Microservices-Based Provisioning and Control of Network Services for Heterogeneous Networks

Shameemraj M. Nadaf, Sipra Behera, Hemant K. Rath, Garima Mishra, Raja Mukhopadhyay, Sumanta Patro

***Abstract*—**Microservices architecture has been widely embraced for rapid, frequent, and reliable delivery of complex applications. It enables organizations to evolve their technology stack in various domains. Today, the networking domain is flooded with plethora of devices and software solutions which address different functionalities ranging from elementary operations, viz., switching, routing, firewall etc., to complex analytics and insights based intelligent services. In this paper, we attempt to bring in the microservices based approach for agile and adaptive delivery of network services for any underlying networking technology. We discuss the life cycle management of each individual microservice and a distributed control approach with emphasis for dynamic provisioning, management, and orchestration in an automated fashion which can provide seamless operations in large scale networks. We have conducted validations of the system in lab testbed comprising of Traditional/Legacy and Software Defined Wireless Local Area networks.

***Keywords*—**Microservices architecture, software defined wireless networks, traditional wireless networks, automation, orchestration, intelligent networks, network analytics, seamless management, single pane control, fine-grain control**.**

I. Introduction

DUE to the Corona Virus Disease (COVID-19) pandemic, the work scenario at offices and on-premises has taken a setback. Work from Home (WFH) has become new norm in current situations. Many industries have delivered the infrastructure resources such as Laptops, Tablets, Workstations etc., on a war footing directly to household of employees. IT industries and other organizations across the globe have come up with different work models wherein the number of staff working on premises and at office locations is currently restricted to essential/critical categories only.

There has been a drastic change in the operations at on-premises/office locations to adhere strictly to COVID-19 safety measures and guidelines. Different business organization such as, Big Enterprises (BEs), Small and Medium Enterprises (SMEs), Startup companies, Government offices etc., have a mix of staff and based on the nature of work have been divided to WFH or Work from Office (WFO). This drastic shift in the work environment requires use of different sets of technologies. Considering the networking needs of each business entity, the use of remote access technologies is predominant. The access to data centers, branch offices and other premises is critical for many staff members to perform day-to-day operations. Disruptions in the networking can lead to delays and may cause deviations in delivering the customer needs. To support day-to-day operations in the backend, there is a need to maintain the networking infrastructure, which is a heterogeneous environment comprising of vendor products, cloud services, open-source solutions, and management software etc. The maintenance and further sustainability require availability of skilled staff both at on-premises and off-premises at different times. Furthermore, it is essential to adopt new networking technologies (5G, WiFi6 etc.) and networking solutions to boost business prospects, optimize utilization and provide innovative solutions to customers. On the contrary, the number of visits to the offices and other on-premises zones for skilled staff is highly dependent on the prevailing COVID-19 situations and government restrictions. Similar situations can also rise in future.

A better approach at such difficult time is looking at simplifying the network operations by means of customized and dedicated Software Defined Network (SDN) services to cater to needs of the heterogeneous networking environments in place. Technologies such as SDN [1] and Network Function Virtualization (NFV) [2] have emerged as promising solutions for fine-grain control and optimization of high-performance network infrastructures such as Switches, Routers, Firewall, Wide Area Network (WAN) accelerators, and application performance solutions etc. These technologies can help in creating customized services for different management, maintenance, and orchestration needs. Use of the White/Brite box [3] hardware solutions can help in creating customized SDN/NFV related functionalities on actual hardware and can initially be utilized in non-critical functions like branch offices, non-production etc., and moved to critical operations at later stages. Adoption of cloud related services [4] such as Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) etc., can make a positive impact. Typical workloads such as file transfer, backup, mail etc., can be replaced by suitable cloud offerings. Use of on-premises private cloud infrastructure for critical workloads is inevitable since it gives better reliability and productivity.

To adopt SDN/NFV for fine-grained control and use the overall resources which are geographically distributed like one Single Virtualized Infrastructure (SVI), a dynamic on-demand provisioning of services must be realized on top of SVI. Therefore, there is a necessity of a dynamic network deployment and provisioning which is holistic and must cater to different scenarios in a heterogeneous network environment. Furthermore, the application architectures have evolved over the years and new paradigm of small, independent, and agile

Shameem Raj M Nadaf, Sipra Behera, Hemant Kumar Rath, Garima Mishra, Raja Mukhopadhyay, and Sumanta Patro are with the TCS Research & Innovation, Tata Consultancy Services, India (e-mail: sm.nadaf@tcs.com, beherasipra9@gmail.com, hemant.rath@tcs.com, garima.mishra2@tcs.com, raja.mukhopadhyay@tcs.com, sumanta.patro@tcs.com).

microservices is prevalent in the industry. People have slowly started migrating to the microservices for scalability, ease-of operation, and efficiency. Many of the network services required for management, orchestration, and enhancement can be provided by means of microservices. However, there is a need for an efficient system which can smoothen the process of managing the life cycle of these network aware microservices and co-ordinate for achieving the desired goals/business logic in a reliable and scalable manner in heterogeneous network environments. Such a system must be highly distributed, dynamic, and robust in nature.

In this paper, we discuss Microservices Architecture (MSA) based system for smooth and seamless deployment/ provisioning of network services to enable better control and management of the heterogeneous networks. We have performed seamless mobility management and wireless station's link health prediction for usage in network analytics. Furthermore, we have validated this system for multiple services such as monitoring, network analytics etc., on different network types. Moreover, this system also aids in performing a single pane management of the entire network landscape with multitude of services.

The rest of the paper is organized as follows. We discuss the related works in Section II and the problem statement is described in Section III. Further we describe the approach and system architecture in Sections IV and V respectively. Finally, we validate the proposed system through lab testbed deployment as described in Section VI and conclude with details of the future scope of work in Section VII.

