorithm for both energy-awareness and SLA violation reduction in cloud data centers," *Sci. Program.*, vol. 2016, pp. 1–11, Mar. 2016.

[54] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," *MGC@ Middleware*, vol. 4, Dec. 2010, Art. no. 1890799.
[55] OpenAirInterface. *Oai: An Open-Source Community*. Accessed: Dec. 20, 2020. [Online]. Available: https://www.openairinterface.org

KHIZAR ABBAS received the B.S. degree in software engineering from the Government College University Faisalabad (GCUF), Pakistan, in 2014, and the M.S. degree in computer science from the University of Agriculture Faisalabad, Pakistan, in 2017. He is currently pursuing the Ph.D. degree in computer engineering with the Network Convergence Laboratory, Jeju National University, Jeju, South Korea.

He worked as a Lecturer with the Department of Computer Science, Government College University Faisalabad, Pakistan, and also with the Department of Computer Science, University College of Management and Sciences (UCMS), Khanewal, Pakistan. His research interests include software-defined networks, 5G, network slicing, network function virtualization, cloud computing, mobile edge computing, blockchain, and machine learning.

TALHA AHMED KHAN received the B.S. degree in computer science from the FAST National University of Computer and Emerging Sciences (FAST NUCES), Pakistan, and the M.S. degree in computer engineering from Jeju National University, South Korea, in 2019, where he is currently pursuing the Ph.D. degree in computer engineering.

His research interests include the SDN, NFV, 5G mobile networks, intent-based networking, network orchestration, mobile edge computing, and VNF development.

MUHAMMAD AFAQ received the B.S. degree in electrical engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2007, the M.S. degree in electrical engineering with emphasis on telecom from the Blekinge Institute of Technology, Sweden, in 2010, and the Ph.D. degree in computer engineering from Jeju National University, in 2017.

Before starting his Ph.D., he worked as a Research Associate with the Faculty of Computer Science and Engineering, GIK Institute of Engineering Sciences and Technology, Pakistan, and as a Lecturer with the Department of Electrical Engineering, City University of Science and Information Technology. He also worked as an Assistant Professor with the Department of Computer Science and Information Technology, Sarhad University of Science and Information Technology, Pakistan. He is currently working as a Postdoctoral Researcher with the Network Convergence Laboratory, Jeju National University. His research interests include cloud computing, software-defined networking, network function virtualization, wireless networks and protocols, machine learning, and data science.

WANG-CHEOL SONG received the B.S. degree in food engineering and the B.S., M.S., and Ph.D. degrees in electronics from Yonsei University, Seoul, South Korea, in 1986, 1989, 1991, and 1995, respectively.

He has been a Full Professor with the Department of Computer Engineering, Jeju National University, South Korea, since 1996. He has been responsible for the National Software Program with Jeju National University, since 2018. His research interests include VANETs and MANETs, SDN/NFV, intent-based networking, and network management.