Computers, Materials & Continua
DOI:10.32604/cmc.2022.017118
Article

Tech Science Press

An Intent-Driven Closed-Loop Platform for 5G Network Service Orchestration

Talha Ahmed Khan, Khizar Abbas, Afaq Muhammad and Wang-Cheol Song*

Department of Computer Engineering, Jeju National University, Jeju, Korea
*Corresponding Author: Wang-Cheol Song. Email: philo@jejunu.ac.kr

Received: 21 January 2021; Accepted: 15 July 2021

Abstract: The scope of the 5G network is not only limited to the enhancements in the form of the quality of service (QoS), but it also includes a wide range of services with various requirements. Besides this, many approaches and platforms are under the umbrella of 5G to achieve the goals of end-to-end service provisioning. However, the management of multiple services over heterogeneous platforms is a complex task. Each platform and service have various requirements to be handled by domain experts. Still, if the next-generation network management is dependent on manual updates, it will become impossible to provide seamless service provisioning in runtime. Since the traffic for a particular type of service varies significantly over time, automatic provisioning of resources and orchestration in runtime need to be integrated. Besides, with the increase in the number of devices, amount, and variety of traffic, the management of resources with optimization becomes a challenging task. To this end, this manuscript provides a solution that automates the management and service provisioning through multiple platforms while assuring various aspects, including automation, resource management and service assurance. The solution consists of an intent-based system that automatically manages different orchestrators, and eliminates manual control by abstracting the complex configuration requirements into simple and generic contracts. The proposed system considers handling the scalability of resources in runtime by using Machine Learning (ML) to automate and optimize service resource utilization.

Keywords: IBN; ML; 5G; NFV; XOS; OSM; SDN; ONOS; OpenStack

1 Introduction

The variety and diversity in terms of the requirement impose multi-dimensional challenges for the network operators, including optimal handling of diverse network traffics, handling multiple domains, opting from various network components, and managing multiple underlying infrastructure managers. With every new application, a different pattern of requirements appears, demanding complex changes in the underlying network. The uniqueness in requirements due to various applications causes a complex task for the next-generation network's configuration and management. Existing legacy LTE networks do not have the capabilities to cater to diverse applications while considering the variety in terms of requirements. Therefore, 5G is enabled with network slicing, allowing the orchestration of multiple virtual resources over a single physical infrastructure [1–3]. The abstraction of computation from physical infrastructure through virtualization serves as a building block in developing advanced computing paradigms. It is essential to realize that for fulfilling a wide variety of service requirements, 5G also introduces virtualization of its components through Network Function Virtualization (NFV). Specifically, NFV enables the development of different network functions that can operate on generic hardware infrastructure. Also, it enables the shifting of the telco central office to the cloud; hence the programmable virtual resources in the cloud eliminate the scarcity of the resources. However, the allocation of resources in the cloud has a cost, and it highly affects the operating expenses (OPEX) of the network. Also, if optimal approaches are not considered, it reduces the energy efficiency of the infrastructure. In addition to that, Software Defined Networks (SDN) is introduced to handle the networking between the network instances programmatically. The next-generation networks can be controlled through programmable interfaces [4–7].

Currently, SDN and NFV are considered building blocks for the development of 5G. Specifically, NFV Management and Orchestration (NFV-MANO) platform from European Telecommunications Standards Institution (ETSI) is most popular among the developers. MANO consists of an orchestrator that can orchestrate resources over virtual infrastructure managers (VIMs) using design configurations. These configurations are the most crucial parts that define the network's behavior [8–10]. Also, Open-source MANO (OSM) is one of the open-source platforms developed based on MANO standards. OSM consists of an orchestrator that accepts the configurations from the top and orchestrates the resources based on these configurations. OSM is a generic platform where different types of VIMs and SDN controllers can be enabled. It inherently provides Open-VIM that serves as an infrastructure provider. Also, depending upon user choice, OpenStack can be integrated.

Similarly, Central Office Re-architected as a Datacentre (CORD) is a development by the Open Networking Foundation (ONF) for the orchestration of VNF (Virtual Network Functions) over commodity hardware. It consists of everything as a service operating system (XOS) that serves as an orchestrator to deploy various applications through predefined configurations. XOS is integrated with OpenStack as an infrastructure provider and Open Networking Operating System (ONOS) to manage networking between different components. Hence, different platforms enable programmability for the orchestration and control of networks. However, every separate application's unique requirements require different configurations, and manual generation by experts is prone to errors and takes time [11]. Also, the resource allocation using manual approaches is not optimal, and it affects the energy efficiency. The problem with the existing approaches is that the demand cannot be estimated, and maximum resource allocation for smooth service provisioning is necessary. So, automated resource allocation with scaling approaches for optimized resource allocation must achieve efficiency in terms of energy and OPEX. Hence, it is required to generate the configurations automatically while considering the variety of requirements from applications and optimal provisioning. The second issue with the existing platforms is handling unexpected changes in system configurations due to the existence of dynamic traffic patterns in the network. This manuscript aims to provide a solution that enables automatic configurations for different platforms. It automatically controls and updates the system state based on real-time requirements resulted due to the immense amount of traffic. Hence, it provides a platform that manages resources optimally and enables energy efficiency for virtualized networks [12].

IBN (Intent-Based Networking) is one of the key solutions focused by the IETF standards (Internet Engineering Task Force) to automate the network configurations and operations. It is self-configuring, self-healing and self-organizing [13,14]. Hence, in this work a novel Intent-based framework for automatic orchestration of network resources over heterogeneous platforms is incorporated. It uses ML capabilities to proactively update the system state in real-time, eventually achieving resource optimization. A testbed is developed, which enables the automatic configuration of slices over M-CORD, OSM platforms, and ML model is integrated with it to scale the slice resources proactively. Fig. 1 provides an abstracted view of the framework where user provides the intents as high-level requirements. The intents are then translated to the policy configuration for different orchestration platforms depending upon user choice. After that, the orchestration framework keeps the system in the required state by continuously monitoring and updating system configurations. The key contributions of the platform defined in this manuscript are as follows:

(1) Development and design of a generic Intent-based system that enables the automatic configuration of multiple platforms through a single application.
(2) Application of ML for proactive scaling of network slice instances in case of a change in demand while ensuring resource optimization.

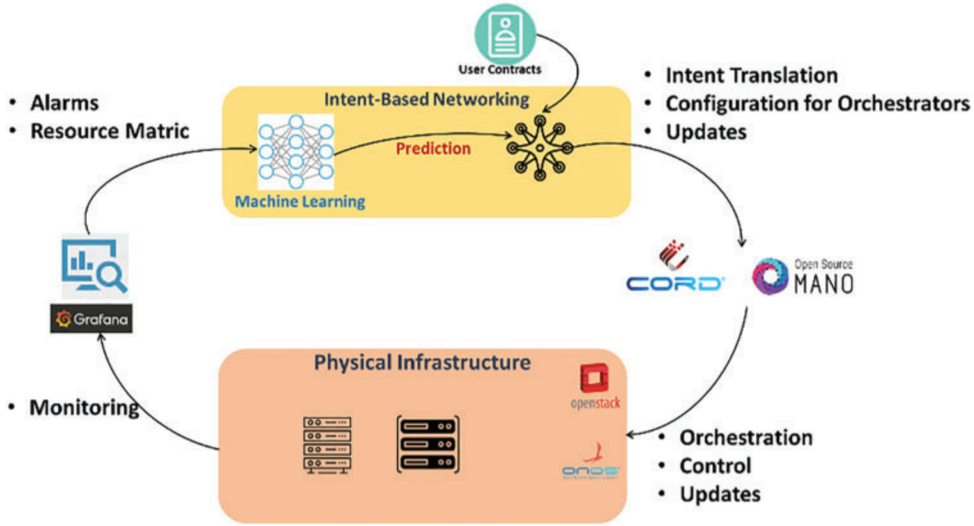


Figure 1: Explains a closed-loop IBN system where users provide high-level contracts translated for different platforms and control the proactive updates through monitoring and prediction

The rest of the paper is organized as follows. Section 2 of this manuscript briefs about the literature review. Section 3 includes system architecture and working, and Section 4 contains the explanation of system components. Section 5 discusses results and Section 6 concludes the work.