II. Related Work

In the enterprise and among the Service Providers (SPs), the maintenance of existing infrastructure along with the adoption of new technologies for better management is a real practical challenge these days [5]. Authors [5] have evaluated the implementation and use of state-of-the-art technologies for building new business models, minimize the Operational Expenditure (OPEX) and digitize the business operations. Keeping in view the use of these new technologies, authors [6] provide the illustrations on adopting the cloud computing technology for enterprise business and service management. Similarly, authors [7] describe the challenges and implementation of SDN as a new networking paradigm for enterprise network management. Selection of the right technology suitable for the business logic and service management in enterprise network is highly essential. These days enterprise network is heterogeneous in nature where apart from existing proprietary physical devices, the virtual software entities are being used for vendor agnostic solutions [8]. Apart from this, some enterprises prefer to use private cloud infrastructure for scalability of services. Use of a single technology for managing these multiple applications and services in enterprise sometimes may not result in desired outcome. In [6], authors use the cloud-based platform for heterogeneous logistic system. Similarly, authors in [8] explain the distributed management of heterogeneous SDN enabled network and legacy network.

Another challenge arises when these heterogeneous enterprise network scales up with multitudes of applications and services. Some of these are like tight coupling among services, recurring end-to-end testing upon addition of new service, chances of single point of failure, platform dependency of different applications etc. These challenges are surpassed using microservices based network application design. Authors [9] provide the comparative analysis between microservice based architecture and monolithic architecture. In MSA, the application services are loosely coupled and designed with single business logic per microservice. Using this architecture, the services in the network are decoupled and this provides a fine granular control over the application's performance thereby overcoming the earlier challenges.

From the above, it is evident that there is a need of a system which manages the life cycle of these microservices in the network. In this perspective, authors [10] have taken initial steps by illustrating the automated management of network with microservices based VNFs. A dedicated platform agnostic system for managing these microservices will smoothen the process and synchronize them. With this system it is easier to manage the application and infrastructure for quick response to failure and change in environmental conditions. This can enable the application to scale without any human interventions. In the following section we describe our approach designing such a system which is practically deployable and helps in fine-grain control of MSA based heterogeneous networks.

III. Problem Statement

Today's Information & Technology (IT) divisions of enterprises, business organizations, SPs, telecom operators (Telecos) etc., are facing challenges to keep up with growing demands of the consumers. IT world is burdened by the high influx of handheld devices and role of these devices in the real world is continuously evolving to address majority of the user needs. The IT related decisions in today's highly competitive market and unforeseen economic conditions are categorically driven by two important aspects: Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). Chief Information Officers (CIOs) must keep in check the overall CAPEX and OPEX related spending within a constrained budget. Vendor dependencies on resources such as Compute, Storage and Network drive more investments and management of infrastructure needs considerable operational costs involving software licenses, accessories, power systems, cooling/air flow systems, backup with disaster management, employee training etc. In practice, it is not feasible to get away with existing infrastructure and bring in new ones for adoption of new technologies.

From enterprise perspective, mainly the pain areas are revolving around how to provide reliable and efficient communication between the application services which may be geographically located either on cloud or other resources. Hence, there is a possibility of having a heterogeneous environment comprising of vendor products, cloud services, opensource solutions, new hardware products etc. in place w.r.t. enterprises/business organization operations. On the other hand, the SPs and Telecos have relatively poor performance w.r.t. achieving optimal CAPEX. There is a compelling drive to invest on technology to catch hold of multiple opportunities w.r.t. consumer demands. However, the reality reflects downward trend in Return on Invested Capital (RoIC). Also, other factors such as regulations, spectrum costs, vendor payouts etc., impact the overall revenue. Moreover, areas such as Internet-of-Things (IoT), Fog/Edge computing, Vehicular networks etc., which have large scale needs will increase the burden by two to three-fold on the communication networks.

In the long run probably best fit for Enterprises/SPs/Telecos will be to have modus operandi involving lower CAPEX and optimized OPEX costs. Currently, the use of different technologies, vendor products, protocols, solutions etc., rather increase the complexity involved and makes it difficult to drive dynamic decisions. Several industry tools and opensource platforms are there which try to cater to different networking needs. However, bringing all of them under a single umbrella to operate in a smooth and seamless manner is a challenging task. A solution which binds any vendor network solution, opensource network solution, hybrid solution etc., to co-exist and drives them towards the optimization goals of dynamic network deployment/provisioning is lacking.

IV. Proposed Approach

We have considered the Domain-driven Design (DDD) and Command Query Responsibility Segregation (CQRS) architecture pattern to derive a functional model. Then, we use MSA as the building block for the actual network services deployment and provisioning. DDD is an architectural pattern primarily based on Domain/Business logic [11] whereas CQRS is an architectural pattern that separates Queries (Read) and Commands (Write) into two different models. MSA is meant for creating loosely coupled services each with a single business task. We do not dwell into details of the DDD principles, CQRS patterns and MSA; specific details have been elaborately discussed in [11]-[13].

As per the principles of DDD, there must be layers of responsibility to achieve the domain related tasks, categorized into three layers: (I) Domain Layer - Any rules or criteria pertaining to the business needs/logic are encapsulated in this layer, (II) Infrastructure Layer - All the technicalities required for implementation are part of this, and (III) Application Layer - This layer acts as an interface to the outside world. It accepts appropriate requests and returns corresponding responses.

