

Integrating RASA Chatbot with an Intent Layer for SDN-Based Network Slice Provisioning

M. Gharbaoui*, P. Castoldi*

*Scuola Superiore Sant'Anna, Italy;

Abstract—The increasing complexity of modern networking infrastructures demands innovative solutions for efficient provisioning and management of network resources. In this paper, we present a RASA-based approach, an open-source conversational AI platform, to develop a chatbot interface for interacting with an intent layer to request Software-Defined Networking (SDN)-based network slices. Our proposed system aims to simplify the process of provisioning network slices by enabling users to express their intent in natural language, abstracting away the complexities of the underlying infrastructure. The chatbot interprets user requests, extracts intents and entities using natural language understanding techniques, and interfaces with the intent layer to orchestrate the deployment of the SDN-based network slices.

Index Terms—Intent-Based Networking; RASA; Network Slicing; SDN; NFV; Chatbot

I. INTRODUCTION

In current modern networking, the concept of network slicing has emerged as a pivotal solution for catering to diverse service requirements and optimizing resource utilization. In fact, network slicing allows for the creation of virtualized, customized networks tailored to specific requirements, offering enhanced flexibility, scalability, and efficiency [1]. As demands for differentiated services continue to grow, the ability to dynamically provision and manage network slices becomes increasingly crucial. However, although announced as one of the key features for 5G and beyond infrastructures, network slicing has been considered a complex and resource-intensive mechanism so far, presenting significant challenges and often requiring specialized knowledge and manual intervention [2].

In this context, Intent-Based Networking (IBN) comes into play as a novel approach in network management to enable loosely-coupled interworking between applications and network operators, and further foster the development of management frameworks thanks to its abstractions and semantics [3]. IBN in fact introduces a paradigm shift by focusing on the abstraction of network policies from underlying infrastructure details thus offering a promising avenue for simplifying network management and automating provisioning tasks. More specifically, IBN enables operators to express their service requirements in high-level manner through directives called *intents*, which are then translated into concrete network configurations and policies by the network operators left with the flexibility of addressing the expressed goals based on their own knowledge and optimization decisions [4]. By leveraging this approach, network providers hold the potential to democratize network slicing, making it more accessible and intuitive for a broader range of stakeholders.

Although IBN represents a corner stone for building orchestration frameworks where network slicing is made accessible also to non-experts, further simplifying the slicing process is fundamental to empower network operators with intuitive tools for efficient network management. An effective approach to address this challenge is to rely on chatbots, which offer several advantages, including simplicity, efficiency, and agility, as they allow users to interact with the network through simple natural language commands, abstracting away the complexities and issues that might be brought by traditional network configuration [5].

In this work, we leverage IBN principles in conjunction with chatbot assistance to tackle the complexities of network slicing. Specifically, we incorporate RASA, an open-source conversational AI platform [6] within an existing intent layer to enhance user experience. The chatbot interprets and translates intents into actionable commands for orchestrating and managing network slices. Thanks to this approach, we envision a more intuitive and efficient means of orchestrating network resources, allowing administrators to dynamically adapt network configurations to evolving demands.

II. RELATED WORK

Adopting IBN within the context of network slicing has been tackled in few research works. [7] provides a solution for network slice orchestration on top of Network Function Virtualization (NFV) platforms to support high-level service requirements that are then converted into low-level TOSCA configurations. Intents are provided as contracts, which define "WHAT" is required by users. [8] uses intents to request transport network slices as part of an end-to-end slice for 5G vertical services. The work focuses on the mechanisms required to translate the user's intent into a transport slice. In [9], authors present a one-touch IBN platform that manages the lifecycle of multi-domain slice resources. The system offers a GUI to select QoS requirements among a limited set of parameters, which are then translated into a slice template.