2 Related Work

The emergence of 5G paved the way for the Internet of Everything, which introduced many new services to the domain. Many research groups, up to some extent, provide a solution to seamless service provisioning. The development of 5G includes many aspects, i.e., RAN (Radio Access Networks), slicing, virtualization orchestration, virtual Network function, architectures, automation, platforms, and infrastructures. Therefore, various standard procedures have been conveyed to developers by the standard organizations, projects, and research groups, including 3GPP (3rd generation Partnership Project), ONF, ETSI (European Telecommunication Institution), 5GPPP, IETF, and many others. Standard organizations and industries, such as Huawei, CISCO, Radysis, and many others propose their innovative solutions. Similarly, open-source solutions, including OAI, CORD, OSM, MANO, ONAP, 5G Pagoda, and many others, are available to further support research works in 5G [4–7].

ETSI, with its NFV-MANO approach, provides an architecture to orchestrate networks. At the same time, ONF provides open-source solutions to achieve the 5G network by initiating CORD and ONOS projects. NGPaaS (Next-Generation Platform as a Service) is one of the initiatives of the 5GPPP project, which aimed at simplifying network configuration and abstraction of configuration procedure for cross-domain orchestration. Other projects of 5GPPP define various solutions to cater to service orchestration and automatic control of network management. It explains PaaS (Platform as a Service) from the top, which serves different underlying platforms as a configurator regardless of their design and provides a single interface to configure multiple platforms [8–12].

IETF standardizes IBN in collaboration with Huawei, CISCO and APSTRA. The goal of IBN is to design a closed-loop architecture, where it abstracts the network so that the users only need to provide high-level ("what" is required) information. The system is responsible for determining ("How" will be achieved) to perform low-level configuration and orchestration. It determines six golden principles of IBN: it must have SSOT (Single Source of Truth), i.e., a single policy must be well defined and determine a well-defined task. It must be one-touch, not one-shot, i.e., it requires only one and simple input but configuring that input depends on the environment's abstraction layers. Also, the system must be closed-loop; that is, it must automatically handle all underlying actions without experts' requirements and must be self-healing and updateable. Similarly, it must be explainable and must have learning capabilities [13]. Finally, the most important is an abstraction: it must cater to different underlying platforms and not be dependent.

Huawei Intent-Driven Networking follows the same approach and Huawei has defined Intent-Driven Datacenter, one of the prestigious one-touch solutions to control multi-domain cloud environments [14]. Similarly, APSTRA's AOS (Apstra Operating System) is one solution to control cloud leaf-spine across domains automatically, and they have defined its edge core cloud connectivity using AOS and CORD [15]. Finally, Cisco, one of the leading networking innovators, also equipped its project with IBN capabilities [16]. The zero-touch project of ETSI has gone far from all the approaches eliminating the need for all experts from administrators and defining fully automated networks [17]. The development and design of such networks is not a very easy task; it requires tons of innovation and knowledge of almost all networking domains. This paper focused on developing a unique solution that follows the IBN to automate network slice orchestration and assurance of slice availability by enabling autonomous update and recovery by using cognitive intelligence. Also, it focused on developing IBN on top of SDN and NFV platforms.

Many of the research works have focused on developing machine learning models for the automatic scaling of network instances in the cloud. Time-series forecasting has been used in different research works to predict scaling decisions [18–20]. The cloud resource scaling for achieving energy efficiency is explained in [21]. Also, the autoscaling of cloud resources for attaining energy efficiency was considered in our previous research work [21], where optimization of cloud resources was achieved using an auto-scaling mechanism. The drawback of the auto-scaling system is that it first experiences the environment and then scales the system to achieve energy efficiency.

In contrast, in our approach, we used ML to maintain cloud resources proactively and efficiently manage them. Different research papers have used the ML models for various applications as VM migration strategy has been designed based on Time Series Forecasting.

Similarly, the cloud workloads evaluation is performed for predictive resource scaling. It explains the behavior of different ML models and how predictions can help scaling cloud resources. In addition to that, PRACTICE: Robust Prediction of Datacentre Time Series work also explained the Machine Learning model's training for automatic handling of cloud infrastructures. Many other prominent works have been developed over the recent years to apply machine learning to predict cloud resource utilization and update the system configuration accordingly. So, in this manuscript, we will predict the VNF instance workload and per the predicted results, we will scale the VNF instances. The uniqueness of our work is the use case that the implementation of ML will achieve.

3 System Architecture and Working

Unline traditional network approach, IBN is a centralized platform that continuously monitors and handles all the failures. The architecture of IBN consists of a top-to-bottom layer approach with a closed-loop design. The top administrative layer controls the underlying layers and requires simple high-level interactions from the operator. In the architecture illustrated in Fig. 2, the first layer is composed of an intent-based application, which from the top accepts a high-level system requirement. Intent determines "what" is required, through which the intent translation engine determines the policy for "how" the intents will be orchestrated on the physical layer. IBN application is composed of two engines: the translation engine that accepts high-level information as intents and generates well-defined configurations that determine "how" to orchestrate the requirements to the lower layers. Additionally, the assurance and update engine, which is based on continuous monitoring feedback of the orchestrated resources, determines optimal resource scaling while it assures service availability and optimal resource provisioning. Fig. 2 illustrates the architecture of an automated closed-loop IBN-based generic platform.

The second layer consists of orchestration and management platforms. This testbed incorporates M-CORD and OSM platforms as VNF orchestrators with different design and control mechanisms. Also, the access network is controlled using Flex-RAN, which is used for orchestrating RAN resources.

M-CORD has an XOS orchestrator that accepts detailed configurations in the form of TOSCA and orchestrates them to the physical layers using ONOS and OpenStack. XOS consists of XOS-CORE that receives the TOSCA pushed using TOSCA-Rest API as shown in Fig. 2 [21]. The XOS-CORE pushes the accepted configuration to XOS-DB, which instantiates Xproto models for each of the configurations. There are two types of Xproto models, the VNFs modelling and SDN control plane application models, which are generated according to the information pushed from XOS-CORE. Basically, under XOS, everything is deployed as a service, so there are two types of services and service models; the first is VNF services representing certain virtual network functions. Secondly, the SDN control applications represent network control plane functions. It uses the XOS-Synchronizers that receive the xproto-models from XOS-DB to orchestrate them over the physical layer. The synchronizers are responsible for keeping the system consistent with the information that exists in XOS-DB as shown in Fig. 2 [22].

OSM is an implementation of ETSI-MANO architecture for VNF orchestration. OSM from top accepts deployment requirements in the form of three YAML templates or the same information through JSON-based REST interface, i.e., VNFD (virtual Network Function Descriptors), NSD (Network Service Descriptors) and NST (Network Slice Templates). VNFD determines the virtual network function configuration in terms of resources, interfacing, and networking. NSD contains the information of how VNFs will be interconnected to provision a service. NST is the slicing template through which OSM determines the slicing and chaining of services. The templates are translated as catalogs, which are translated into underlying platforms using different plugins. Unlike CORD, OSM is a generic platform, and it enables the use of different VIMs and SDN controllers through plugins. It requires the administrators to set up the VIM environment of their choice, e.g., OpenStack and OpenVIM, to orchestrate VNFs. It also provides freedom of using different SDN controllers like ONOS, OpenDaylight, etc. Fig. 2 illustrates the integration of OSM and plugin information.