We make use of the DDD principles and CQRS pattern in microservices to breakdown vast set of networking related services into a catalog of microservices which can be deployed dynamically. In each microservice, DDD is being utilized to identify a particular domain specific task and CQRS for creating segregation of the task responsibilities. For instance, in a Wireless Local Area Network (WLAN), we consider a task of monitoring (domain specific) which is performed by collection of information such as radio characteristics, traffic statistics and station statistics etc. A microservice responsible for WLAN monitoring can be handling two different models (I) collecting WLAN related information and update of monitoring database, and (II) Response for any queries with respect to monitoring by retrieving from database. Further, intermediary microservices may be required to address the challenges of fine-grain control w.r.t. Traditional/Legacy networking technologies as well as the proprietary vendor solutions. Note that, we do not attempt to create any new services/components in the transport/network/physical layer of the Open Systems Interconnection (OSI) model or make any modifications in the existing vendor/opensource network solutions. In our approach, we start with identification Domain/ Business logic to achieve optimal network resource utilization which involves the following mentioned high-level activities:

A. Dynamic Network Services Provisioning

Depending on the needs of a given business entity (viz., branch, data center, service provider network, customer premises etc.) different network services such as, switching, routing, load balancing, traffic engineering, scheduling etc., can be commissioned, or decommissioned in a dynamic manner to optimize the utilization of the available resources. Different technologies such as Legacy, SDN, NFV, Hybrid [14] etc., are to be dealt with as per their core principles.

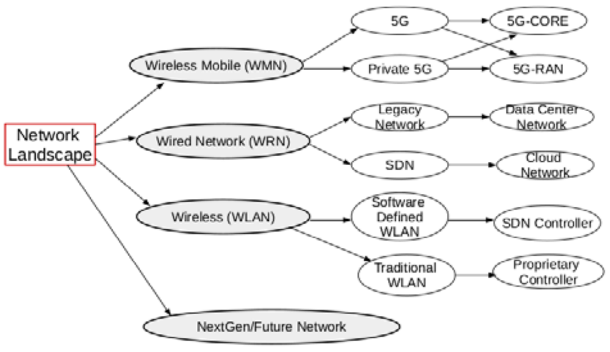

Fig. 1 Network Landscape

B. Lifecycle Management of Network Services

Each network service component commissioned goes through a typical life cycle until decommissioned. Different stages of this life cycle must be managed by appropriate means and fault tolerance techniques must be employed to address reliability concerns. We have proposed a system/method for single pane management and orchestration of entire network landscape of a business enterprise by means of network aware microservices. Fig. 1 illustrates the entire network landscape that we consider for designing the system.

V. Proposed System Architecture

In the proposed system as demonstrated in Fig. 2, the network aware microservices are the applications which interact with networking elements such as network devices, middle management software, vendor software, network data sources etc. These microservices are used for monitoring, data collection, meaningful data extraction, analysis, configuration, control, and event handling etc. For each single autonomous network which may be wired or wireless in nature our system comprises of a Control Unit (CU - one primary and one secondary controller), an IaaS Platform, a communication services platform, a co-ordination services platform, and a set of network aware micro-services applications.

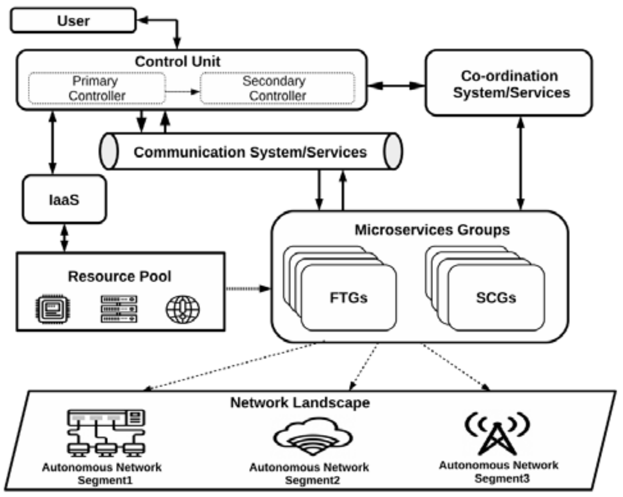
User
Control Unit
Primary Controller
Secondary Controller
Co-ordination System/Services
Communication System/Services
IaaS
Microservices Groups
Resource Pool
FTGs
SCGs
Network Landscape
Autonomous Network Segment1
Autonomous Network Segment2
Autonomous Network Segment3

Fig. 2 System for Single pane management of Networks

A. CU (Primary and Secondary Controllers)

CU is responsible for life-cycle management of network aware microservices and co-operation with CUs of other Autonomous Network Segments (ANS). For reliability and fault-tolerance, a primary and secondary controller combination is provided in each CU. The secondary controller is inactive and receives all events/updates from the primary controller. It becomes active only when the primary fails. Considering z number of ANSs each with a CU, co-operation amongst the different active controllers is achieved by means of periodic message exchange related to microservices activated, utilization, event notifications etc., as illustrated in Fig. 3. The responsibilities of the active controller of a CU in one ANS can be offloaded to active controller of corresponding CU in any other ANS in case of total failure (primary and secondary fail) or high service load.

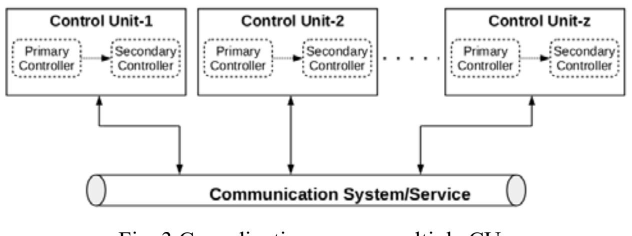

Fig. 3 Co-ordination among multiple CUs

When a user sends a request specific to one ANS, CU goes through various states as illustrated in Fig. 4. Initially when CU is ready to serve any user's request it is in the START state. When the user logs in, the user session is validated by the CU; it transitions from START to IDLE state where it waits for the user to provide the network and service requirements. Upon receiving the initial user input, it transits to DEPLOY MICROSERVICE state, where it identifies the list of microservices, creates workflow, maps the list of microservices to workflow and communicates to the IaaS to deploy the microservices. Here workflow is the list of tasks identified by CU according to the user provided service requirement in an ANS. If any error occurs during this, the CU goes back to IDLE state. However, on successful deployment of microservices the CU transits into CONFIGURE MICROSERVICE state. In this state it configures the microservices with the initial configurations. Post this, the microservices starts operating with initial received configurations and the CU transits to MONITOR state. In this state, it monitors the health of the microservices, and the task assigned to them. However, for the current user request if the CU receives any change in service information, it triggers an update configuration action and transits back to CONFIGURE MICROSERVICE state from MONITOR state. On completion of the task, it transits from MONITOR to REMOVE MICROSERVICE state. In this state it communicates to the IaaS to remove the deployed microservices from the Microservice Groups (MGs). Further when the session ends, it transits to END state.

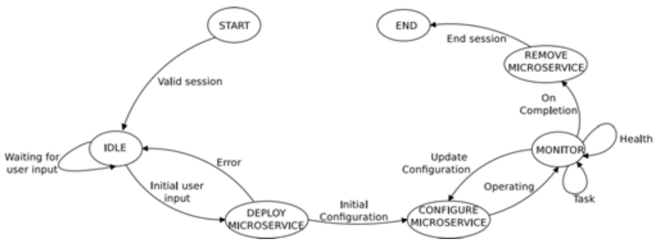

Fig. 4 State diagram of a CU

B. IaaS Platform

One or more IaaS platforms are required for dynamic provisioning of the network aware microservices corresponding to each ANS inside the network landscape. Each IaaS platform helps in hosting microservices on top of the Resource Pool (CPU, Memory, Storage and Network). The active controller in a CU interacts with IaaS platform by means of suitable Application Programming Interfaces (APIs) or any other interfaces provided. The number of microservices to be hosted, resource to be allocated to each microservice, configuration and role of each microservice, details of the ANS the microservice needs to operate on, up-scaling and down-scaling of each microservice etc., are handled by the active controller of corresponding CU. Transferring the responsibilities of one MG to another is also handled by the active controller. Detailed functioning of the IaaS platform is out of scope of the paper.

C. Communication Services

The communication services are responsible for: (I) communication between different microservices, (II) communication between microservices and the active controller of corresponding CU and (III) microservices to any third-party component. A standard publish-subscribe messaging platform is used to handle large number of message exchanges taking place among the entities. Creation of individual communication channels based on the needs and adding/removing subscribers to communication channel is handled by the active controller via the messaging system interfaces.

D. Co-ordination Services

Apart from communication services, proper co-ordination amongst the different microservices and the active controller is very essential for efficient functioning and single pane management. Typical services necessary for co-ordination such as service registry, synchronization, leader election and group management etc., are realized through a standard distributed co-ordination platform. The active controller takes care of providing fault tolerance and scalability for each network aware service by creating one or more MGs with the use of resources provisioned by IaaS.

E. Microservice Groups

Fault Tolerance Groups (FTGs) and Scalability Groups (SCGs) are the two major categories in MGs as illustrated in Fig. 5. Each FTG comprises of one or more master microservices and one or more slave microservices. The master microservices act upon the service requests and each slave request proxies for every received service request to one of the masters. One of the slaves microservices is chosen as master when any one of the masters fail. New FTG is constituted in case of a total failure where no slave microservice is available and the responsibilities of the failed FTG are transferred to the new FTG. In case of SCGs, each microservice acts as a master and serves the received service requests. There is a Service Landscape (SL) as illustrated in Fig. 6, maintained in each active controller. It consists of the details of the services provided by different network aware microservices available for activation (deployed in resource pool of IaaS). Based on the specification in the SL, active controller decides the formation of FTGs and SCGs depending upon the factors such as criticality of the service, nature of service etc.

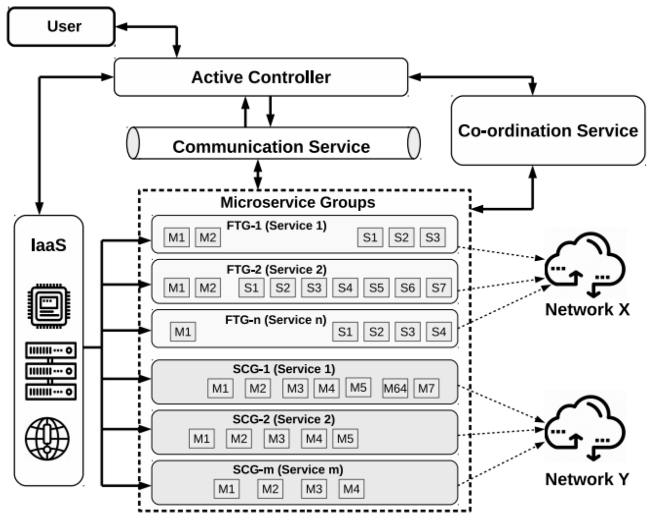

Fig. 5 MGs (FTGs and SCGs)