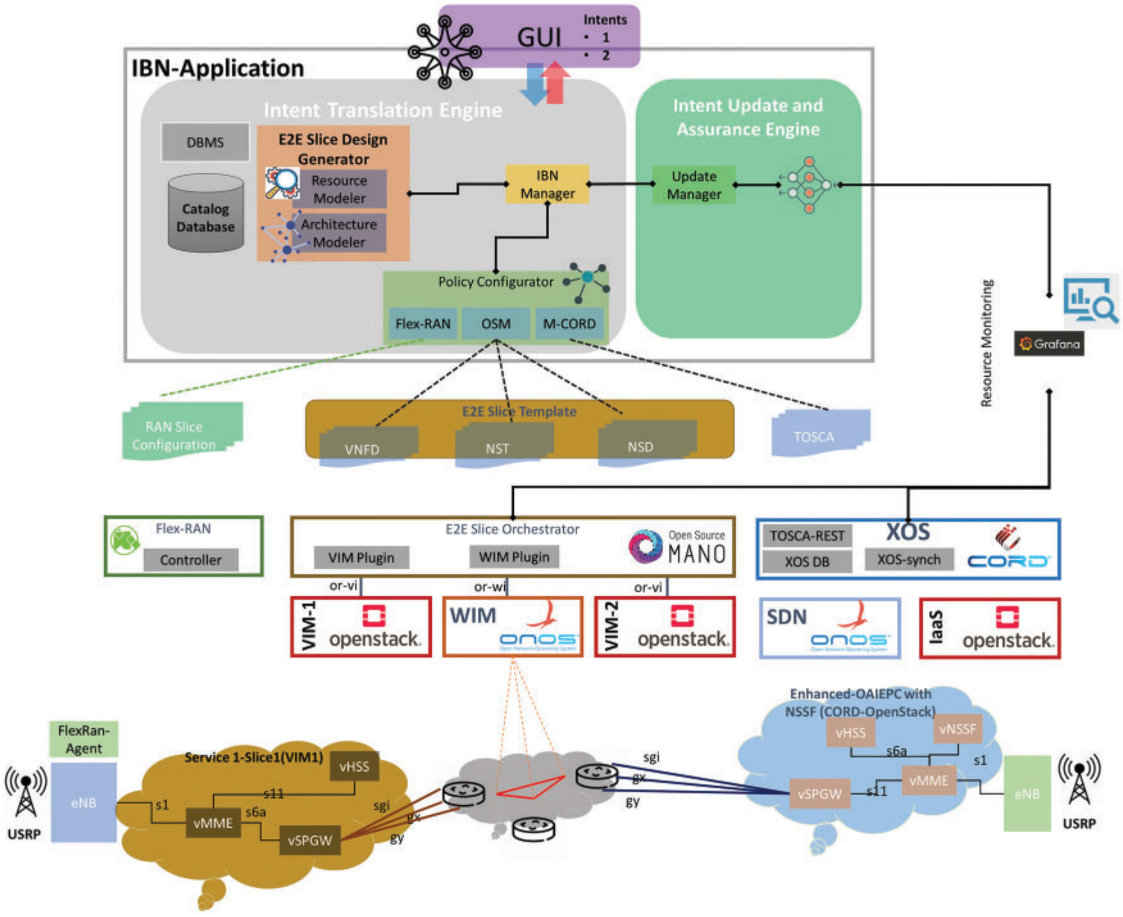


Figure 2: Explains a closed-loop IBN system where users provide high-level contracts translated for different platforms and control the proactive updates through monitoring and prediction

The third layer is a physical layer consisting of the different underlying controllers, VIMs and SDN controllers depending on the management platform's choice. CORD testbed use case consists of multiple VNFs provisioned under OpenStack and ONOS. The OpenStack synchronizer in XOS controls the OpenStack cloud controller module. It uses the Xproto Model for each

VNF provision on the OpenStack cloud. Similarly, ONOS control application synchronizers use ONOS to control the networking between VNFs running on OpenStack cloud environment using the ML2 plugin [23]. Here the ONOS VTN (virtual Tenant Networking) application is worth mentioning because it defines how different network nodes will communicate and their relation by enabling SFC (service function chains) for different slice tenants. So, it provides basics for how multiple slices will be orchestrated. We can control the behavior of interconnecting VNFs, and it enables different service QoS based on opted stitching between VNFs, and resources orchestrated for each of the VNFs. Besides, the ONOS vRouter application routes the traffic outside the underlying infrastructure. The physical layer consists of multiple virtual network functions with multiple instances to serve different slices [24].

The overall architecture is illustrated in Fig. 2, in which the system starts from inserting intents, determining specific high-level information of "what" to be orchestrated. Intent Manager sends a request to Policy Store to provide specific information related to the contract on receiving intents. After that, resources for the information are determined using resource management services. After, policy configurators for different platforms and domains are used to generate the configurations in the underlying management platforms' respective format. The configurations are then provided to the orchestrator to orchestrate them to the physical layer. In the case of CORD platform, a unique network slice selection function is introduced to select the intended service slice while in operation. According to 3GPP, there are three types of network slicing when we opt for LTE architecture; without RAN slicing, with RAN slicing, and tenant-based disaggregated RAN and Core network slicing. In this testbed, we have considered both the RAN slicing case, and without the RAN slicing. The NSSF is introduced in the case when we do not have any RAN slicing. So, on the management layer, the NSSF keeps the core-network slicing information updated with XOS-DB using its synchronizer. In real-time, it can select newly generated slices or respond according to the changes in configurations as shown in Fig. 6. In our OSM test bed we have utilized FlexRAN controller for the management creation and selection of RAN slices and core slices are also selected in response to specific slice demands. As it is a closed-loop architecture, the second part is to continuously monitor the orchestrated resources and update the system for optimal resource configuration. Hence, perceiving the current state of the system and projecting the system status to the NN model, the future state of the system is determined, which proactively updates the system configurations to assure service provisioning [25–33].

4 System Component and Working

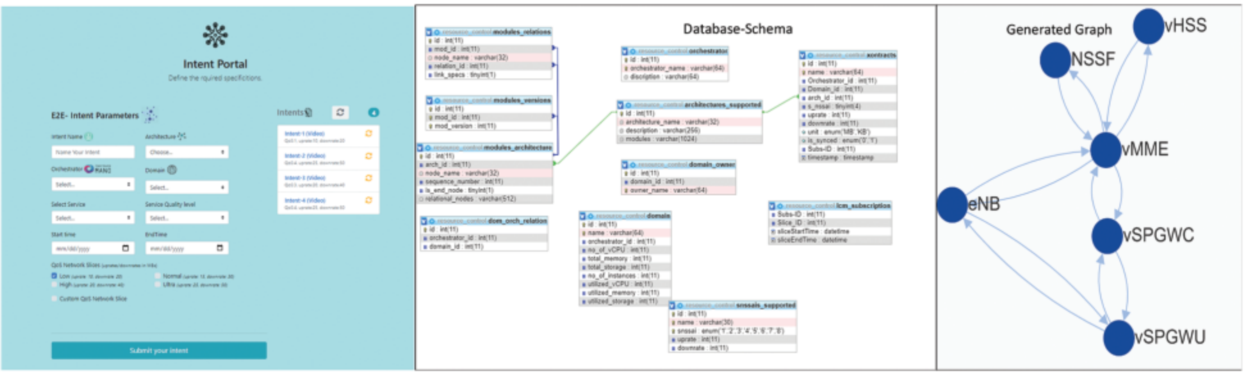
The system consists of three layers: the intent application layer, the management and orchestration layer and the physical layer. Intent application layer is the significant contribution of this architecture, and it is the main layer of the architecture. The management layer majorly consists of M-CORD, but monitoring procedures are additionally added for dynamic update management, and NSSF synchronizer for the specific purpose of keeping updated information for selecting slices. At the physical layer, OAI is majorly used with few enhancements: firstly, NSSF is included to our scenario, which is responsible for selecting proper slices, whereas OAI-SIM and OAI-EPC are converted to services under M-CORD. The proposed architecture is explained as follows:

4.1 Intent Translation Engine

It is the policy engine of an intent-based application and is used to generate the configuration using a well-defined policy mechanism. The working of each of the components of Intent Policy Engine is as follows:

4.1.1 GUI

Intent and Contracts are interchangeable terminologies used in this manuscript for the high-level input of the IBN-application. The first component consists of a specified definition for orchestrating a service on multiple domains using various platforms. An intent is determined using various fields: 1) the S-NSSAI (Single Network Slice Selection Assistance information) is included in determining which kind of service needs to be orchestrated. S-NSSAI is based on the 3GPP concept, where different domains of network services are divided into groups and each S-NSSAI determines a single type of service, 2) many operators and developers have defined their own architectures for the network, archID is included as an input field to determine specific architecture, 3) parameters include the determining of the underlying platform and domain VIM to be selected, 4) it consists of QoS fields to define specific requirements for the specified contract and instantiate time and subscription end time to retrieve the resources. The front-end used to insert contracts is depicted in Fig. 3, which shows the initialization of four different contracts and their status.



resource_control modules_relations
id : int(11)
mod_id : int(11)
node_name : varchar(32)
relation_id : int(11)
link_specs : tinyint(1)

resource_control modules_versions
id : int(11)
mod_id : int(11)
mod_version : int(11)

resource_control modules_architecture
id : int(11)
arch_id : int(11)
node_name : varchar(32)
sequence_number : int(11)
is_end_node : tinyint(1)
relational_nodes : varchar(512)

resource_control dom_orch_relation
id : int(11)
orchestrator_id : int(11)
domain_id : int(11)

resource_control orchestrator
id : int(11)
orchestrator_name : varchar(64)
discription : varchar(64)

resource_control architectures_supported
id : int(11)
architecture_name : varchar(32)
description : varchar(256)
modules : varchar(1024)

resource_control domain_owner
id : int(11)
domain_id : int(11)
owner_name : varchar(64)

resource_control domain
id : int(11)
name : varchar(64)
orchestrator_id : int(11)
no_of_vCPU : int(11)
total_memory : int(11)
total_storage : int(11)
no_of_instances : int(11)
utilized_vCPU : int(11)
utilized_memory : int(11)
utilized_storage : int(11)

resource_control xontracts
id : int(11)
name : varchar(64)
Orchestrator_id : int(11)
Domain_id : int(11)
arch_id : int(11)
s_nssai : tinyint(4)
uprate : int(11)
downrate : int(11)
unit : enum('MB','KB')
is_synced : enum('0','1')
Subs-ID : int(11)
timestamp : timestamp

resource_control lcm_subscription
Subs-ID : int(11)
Slice_ID : int(11)
sliceStartTime : datetime
sliceEndTime : datetime

resource_control snssais_supported
id : int(11)
name : varchar(30)
snssai : enum('1','2','3','4','5','6','7','8')
uprate : int(11)
downrate : int(11)