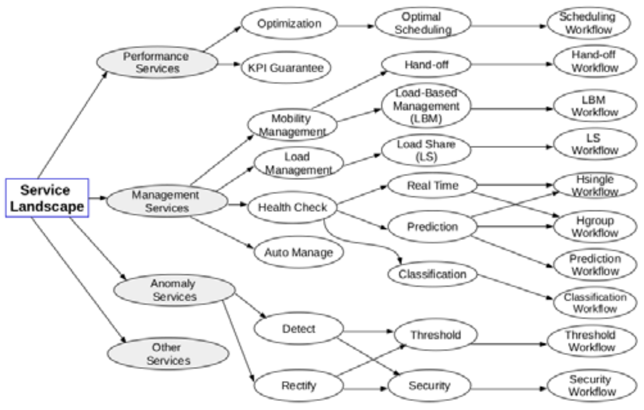

Fig. 6 Service Landscape

Each microservice that is deployed in an MG undergoes a set of states in its life cycle as illustrated in Fig. 7. When the IaaS initiates the deployment of a microservice it is in the START state. Upon successful deployment, it transits to the IDLE state. In this state, it waits for any configuration message from the CU. After receiving the configuration message from the CU, it transits to BUSY state. In BUSY state it runs the task for which the microservice is deployed. While running any task if the CU sends any configuration update (other than removal), microservice handles that. In case of removal of configuration, the microservice transits to IDLE state from BUSY state. On completion of the task or in case of any error, the microservice transits from BUSY to END state. The CU acts as a cognitive brain, where it learns the network functionalities/parameters and application requirements over time, and intelligently orchestrates the desired microservices in a semi-autonomic manner. Since the orchestration mechanism is assisted (with prior knowledge of the network, applications, protocols, and standards), we call the process as semi-autonomic.

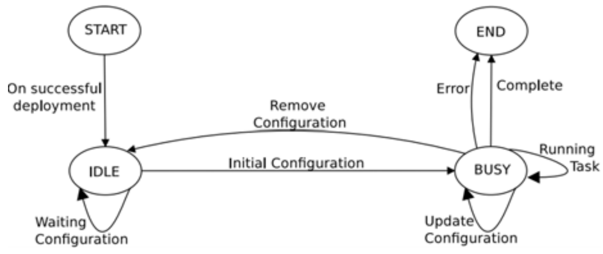

Fig. 7 State Diagram of a Microservice

VI. System Validations through Lab Testbed

We have validated the proposed system in a lab testbed deployment as illustrated in Fig. 8. Technologies used and the infrastructure consumed for different components of the system are described in Table I. The lab testbed comprises of Virtual Machines (VMs) hosted in a resource pool using the VMware ESX hypervisor.

Elastic search [15] is used as the database (non-SQL) and the Graphical User Interface (GUI) is created using Angular JS/ Java Struts technologies. For messages exchange, we have used the Apache Kafka platform [16] deployed as a cluster of three different VMs. Further, for the co-ordination activities such as service registry, group management, status tracking etc., we have used the Apache Zookeeper platform [17] deployed as Quorum of three different VMs. Java Spring based configuration server is also part of the co-ordination system that manages the configurations of each microservice. The microservices are deployed as Docker containers [18] inside the IaaS platform. Apache Mesos has been used as the IaaS platform for containers orchestration. A local docker registry has been used for storage and retrieval of the corresponding container images for different ANS.

TABLE I
SYSTEM DEPLOYMENT DETAILS

System Component	Technology	Deployment Infrastructure
Database	Elasticsearch non-SQL database hosted as a 3-node cluster deployment	3 VMs with 4 CPU, 8 GB RAM and 200 GB HDD each
Dashboard (Graphical User Interface)	Angular JS Frontend and Java Struts 2.0 backend	2 VMs with 4 CPU, 8 GB RAM and 200 GB HDD each
CU (Primary & Secondary Controllers)	Java Spring application deployed in Java Spring boot framework	2 VMs with 4 CPU, 8 GB RAM and 200 GB HDD each
IaaS Platform	Apache Mesos Platform [2 master nodes, 10 slave nodes] for containers orchestration	12 VMs with 8 CPU, 32 GB RAM and 500 GB HDD each
	Local Docker Registry for hosting private docker container images	1 VM with 4 CPU, 8 GB RAM and 500 GB HDD
Communication Service	Apache Kafka Platform [3 brokers cluster deployment] for message exchange between dashboard, CU and the Microservices	3 VMs with 4 CPU, 8 GB RAM and 200 GB HDD each
Co-ordination Service	Apache Zookeeper Platform [3 nodes Quorum deployment] for service registry, fault tolerance, group management etc.	3 VMs with 4 CPU, 8 GB RAM and 200 GB HDD each
	Java Spring based Centralized Configuration server with file system backend for management of microservices related configurations	1 VM with 4 CPU, 8 GB RAM and 500 GB HDD
MGs	Java Spring application as docker containers in Apache Mesos platform	Infra requirement varies based on the functionality of each microservice [hosted inside the Mesos nodes]

In Fig. 8, CU is the central component of the system. It has knowledge different network types and the corresponding services. It parses the user provided requirements through the GUI by subscribing to a topic on Kafka message bus channel. Then, it creates the workflow and identifies the microservices required to perform the tasks. It then commands the IaaS to deploy the required microservice images as the containers as MGs through HTTP Representational State Transfer (REST) communication. IaaS platform internally communicates with the local docker registry to pull the corresponding microservice images for a workflow and deploy them as containers inside the resource pool. Further upon successful deployment of microservices for a workflow, CU decides the configurations for each microservice and shares them to configuration server through HTTP REST communication. The configuration server further broadcasts the configuration to respective microservice through Kafka Spring Cloud Bus Topic. Further, CU monitors all the events of the microservices through Kafka bus and monitors the health status of each microservice via the IaaS platform. CU is responsible for commissioning and decommissioning of each microservice and handling the events in between.