Figure 3: Front end of IBN application showing the inputs required from the user. It also shows the instantiation of different intents. It illustrates an example LTE architecture model generated using the catalog repository of IBN-Application and its representation on the front end

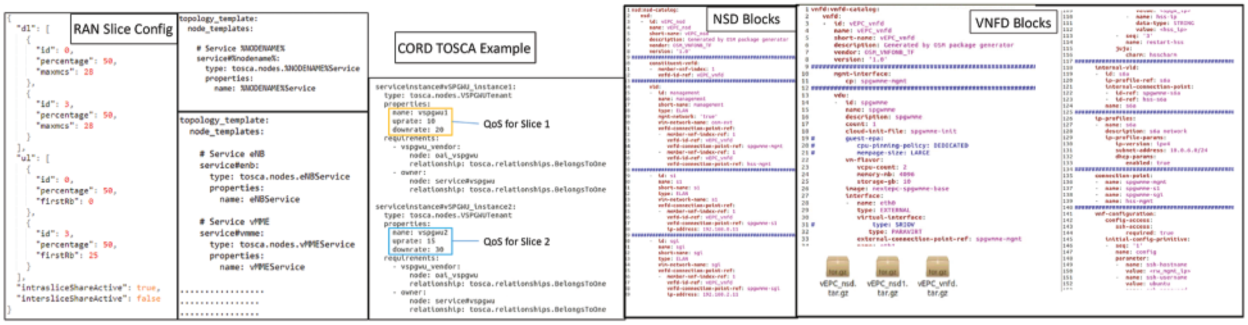
The design and flow of intent GUI is shown in Fig. 3. It shows how a user can navigate the GUI by selecting different options. The IETF standard explains that every input from the front-end must be well-defined, and it must represent a conflict-free output to be translated by an intent-based application. Besides, the GUI offers only drop-down options to be selected while considering the availability of the type of functionality. As IBN is a closed-loop system, users are limited to request the intent within the domain of underlying platforms. Hence, the GUI is designed such that every input from the front end can be translated to a specific intent by the user, so that it can be implemented by the underlying platforms.

4.1.2 Catalog Repository

The contract information is used to generate configuration through intent translation. The system consists of a well-defined Catalog Repository that stores required information for orchestrating each intent generated from the front-end. Catalog repository determines the relation among different VNF services, and it has a well-scripted policy that generates a graph that consists of complete end-to-end slice information. It consists of the following tables: xcontracts, architectures supported, modules-architecture and modules-relation table for generating architecture models. The xcontracts table stores the list of contracts inserted through a GUI interface by the user/operator. The architectures supported table relates a unique key with each architecture name. The modules-architecture table consists of modules and their dependent nodes information; for example, SPGWC service is dependent on vMME and SPGWU functionalities. The modules-relation table contains the information associated with link specifications from one module to the other. A service Graph generated using the catalog repository and DB-Table are illustrated in Fig. 3.

4.1.3 *Policy Configurator*

The graph generated can be considered as vNFFG (VNF Forwarding Graph). Fig. 4 represents the graph generated by the IBN application for M-CORD service orchestration intent. The resource modeler module, which is available in the e2e design generator component, determines the resources of each contract. The resources are modeled according to service requirements and the number of available resources in the underlying domain/platform. There are two types of resource modelers: the RAN resource modeler and the VNF resource models. The respective Policy Configurator uses each kind of resource model that generates a complete orchestration configuration for the underlying platform. The M-CORD's case intents are translated to the TOSCA YAML and in case of OSM they are translated to NSD, VNFD, and NST. Similarly, RAN configurations are generated in JSON format to be provided to FlexRAN. An overview of each type of configuration is illustrated in Fig. 4.



```
{
  "dl": [
    {
      "id": 0,
      "percentage": 50,
      "maxmcs": 28
    },
    {
      "id": 3,
      "percentage": 50,
      "maxmcs": 28
    }
  ],
  "ul": [
    {
      "id": 0,
      "percentage": 50,
      "firstRb": 0
    },
    {
      "id": 3,
      "percentage": 50,
      "firstRb": 25
    }
  ],
  "intrasliceShareActive": true,
  "intersliceShareActive": false
}
```

```
topology_template:
  node_templates:

    # Service %NODENAME%
    service#%nodename%:
      type: tosca.nodes.%NODENAME%Service
      properties:
        name: %NODENAME%Service
```

```
topology_template:
  node_templates:

    # Service eNB
    service#enb:
      type: tosca.nodes.eNBService
      properties:
        name: eNBService

    # Service vMME
    service#vmme:
      type: tosca.nodes.vMMEService
      properties:
        name: vMMEService

................
................
................
```

```
serviceinstance#vSPGWU_instance1:
  type: tosca.nodes.VSPGWUTenant
  properties:
    name: vspgwu1
    uprate: 10
    downrate: 20
  requirements:
    - vspgwu_vendor:
        node: oai_vspgwu
        relationship: tosca.relationships.BelongsToOne
    - owner:
        node: service#vspgwu
        relationship: tosca.relationships.BelongsToOne

serviceinstance#vSPGWU_instance2:
  type: tosca.nodes.VSPGWUTenant
  properties:
    name: vspgwu2
    uprate: 15
    downrate: 30
  requirements:
    - vspgwu_vendor:
        node: oai_vspgwu
        relationship: tosca.relationships.BelongsToOne
    - owner:
        node: service#vspgwu
        relationship: tosca.relationships.BelongsToOne
```

```
nsd:nsd-catalog:
  nsd:
  - id: vEPC_nsd
    name: vEPC_nsd
    short-name: vEPC_nsd
    description: Generated by OSM package generator
    vendor: OSM_VNFONB_TF
    version: '1.0'
##################################################
    constituent-vnfd:
    - member-vnf-index: 1
      vnfd-id-ref: vEPC_vnfd
##################################################
    vld:
    - id: management
      name: management
      short-name: management
      type: ELAN
      mgmt-network: 'true'
      vim-network-name: osm-ext
      vnfd-connection-point-ref:
      - member-vnf-index-ref: 1
        vnfd-id-ref: vEPC_vnfd
        vnfd-connection-point-ref: spgwnne-mgmt
      - member-vnf-index-ref: 1
        vnfd-id-ref: vEPC_vnfd
        vnfd-connection-point-ref: hss-mgmt
##################################################
    - id: s1
      name: s1
      short-name: s1
      type: ELAN
      vim-network-name: s1
      vnfd-connection-point-ref:
      - member-vnf-index-ref: 1
        vnfd-id-ref: vEPC_vnfd
        vnfd-connection-point-ref: spgwnne-s1
        ip-address: 192.168.0.11
##################################################
    - id: sgi
      name: sgi
      short-name: sgi
      type: ELAN
      vim-network-name: sgi
      vnfd-connection-point-ref:
      - member-vnf-index-ref: 1
        vnfd-id-ref: vEPC_vnfd
        vnfd-connection-point-ref: spgwnne-sgi
        ip-address: 192.168.2.11
```

```
vnfd:vnfd-catalog:
  vnfd:
  - id: vEPC_vnfd
    name: vEPC_vnfd
    short-name: vEPC_vnfd
    description: Generated by OSM package generator
    vendor: OSM_VNFONB_TF
    version: '1.0'
##################################################
    mgmt-interface:
      cp: spgwnne-mgmt
##################################################
    vdu:
    - id: spgwnne
      name: spgwnne
      description: spgwnne
      count: 1
      cloud-init-file: spgwnne-init
#     guest-epa:
#       cpu-pinning-policy: DEDICATED
#       mempage-size: LARGE
      vm-flavor:
        vcpu-count: 2
        memory-mb: 4096
        storage-gb: 10
      image: nextepc-spgwnne-base
      interface:
      - name: eth0
        type: EXTERNAL
        virtual-interface:
#         type: SRIOV
          type: PARAVIRT
        external-connection-point-ref: spgwnne-mgmt
```

```
          - name: hss-ip
            data-type: STRING
            value: <hss_ip>
        - seq: '3'
          name: restart-hss
        juju:
          charm: hsscharm
##################################################
    internal-vld:
    - id: s6a
      ip-profile-ref: s6a
      internal-connection-point:
      - id-ref: spgwnne-s6a
      - id-ref: hss-s6a
      name: s6a
##################################################
    ip-profiles:
    - name: s6a
      description: s6a network
      ip-profile-params:
        ip-version: ipv4
        subnet-address: 10.0.6.0/24
        dhcp-params:
          enabled: true
##################################################
    connection-point:
    - name: spgwnne-mgmt
    - name: spgwnne-s1
    - name: spgwnne-sgi
    - name: hss-mgmt
##################################################
    vnf-configuration:
      config-access:
        ssh-access:
          required: true
      initial-config-primitive:
      - seq: '1'
        name: config
        parameter:
        - name: ssh-hostname
          value: <rw_mgmt_ip>
        - name: ssh-username
          value: ubuntu
```

Figure 4: The example of configurations generated for LTE architecture in the case of OSM and CORD also includes FlexRAN configurations

Fig. 5, Illustrates the follow-up requests inside the IBN application, starting from user intents to IBN manager, which request the e2e designer to receive the architecture model and resource model. After that, it requests the policy configurator for policy configuration, where the policy configurator further requests the orchestrators to instantiate the services on the platform.

4.2 *M-CORD-Based Testbed*

M-CORD is vastly supported by industry and is an ONF-based open-source standard platform for implementing next-generation mobile networks. It consists of an XOS orchestrator integrated with ONOS and OpenStack. XOS is a service-based orchestrator and services are of two types; one is a VNF service that represents a particular virtual network function such as vMME (virtual Mobility Management Entity), SPGWC/U (Service Provider Gateway Control Plane/User Plane), vHSS (virtual Home Subscription Server), whereas other type represents an SDN control application that can determine a certain networking functionality such as VTN, vRouter, etc. For this testbed M-CORD setup, the VNFs are developed as services under CORD. The VNFs used by this testbed are open-source development provided by OAI (OpenAirInterface). The OAI-SIM (OAI-Simulator) is used, which contains eNodeB with the simulated user equipment. The OAI-SIM supports up to 30 simulated user equipment. We also used an OAI-EPC (OAI-Evolved Packet Core) that consists of virtual implementation of core network functions including vMME, SPGWC, SPGWU and vHSS. A very important part of the testbed is enhancing the OAI-EPC using NSSF (Network Slice Selection Function). As one of the goals is slice orchestration, the slice information is inputted to the system from the top in the form of intents. So, it is essential to determine how a slice should be allocated to a user to guarantee the service in real-time. So, we have introduced NSSF as shown in Fig. 6, which has two parts; one is NSSF synchronizer, and other is a VNF instance running as a service under M-CORD. The synchronizer continuously monitors the updates in XOS-DB and reflects them to the database present in NSSF VNF. Hence, whenever a new intent or update of scaling of slice decision is made, NSSF feeds that information in its database and selects new slicing according to the updated information.

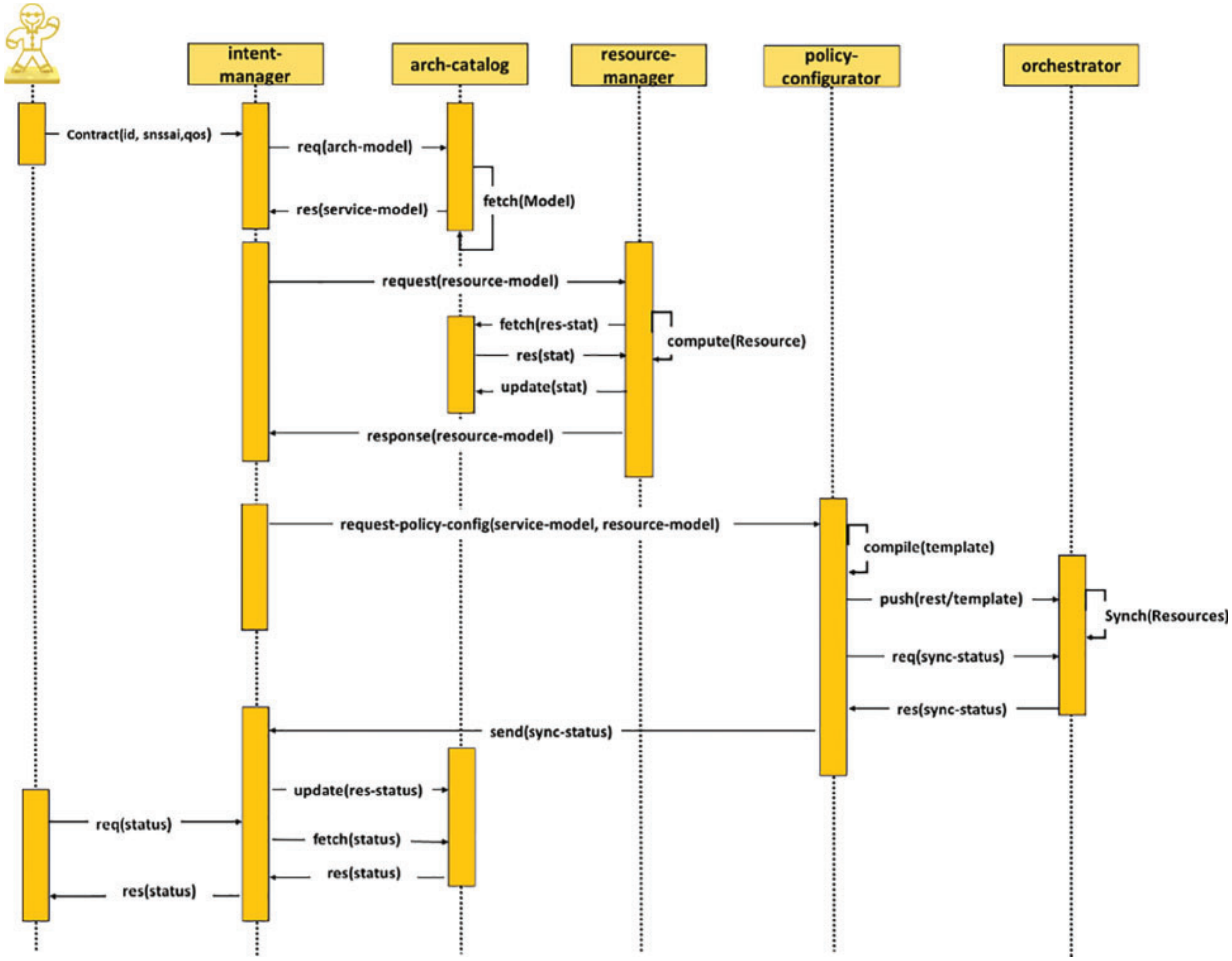
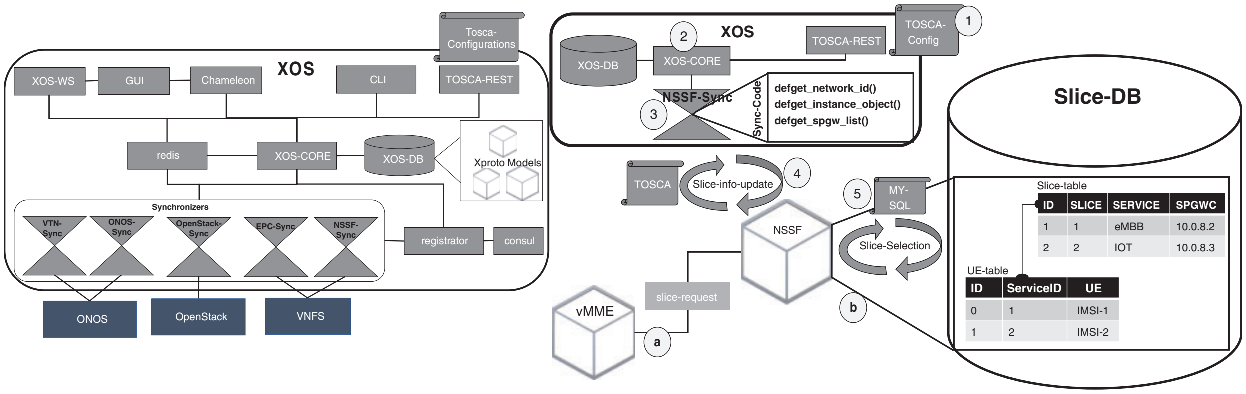


Figure 5: Flow of information for intent translation engine

The testbed under M-CORD UE connects to the CORE network using eNodeB, which forwards the connection request to vMME to support NSSF based slice selection. So, vMME sends a request to vHSSS to check user subscription information. Upon the confirmation of subscription, it requests NSSF to select a proper core network instance. After the selection of slice instance, it requests SPGWC to allocate data plane according to the slicing information: an example architecture with multiple slices is shown in Fig. 6, whereas Fig. 2 represents the deployment on the physical platforms [33].