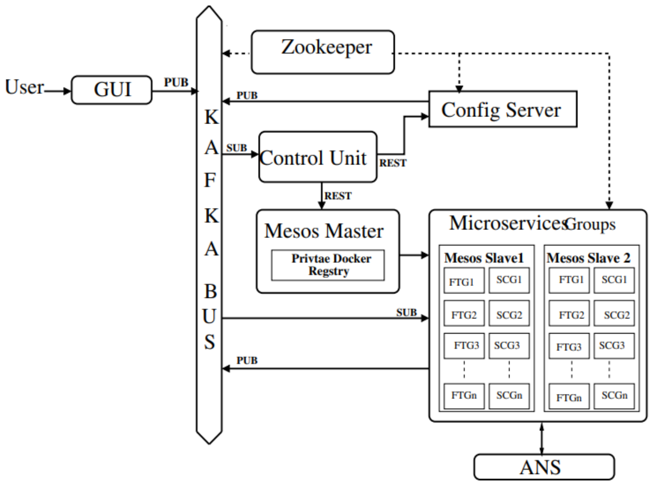

Fig. 8 Lab Testbed Setup

A. Evaluation of Microservices Lifecycle Management

The services specific to any ANS serving a very specific function are considered as microservice. In this scope the microservices perform functions like sensing network parameters, retrieving the network parameters specific to use-case, performing analysis using AI/ML algorithms, deciding the policy, and implementing the policy on the network. In the proposed system these microservices play a vital role. These microservices are bundled as Docker images and stored at private docker registry on IaaS system. The workflow initiated by CU is accomplished by starting the run time instance of the images called docker containers. These containers are managed by Apache Mesos [19] and Marathon [20] on the IaaS system. As illustrated in Table I, we have 2VM server performing the job of Mesos Master along with the Marathon and 10 VM server acting as Slaves for the Mesos Master. These slave servers host the containers deployed in run-time w.r.t. one workflow. The container running status and health is monitored by Mesos Master which passes this information to CU.

In the lab testbed, the microservice images are deployed in groups either FTGs or SCGs. Once the microservice application inside a container receives the configuration, it connects to the Kafka cluster for communication and zookeeper quorum for co-ordination. Zookeeper is responsible for this service registry, service discovery. In case of a critical application that demands fault tolerance, CU instructs Mesos Master to deploy multiple instances of the same docker image in FTG mode. In such scenario among the deployed instances of a single docker image per workflow, one acts as a leader instance and others work as follower. The coordination among these instances is managed by zookeeper which is responsible for leader instance election and leader management [21]. The

Znodes of each instance of a service forms a cluster which has information about live nodes, leader nodes and follower nodes. This information stored at cluster is shared with CU through REST endpoint for monitoring the FTGs in real-time.

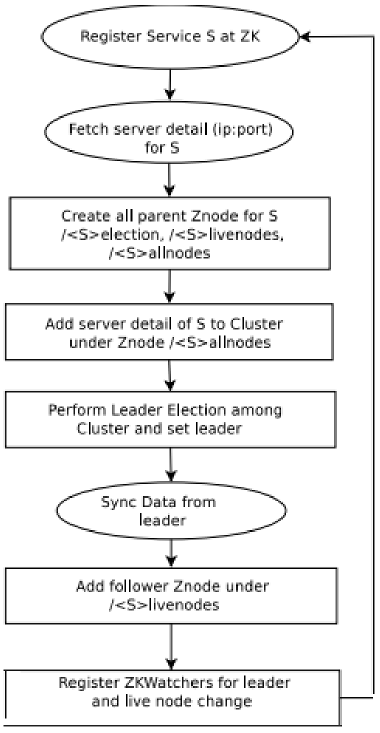
Register Service S at ZK
Fetch server detail (ip:port) for S
Create all parent Znode for S /<S>election, /<S>livenodes, /<S>allnodes
Add server detail of S to Cluster under Znode /<S>allnodes
Perform Leader Election among Cluster and set leader
Sync Data from leader
Add follower Znode under /<S>livenodes
Register ZKWatchers for leader and live node change

Fig. 9 Leader Election in Fault Tolerance Groups

Fig. 9 illustrates the leader election process for the microservice deployed in FTG mode. Depending on the size of FTG decided by CU, Mesos master starts the container of service, S. Each container instance of the service 'S' inside an FTG registers itself with zookeeper as a 'ZK client'. Afterwards, the zookeeper fetches the client's server details like IP address and port number. Once an instance of a S registers, corresponding parent Znodes are created at zookeeper. These parents Znodes are of type election, livenodes and allnodes. Apart from creating these Znodes, the server details of each instance in one FTG are added to cluster by creating Znodes under allnodes. The leader election is taken care by zookeeper among the Znodes under allnodes and labels one instance in FTG as master and others as followers. Follower nodes are added under livenodes. The data from master Znode are synchronized with the follower. In case the master Znode goes down, the cluster is rebalanced, and leader election happens among the available instances under livenodes. This is ensured by registering a ZKWatcher at ZK for monitoring leader change and live node change.

B. Use-Cases for Different ANS

We consider wireless as network category from Network Landscape in Fig. 1, as the ANS. For wireless network we have considered two different network categories: (I) Odin [21] (Software Defined WLAN) and (II) Aruba [23] (Traditional WLAN). Both use-cases utilize the microservices with core functionalities of SADR principles as mentioned in [24]. As illustrated in Fig. 10, the microservices are categorized based on SADR principle and deployed on IaaS platform as MGs. The MGs with functionality of sensing and respond interact with the south-bound ANS like Odin and Aruba network as illustrated in Fig. 10. In lab testbed each ANS comprises of number of devices like SDN controllers, Access Points (APs) and wireless stations.