Slice-table

ID	SLICE	SERVICE	SPGWC
1	1	eMBB	10.0.8.2
2	2	IOT	10.0.8.3

UE-table

ID	ServiceID	UE
0	1	IMSI-1
1	2	IMSI-2

Figure 6: XOS internals and NSSF working inside M-CORD platform

4.3 OSM-and-FlexRAN-Based Testbed

OSM is an open-source VNF orchestrator based on ETSI-MANO architecture, whereas FlexRAN is open-source software provided by Eurocom Mosaic 5G project. Fig. 2 illustrates the OSM-based testbed under IBN application. The testbed is enabled with multisite VIM environments, where at each site, we have deployed our EPC. Multiple EPC instances can be deployed to depict multiple slices. For the slicing of RAN, we have integrated FlexRAN. In addition, we provide the information of core network slices to be stitched with access to the FlexRAN controller through the policy configurator using the e2e template.

In contrast, the M-Cord-based testbed NSSF is used to select the core-network slice. According to 3GPP standards [34], the network slicing and slice selection procedure differ when RAN is sliced and when RAN is not sliced. In our case, RAN is sliced; hence tenant-based RAN and core network slice stitching is used.

4.4 Monitoring Tool

In the CORD testbed, Nagios is used for VNF monitoring, as it is one of the components of OpenStack distribution available in CORD and is programmable. It is used in OpenStack to collect metering information from the ceilometer of OpenStack. Ceilometer consist of log information about CPU, Memory, storage, and network utilization of each of the VNFs. It is used to collect monitoring information of VNF. We also use sFlow with ONOS to collect the data plane bandwidth monitoring information. Through the information from both, Nagios and sFlow application, the updates are decided using alarms and ML-based predictions. Alarms are used to determine the threshold values to ping the ML model to check the resource status in the future. Like CORD, OSM also has a built-in VNF monitoring tool that collects the telemetry logs from

OpenStack ceilometer; an overview of monitoring is illustrated in Fig. 2. After collecting the logs, we can visualize the logs and alarms using the Grafana tool [34].

4.5 Intent Assurance and Update Engine

The purpose of continuous monitoring of resources is to react according to the system state to ensure stable service provision. This section explains the detection of failure and the techniques followed to update system configurations. We use a reactive approach that introduces alarms, which are activated directly upon receiving max utilization according to preset value for any resource. It requests an update manager with the information of the resource status. A proactive approach is introduced using an ML model that continuously receives the current resource status and updates the system according to future state requirements. We use an open-source cloud data center data, provided by GWA-T-13 MATERNA, a German-based distributed cloud dataset [35]. It contains the dataset of three different traces; each trace has 520,527 and 547 VMs, where data of each VM. It consists of multiple parameters, including CPU utilization, memory utilization, disk writes and network throughput. The complete detail of each parameter is shown in Tab. 1.

Table 1: GWA-T-13 Materna dataset description

Index	Name	Description
0	Timestamp	Number of milliseconds
1	CPU cores	Number of CPU cores
2	CPU capacity	Number of cores X speed per core
3	CPU usage	In MHZ
4	CPU usage	In percentage
5	Memory provisioned	Total memory of VM in KB
6	Memory usage	Actively used Memory in KB
7	Memory usage	Percentage
8	Disk write	In KB/s
9	Disk size	In GB
10	Network in throughput	In KB/s
11	Network out throughput	In KB/s

As our input data set has complex arrangements, a high dimensional function is required. We have tried two approaches: RNN (Recurrent Neural Network) and second is NN-MLP (Neural Network-Multilayer Perceptron) NN-MLP performed better so we have included the results of NN-MLP in the Tab. 2 and results for RNN are shown in Tab. 3. The results are shown with different input parameters from the dataset. From the results we can see that the best-trained model we got is only the prediction CPU utilization from current utilization of CPU. It is because if you get a chance to visit the dataset mostly the trend of CPU utilization stays on average and there are few spikes. As the operating system keeps the CPU utilization at max to execute the jobs quickly. The other unique thing that came out was that as we have increased the number of input parameters for the model, the accuracy was decreased. When we investigated this trend, the only issue is that the correlation of each of the parameters with each other is almost non-existent at least in this dataset. So, for right in our testbed, we incorporated the NN-MLP model having input features set to all.

Table 2: Training and testing accuracy of the NN-MLP with 3 types of input data

	NN-MLP	
Input data	Train accuracy	Test accuracy
cpu	0.914	0.890
cpu-mem	0.852	0.837
all	0.813	0.810

Table 3: Training and testing accuracy of the RNN with 3 types of input data

	RNN	
Input data	Train accuracy	Test accuracy
cpu	0.821	0.792
cpu-mem	0.764	0.716
all	0.702	0.650

Although the training and testing results are significant, our future goal is to find out more efficient and effective ways to train models and find a dataset of a real-time deployed VNF in a telecommunication environment. For right now, for lab-test purposes, the model shows a significant performance.

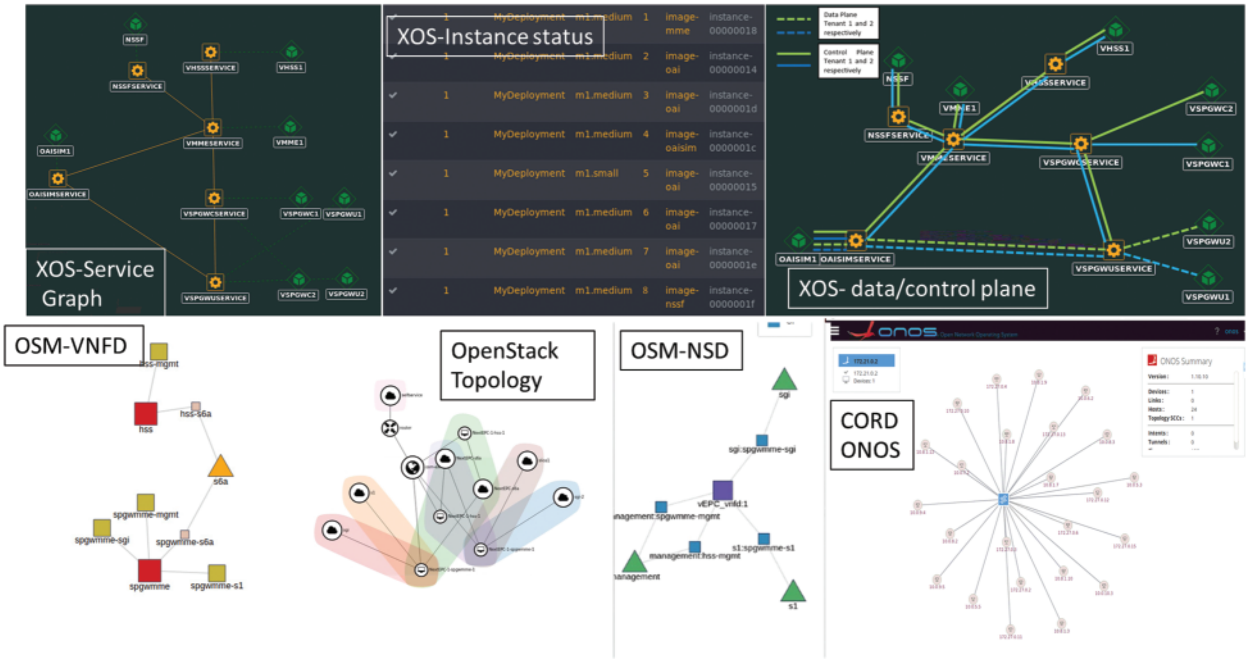
5 Results

As this system addresses multi-dimensional challenges, it contains multi-dimensional results. The first dimension includes the orchestration of services across multiple platforms and domains to demonstrate the generic nature of IBN. Fig. 7 illustrates XOS and OSM's status by depicting XOS service graph and OSM network service descriptor graph. Also, it includes the visualization instantiation of EPC instances at OpenStack in different cases. CORD-ONOS is also visualized in Fig. 7.

The IBN application is deployed in a separate machine and it has a web interface for inserting the contracts. Secondly, we used M-CORD 4.1 CiaB (Cord in a Box) version where all the M-CORD testbed resides in a single server. The testbed is like the designed architecture described in Fig. 2. The second dimension for the results is SLA (Service Level Agreement) assurance, whether the IBN-application assures the slices' availability at the intended bandwidth throughout the lifecycle. The test was performed using the following configuration. In the case of cord, five slices with different bandwidth were instantiated and an increase in OSM at each site single EPC was instantiated.