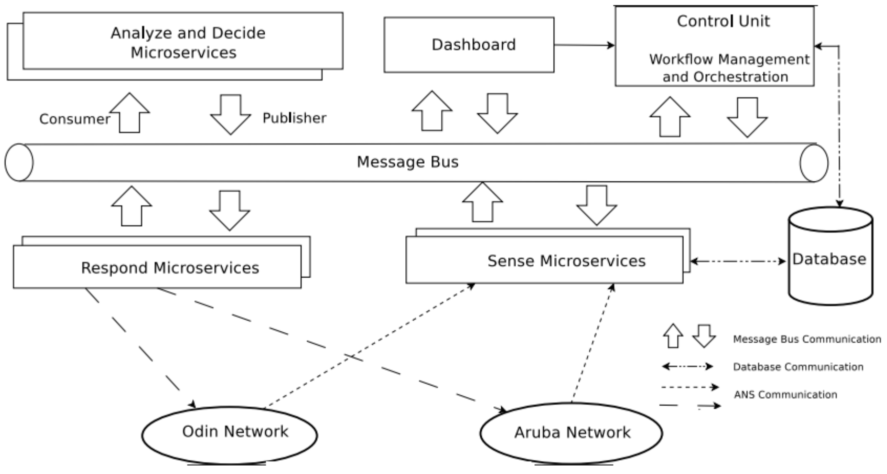

Fig. 10 System validation in Aruba and Odin ANS setup

Use-case 1 Link Health Prediction: A lab testbed of Aruba ANS with two APs and one mobile user running real-time applications walking from one place to other is considered. Both network and service specific micro services are deployed as MGs which communicate with the Aruba network (Traditional WLAN) ANS in this use-case and perform network and service specific workflows. In this use-case the proposed system fulfills the health check service requirement in a TWLAN network by using ML based prediction algorithm. In the Traditional WLAN network, in order to monitor link health between a wireless station and AP, two possible options are available like doing on-the fly prediction at APs or performing prediction on the collected network statistics offline and provide a prescriptive report to the network administrator. Without the proposed system there will be a single application that will sense the network statistics, store them for offline prediction or do the online prediction at AP. However, with the proposed system this service requirement is achieved in a modular way with the help of multiple microservices deployed as MGs performing their respective works without interrupting AP's normal work of routing the packets. After the initiation of prediction workflow by CU, the Mesos Master deploys the required microservices as Docker containers on the MGs. One group of microservice performs the retrieval of required network statistics and publish on the dedicated Kafka topic. These are further consumed by another group of microservice that performs the online ML prediction and publishes the report on a dedicated Kafka topic. This is further consumed by decide MG which sets the policies on the Aruba ANS with the help of respond MG.

Use-case 2 Seamless mobility by smooth handoffs: Similar to Aruba ANS, in this use-case a lab testbed of Odin ANS with two APs, one SDN controller and one mobile user running real-time applications walking from one place to other are considered. In this use-case, the proposed system fulfills the mobility management service requirement in Software Defined WLAN network. In the normal scenario, handoff decision is wireless station initiated which results in hard handoff. However, with the proposed system, handoff decision (both prescriptive and reactive) is taken by the CU while performing the workflow. This reduces the workload on the APs of the network and seamless mobility with significant reduction in packet drops and jitter is observed during handoff. Experiment details are out of scope of this paper. The techniques used for seamless mobility are presented in Algorithm 1.

In Step 1-4, the CU receives the user input and decides the type of workflow (handoff workflow as in Fig. 6). The CU then decides the types of microservices required to complete the workflow. It decides the image name and resource requirement for their deployment. In this case, four microservices are required - sense, analyze, decide, and respond. Sense microservice collects link and load parameters from the network. Analyze microservice runs the proposed handoff algorithm to generate the report on the wireless station's link performance with connected AP as well as all the available APs of the network. Decide microservice consumes the analysis report and provides the decision whether handoff is required or not as a decision report. Finally respond microservice consumes the decision report; triggers the handoff for the wireless station by communicating with the wireless devices.

To deploy these microservices, CU performs Step 7. In Step 7-13, CU initiates and performs the deployment of microservices and sets the status of the workflow to active. In case of any error (Step 27-28), it sets the status as error. CU then performs the configuration of microservices (Step 15-16). In case of failure (Step 23-24), it sets the status as error in configuration. In Step 18, the microservices start their respective core functionalities and publish their report on Kafka bus for consumption of other microservices. These microservices also share their health status to the CU via Kafka bus, which enables the CU to monitor their life cycle and events (Step 19-20). In Step 21, upon completion of the desired workflow, the Mesos Master on IaaS destroys the microservices and clears the resources assigned to them so that it can be re-utilized by any other workflow.

Algorithm 1 Seamless Mobility with Handoff

```
Input: Service $S$, and Network $N$ from user
Output: Complete Workflow $w_S^N$ for Service $S$ and Network $N$
    Initialize : List of existing Workflows in the System $W_L = []$; i.e., $\forall\ w_S^N = []$,
    list of required Microservices $M = []$ for $w_S^N$ and running status of workflow
    $status_{w_S^N} = inactive$
    Begin: Process user input and publish to Kafka bus $K_{bus}$ along-with request ID
    $Req_{id}$
 1: $CU$ subscribes the $K_{bus}$ with $Req_{id}$
 2: process the $Req_{id}$ and extract $N$ & $S$
 3: if $N == TWLAN$ & $S == Handoff$ then
 4:     $w_S^N \leftarrow handoffworkflow$
 5:     $M \leftarrow Se, A, D, R$
 6:     Add $w_S^N$ to $W_L$
 7:     $CU$ initiates REST call $C_{dep}$ with $IaaS$ for deploying $M$
 8:     if $C_{dep} \succ$ then
 9:         Mesos Master at $IaaS$ checks Image $I$ availability for $M$ in Docker Registry
            $R_{reg}$
10:         $status_{w_S^N} \leftarrow active$
11:         if $I \subset R$ then
12:             Mesos Master deploys the $M$ and acknowledges $CU$
13:             $CU$ initiates the configuration for $M$ by preparing the config files $f_M$ $\forall$
                microservice in $M$
14:             $CU$ initiates REST call $C_{conf}$ with config server to upload $f_M$
15:             if $C_{dep} \succ$ then
16:                 central config server broadcasts $f_M$ to deployed $M$
17:                 $Se, A, D, R$ in $M$ receive respective configurations
18:                 $Se, A, D, R$ in $M$ perform respective core functions
19:                 $CU$ monitors $M$ through $K_{bus}$
20:                 $CU$ provides service output to the user through GUI
21:                 $status_{w_S^N} \leftarrow Complete$
22:             else
23:                 Error in configuration of $M$
24:                 $status_{w_S^N} \leftarrow error$
25:             end if
26:         else
27:             Error in deployment of $M$
28:             $status_{w_S^N} \leftarrow error$
29:         end if
30:     else
31:         $status_{w_S^N} \leftarrow error$
32:     end if
33: end if
```