The iperf-test is performed, and the results are shown in Fig. 8. Results reflect the exact down-link capabilities allocated to each of the slices given from the top with very small downfalls. Similarly, Fig. 8 also shows the up-link consistency and fulfillment of contracts for each created slice. OAI-SIM is used instead of a real RAN or industry-standard equipment in the case of CORD, so the test was performed with meager bandwidth allocation to each slice. Also, the core purpose of the test results is not to test the industrial standard bandwidth achievement, but it is to check whether the proposed system can orchestrate the resources under the requested contracts.

Also, as M-CORD is an industry-standard platform, its capabilities are out of the scope of this paper. Many industrial demonstrations for the usefulness and Bandwidth and performance testing of M-CORD environments can be referenced from RadiSys [36–40].



✓	1	MyDeployment	m1.medium	1	image-mme	instance-00000018
✓	1	MyDeployment	m1.medium	2	image-oai	instance-00000014
✓	1	MyDeployment	m1.medium	3	image-oai	instance-0000001d
✓	1	MyDeployment	m1.medium	4	image-oaisim	instance-0000001c
✓	1	MyDeployment	m1.small	5	image-oai	instance-00000015
✓	1	MyDeployment	m1.medium	6	image-oai	instance-00000017
✓	1	MyDeployment	m1.medium	7	image-oai	instance-0000001e
✓	1	MyDeployment	m1.medium	8	image-nssf	instance-0000001f

Figure 7: It firstly illustrates XOS-Service graph generated instances from XOS-GUI and shows the data/control plane flow for each of the two slices. Secondly, it shows the OSM VNFD and NSD graph; besides that, it shows OpenStack topology for OSM case and ONOS graph on CORD ONOS

Similarly, in OSM, the USRP devices were used as physical radio and are still not the industrial standard for RAN. Also, FlexRAN was used to slice the RAN, and results depict the orchestration of resources through OSM and FlexRAN for e2e slice creation through the IBN application.

The second part of the test is slice assurance; for example, if we have inserted a contract and the slice and resources were all consumed by the current user, more users must be served. The solution implemented using ML is scaling up resources on-demand, specially SPGWU service instances. Because of not having industry-standard equipment and extensive resources, our testbed has a limitation of resources so for performing a test, only one slice was instantiated, and it was allocated with 1 vCPU core and 2 GB of RAM to observe the update procedure and the traffic from 3 UE generated and they started streaming traffic and connection request one after other with time intervals. The test was performed to achieve the resource overload condition, or we can see whether the resources are updated on runtime or not. The overview of the result is depicted in Fig. 8, and X-Axis represents the noted interval with time and Y-Axis represents the values of no of Instances of SPGWU, No of User, CPU utilization, Predicted Utilization and Traffic Streamed. The results are shown by using the information of only one instance and the overall user equipment. So, we can see at stage 7 the number of instances of SPGWU increased. The user equipment also jumped from 2 to 3 and at the stage CPU utilization of the instance was 89, and predicted utilization in stage 6 was 78 and predicted CPU utilization at stage 7 is 97, so in between the noted time, it changed the number of SPGWU instances. In the future we can see for this instance the usage pattern goes down, which means UE 3 traffic was streamed through the newly generated SPGWU instance.

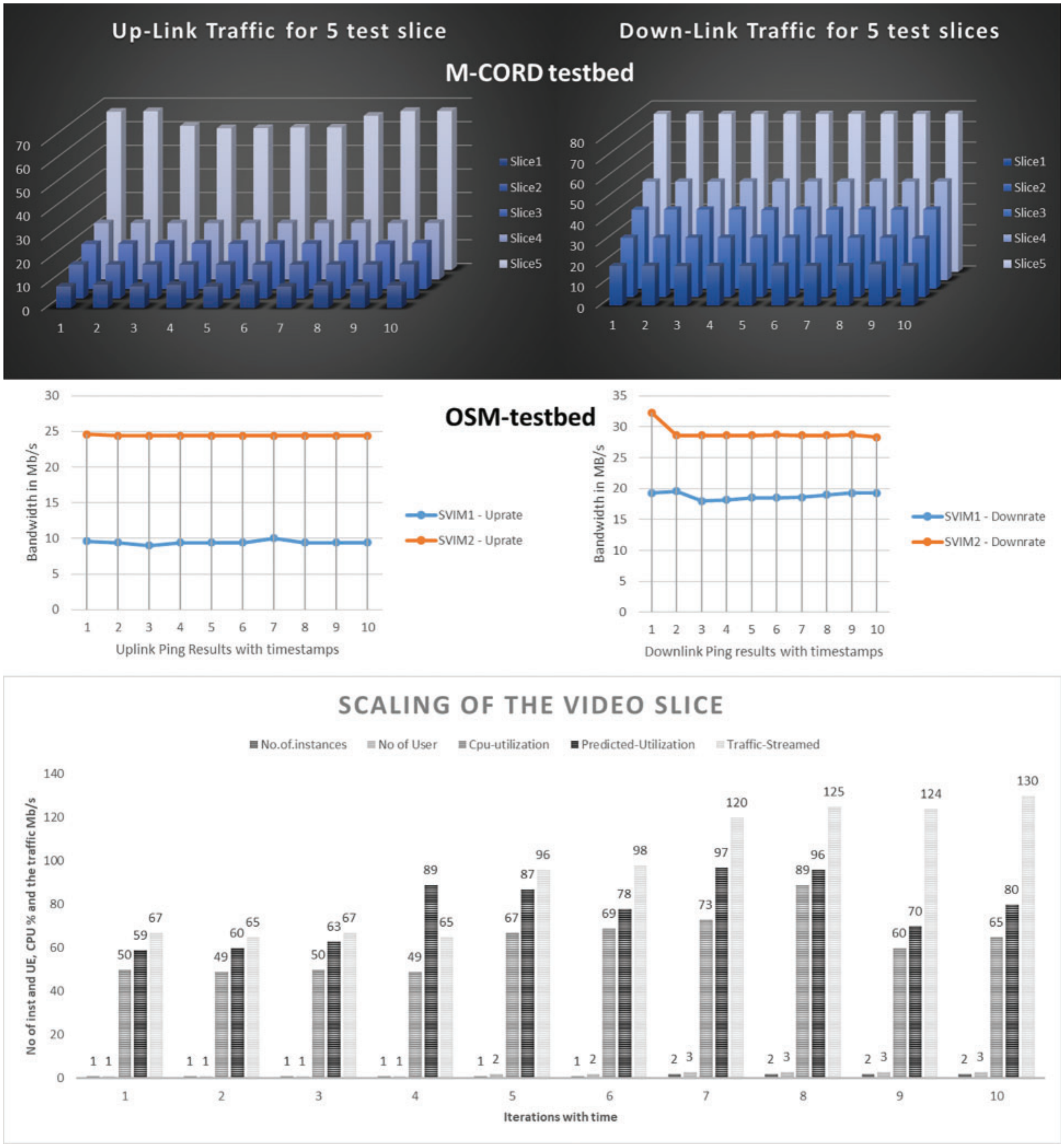


Figure 8: It illustrates the bandwidth results on both the CORD and OSM testbed. Also, depicts the scaling of VNF for a video slice case

6 Conclusion

The IBN-based platform has been developed, and it enables automation for addressing multi-dimensional issues of next-generation networks. The proposed IBN application from the top enables the simplification in the orchestration of the network and eliminates the need for specific experts to orchestrate the network. It also considers generalization for end-to-end orchestration as a single interface; it enables handling multiple platforms and multiple domains. It abstracts how to configure specific domains/platforms and enables users to achieve what is required by a high-level interface. The automation of orchestration of different platforms at different domains was addressed and AI-driven assurance of the provisioned services were considered. The AI-driven updates explain the effectiveness of AI on the different aspects of networks, including proactive updates and optimization. The different standard LTE architectures were discussed specifically, a different variation of OAI-LTE components was used. Most importantly the proposed work in the manuscript achieved and developed a special case of LTE network architecture by introducing NSSF inside one of its testbeds. IBN application also considered the management of different platforms, including FlexRAN, OSM and CORD. The shift of telco central office to cloud due to virtualization and enablement of programmability for orchestrating the network provided the freedom to enhance the networks without caring about specific hardware and in many ways. This manuscript benefited from that programmability and applied automation, generalization, and AI to achieve what was essential while designing the manuscript considered standard IBN architecture and used many of the standard open-source platforms.