VII. Conclusion and Future Work

In this paper we propose a system for provisioning and management of network services using microservices based framework with emphasis on life cycle management. This system supports different network types forming its network landscape and different service type forming its SL. It is vendor agnostic and adaptable to changes in different networking technologies. We have deployed the proposed system in a testbed setup comprising of Traditional WLAN and Software Defined WLAN networks and successfully conducted experiments like seamless mobility and link health predictions. We aim to deploy the proposed system on a large-scale hybrid network with WiFi6 and other wireless networks including private 5G in future.

Acknowledgment

We are thankful to Bighnaraj Panigrahi, Nikita Trivedi, Samar Shailendra and Ashwin R for their contribution during the initial stage of this work.

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” IEEE Communications Surveys and Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] “Network Functions Virtualization (NFV) – ETSI standards,” http://www.etsi.org/technologies-clusters/technologies/nfv, Nov. 2021.

[3] “The Future of Data Center Network Switches Looks ‘Brite’,” https://www.gartner.com/en/documents/2928517, Nov. 2019.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, T. Stoica and M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing,” Communications Magazine, ACM, vol. 53 issue 4, pp. 50–58, April 2010.

[5] J. A. Wickboldt, W. P. De Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville, “Software-defined networking: management requirements and challenges,” IEEE Communications Magazine, vol. 53, no. 1, pp. 278–285, 2015.

[6] M. Li, P. Lin, G. Xu, and G. Q. Huang, “Cloud-based ubiquitous object sharing platform for heterogeneous logistics system integration,” Advanced Engineering Informatics, 2018.

[7] C. Lorenz, D. Hock, J. Scherer, R. Durner, W. Kellerer, S. Gebert, N. Gray, T. Zinner, and P. Tran-Gia, “An SDN/NFV-Enabled Enterprise Network Architecture Offering Fine-Grained Security Policy Enforcement,” IEEE Communications Magazine, vol. 55, no. 3, pp. 217–223, 2017.

[8] N. Gray, S. Lange, T. Zinner, B. Pfaff, and D. Hock, “Evaluation of a distributed control plane for managing heterogeneous SDN-enabled and legacy networks,” in in Proc. of IEEE Seventh International Conference on Communications and Electronics (ICCE), 2018.

[9] O. Al-Debagy and P. Martinek, “A comparative review of microservices and monolithic architectures,” in in Proc. of IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), 2018.

[10] R. de Jesus Martins, A. G. Dalla-Costa, J. A. Wickboldt, and L. Z. Granville, “SWEETEN: Automated Network Management Provisioning for 5G Microservices-Based Virtual Network Functions,” in in Proc. of 16th International Conference on Network and Service Management (CNSM), 2020.

[11] “What are Microservices?” https://microservices.io/, Nov. 2021.

[12] “What is Command Query Responsibility Segregation (CQRS),” https://culttt.com/2015/01/14/ command-query-responsibility-segregation-cqrs/, Nov. 2019.

[13] “Creating and Using a Command bus,” https://culttt.com/2014/11/10/creating-using-command-bus/, Nov. 2019.

[14] X. Huang, S. Cheng, K. Cao, P. Cong, T. Wei, and S. Hu, “A Survey of Deployment Solutions and Optimization Strategies for Hybrid SDN Networks,” IEEE Communications Surveys and Tutorials, vol. 21, no. 2, pp. 1483–1507, 2019.

[15] “Elastic search – The heart of Elastic Stack,” https://www.elastic.co/elasticsearch, Sept. 2020.

[16] “Getting Started with Kafka,” https://kafka.apache.org/intro, Sept. 2020.

[17] “Welcome to Apache Zookeeper,” https://zookeeper.apache.org/, Sept. 2020.

[18] “Containerization with Spring Boot and Docker,” https://www.split.io/blog/containerization-spring-boot-docker/, 2020.

[19] “Mesos Architecture,” https://mesos.apache.org/documentation/latest/architecture/, 2020.

[20] “Marathon: A container orchestration platform for Mesos and DC/OS,” https://mesosphere.github.io/marathon/, 2020.

[21] “ZooKeeper Recipes and Solutions,” https://zookeeper.apache.org/doc/current/recipes.html#scleaderElection, 2020.

[22] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, “Towards programmable Enterprise WLANS with Odin,” in Proceedings of the first workshop on Hot topics in software defined networks, 2012.

[23] “ArubaOS 8.3.0.x,” https://www.arubanetworks.com/techdocs/ArubaOS 83x Web Help/Content/PDFs/ArubaOS%208.3.0.x%20API%20Guide.pdf, Jan. 2020.

[24] “Seamless Integration of 5G,” https://www.tcs.com/content/dam/tcs/pdf/research-innovation/insights/ Reimagining-Research-seamless-intergration-of-5G.pdf, 2020.