Acknowledgement: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2017-0-01633) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF2016R1D1A1B01016322).

Funding Statement: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF2016R1D1A1B01016322).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng *et al.*, “Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN,” *Computer Standards & Interfaces,* vol. 54, pp. 216–228, 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 Communications Surveys & Tutorials,* vol. 20, no. 3, pp. 2429–2453, 2018.

[3] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, “Network slicing in 5G: Survey and challenges,” *IEEE Communications Magazine,* vol. 55, no. 5, pp. 94–100, 2017.

[4] ONF. Impact of SDN and NFV on OSS/BSS. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/sb-OSS-BSS.pdf (01 March 2016).

[5] ONF. Central Office Re-architected as a Datacenter (CORD). [Online]. Available: https://wiki.opencord.org/pages/viewpage.action?pageId = 1278047 (03 June 2016).

[6] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca *et al.*, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Communications Magazine,* vol. 55, no. 5, pp. 80–87, 2017.

[7] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine,* vol. 55, no. 5, pp. 72–79, 2017.

[8] 3GPP. Management of orchestration; Concepts, use cases and requirements. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId= 3273 (06 April 2021).

[9] 3GPP. Scenarios and requirements for small cell enhancements. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId= 2590 (18 July 2020).

[10] A. M. Kentis, O. E. Zaballa and J. Soler, "Policy framework for the next generation platform as a service," in *Proc. of 27th European Conf. on Networks and Communications*, June 18–20,, IEEE, pp. 125–129, 2020.

[11] 5GPP. 5G PPP Architecture Working Group: View on 5G Architecture. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2020/02/5G-PPP-5G-Architecture-White-Paper_final.pdf (February 2020).

[12] A. Kaloxylos, "A survey and an analysis of network slicing in 5G networks," *IEEE Communications Standards Magazine,* vol. 2, no. 1, pp. 60–65, 2018.

[13] IETF. Intent-Based Networking-Concepts and Overview. [Online]. Available: https://tools.ietf.org/id/draft-irtf-nmrg-ibn-concepts-definitions-01.html (09 March 2020).

[14] ADN Solution (Autonomous Driving Network). [Online]. Available: https://e.huawei.com/en/material/networking/b040287c277b406eb61ba0ff5087e48a (22 September 2020).

[15] Apstra. Intent-Based Networking: A Next-Gen Vision for the Next-Gen Network. [Online]. Available: https://go.apstra.com/white-paper-apstra-intent-based-networking.

[16] CISCO. Intent-Based Networking Building the bridge between business and IT. 2020. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/digital-network-architecture/nb-09-intent-networking-wp-cte-en.pdf.

[17] ETSI. Zero-touch network and Service Management (ZSM). [Online]. Available: https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf (February 2020).

[18] A. Alkasem, H. Liu, D. Zuo and B. Algarash "Cloud computing: A model construct of real-time monitoring for Big dataset analytics using apache spark," *10th Int. Conf. on Computer and Electrical Engineering,* 11–13, October Edmonton, Canada, pp. 933, 2017.

[19] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings *et al.*, "A connectionist approach to dynamic resource management for virtualised network functions," in *Proc. of the 12th Conf. on Int. Conf. on Network and Service Management (CNSM 2016)*, International Federation for Information Processing, Laxenburg, AUT, pp. 1–9, 2016.

[20] P. Semov, H. Al-Shatri, K. Tonchev, V. Poulkov and A. Klein, "Implementation of machine learning for autonomic capabilities in self-organizing heterogeneous networks," *Wireless Personal Communications,* vol.92, no.1, pp. 149–168, 2017.

[21] M. Asif, A. Muhammad, T. A. Khan, J. J. D. Rivera, J. Iqbal *et al.*, "Energy-efficient auto-scaling of virtualized network function instances based on resource execution pattern," *Computers & Electrical Engineering,* vol. 88, pp. 106814, 2020.

[22] ONF. M-CORD as an Open Reference Solution for 5G Enablement. [Online]. Available: https://opennetworking.org/m-cord/.

[23] L. Peterson, S. Baker, M. De Leenheer, A. Bavier, S. Bhatia *et al.*, "Xos: An extensible cloud operating system," in *HPDC'15: The 24th Int. Symp. on High-Performance Parallel and Distributed Computing*, 16 June, Portland, Oregon, USA, pp. 23–30, 2015.

[24] OAI. OAI Public License V1.1. [Online]. Available: https://www.openairinterface.org/?page_id = 698.

[25] K. Katsalis, N. Nikaein and A. Huang. "JOX: An event-driven orchestrator for 5G network slicing," in *IEEE/IFIP Network Operations and Management Symp.*, 23–27 April, Taipei, Taiwan, IEEE, pp. 1–9, 2018.

[26] N. Nikaein, C. Chang and K. Alexandris. "Mosaic5 g: Agile and flexible service platforms for 5G research," *ACM SIGCOMM Computer Communication Review,* vol. 48, no. 3, pp. 29–34, 2018.

[27] A. Khizar, M. Afaq, T. A. Khan, A. Mehmood and W. C. Song, "IBNSlicing: intent-based network slicing framework for 5G networks using deep learning," in *The 21st Asia-Pacific Network Operations and Management Symp. (APNOMS 2020)*, 22–25 September, Daegu, Korea, IEEE, pp. 19–24, 2020.

[28] K. Abbas, K. T. Ahmed, A. Rafiq, W. C. Song and S. J. Seok, "A ltewifi spectrum aggregation system for 5 g network: A testbed," in *2020 Int. Conf. on Information Networking (ICOIN)*, 7–10 January, Barcelona, Spain, IEEE, pp. 753–755, 2020.

[29] K. Abbas, M. Afaq, T. A. Khan, A. Rafiq, J. Iqbal *et al.*, "An efficient sdn-based lte-wifi spectrum aggregation system for heterogeneous 5 g networks," *Transactions on Emerging Telecommunications Technologies,* pp. e3943, 2020. https://doi.org/10.1002/ett.3943.

[30] J. D. Ravera, T. A. Khan, A. Mehmood and S. Wang-Cheol, "Network slice selection function on M-cORD," *KNOM Review,* vol. 21, no. 2, pp. 2–18, 2018.

[31] M. Afaq, S. "Rehman and WC song "Large flows detection, marking, and mitigation based on sFlow standard in SDN," *Journal of Korea Multimedia Society,* vol. 18, no. 2, pp. 189–198, 2015.

[32] M. Afaq, J. Iqbal, T. Ahmed, I. U. Islam, M. K. *et al.*, "Towards 5G network slicing for vehicular ad-hoc networks: An end-to-end approach," *Computer Communications,* vol. 149, pp. 252–258, 2020.

[33] T. A. Khan, A. Mehmood, J. J. D. Rivera and W.-C. Song, "Machine learning approach for automatic configuration and management of 5 g platforms," in *2019 20th Asia-Pacific Network Operations and Management Symposium*, APNOMS 2019 in Matsue, Japan, IEEE, pp. 1–6, 2019.

[34] T. A. Khan, A. Mehmood, J. J. Diaz Ravera, A. Muhammad, K. Abbas *et al.*, "Intent-based orchestration of network slices and resource assurance using machine learning," in *NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, pp. 1–2, 2020.

[35] G. Urul, "Energy Efficient Dynamic Virtual Machine Allocation with CPU Usage Prediction in Cloud Data Centers" MS thesis, Graduate School of Engineering and Science, Bilkent University, Ankara, Turkey, 2018.

[36] Radisys. Demo MWC-A 17-Multi-Access CORD. *Mobile World Congress Americas 2017*, San Francisco. [Online]. Available: https://hub.radisys.com/videos/demo-mwc-a-17-multi-access-cord (12–15 September 2017).

[37] Radisys. M-CORD-The Open Platform for Emerging 5G Applications. [Online]. Available: https://hub.radisys.com/mobe/sb-mcordopenplatform.

[38] J. Ali and B.-H. Roh, "Quality of service improvement with optimal software-defined networking controller and control plane clustering," *Computers, Materials & Continua,* vol. 67, no. 1, pp. 849–875, 2021.

[39] J. Ali and B. H. Roh, "An effective hierarchical control plane for software-defined networks leveraging TOPSIS for End-to-end QoS class-mapping," *IEEE Access,* vol. 8, pp. 88990–89006, 2020.

[40] J. Ali, B. H. Roh and S. Lee, "Qos improvement with an optimum controller selection for software-defined networks," *Plos one,* vol. 14, no. 5, pp. 1–37, 2019. https://doi.org/10.1371/journal.pone.0217631.