The integration of chatbots within IBN frameworks has also gathered attention in recent years. In [10], authors propose a framework that relies on a knowledge-based chatbot to distinguish between in-scope and out of scope intents. In [11], authors propose an IBN system (IBNS) that translates customer expectations into management services enabling 5G/6G network elements configuration. The IBNS leverages on a chatbot to take benefit from Natural Language Processing (NLP) while allowing customers to express their intent by

providing answers to simple questions (e.g., Who, What, When and Where). Similarly, a task-oriented chatbot is proposed in [12] and employed for Named Entity Recognition (NER) and intent classification. To train his network models, the author relies on a manually created knowledge base, which can be adapted to any business domain.

Furthermore, several studies have investigated the application of chatbots in IBN orchestration and automation. In [13], authors introduce an intent-driven system that translates high-level service demands and transforms them into continuously monitored deployments. The interaction between IBNS and its customers is performed through a chatbot interface based on an NER model to identify the intent and extract its relevant parameters. In [14], an IBN framework called LUMI is presented. It relies on an abstraction layer between natural language intents and network configuration commands referred to as Nile (Network Intent LanguageE) and is implemented using a chatbot-like interface.

Additionally, few works have adopted RASA framework in their IBN implementation. In [15], authors compare two techniques of intent translation for path creation: a technical-centric one based on specific input without any freedom in user expressions; and a human-centric one based on RASA using unrestricted natural language. They demonstrated that the performance of the IBN framework strictly depends on the quality of the available data and the performance of the classifiers. In [16], authors present Chat-IBN-RASA, a conversational AI chatbot based on RASA. The chatbot allows users to interact with the network management system for the establishment of a connection between 2 sites. Although a Proof-of-concept is provided, the work is still at a preliminary stage. Finally, [17] presents ETS-Chatbot, that extends the functionalities of the RASA chatbot to be used in an IBN context. The authors consider an enterprise environment as a use case and show how their presented framework can accurately classify the intent with high precision.

While the above studies demonstrate the potential of chatbots in IBN, several challenges still remain to be addressed. In this work, we develop a unified framework that allows customers and/or network operators to efficiently deploy their SDN network slices by only specifying their high level goals while easily interacting with a chatbot interface.

III. REFERENCE SCENARIO

A. IBN Architecture

Intent-Based Networking facilitates flexible interactions between service providers, customers, and network operators. It allows service providers and customers to express operational goals through high-level directives called *Intents*, while granting network operators the flexibility to optimize how these goals are achieved. Examples of intents include steering traffic from/to specific areas, ensuring QoS for premium users or providing guaranteed bandwidth for specific services. IBN relies on an Intent Layer, which oversees the lifecycle of intents from their initiation through fulfillment and assurance processes in a closed loop workflow. Fulfillment operations

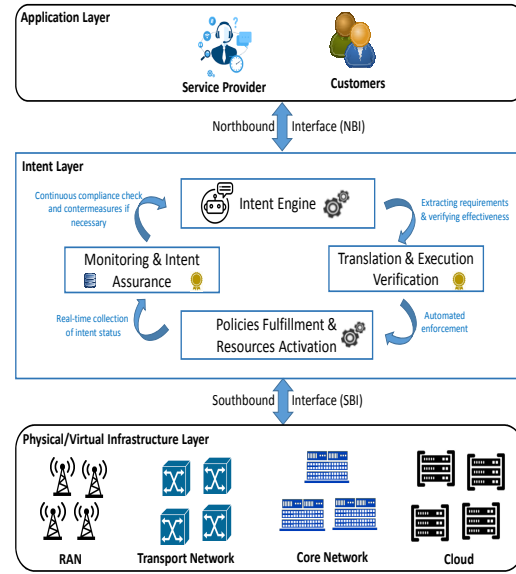


Fig. 1: Intent-based Networking - Reference architecture.

involve processing intents from user initiation to network realization, including configuration tasks. Assurance operations ensure that the network adheres to the intended goals, utilizing real-time monitoring data for assessment and adjustment.

With reference to Fig. 1, these operations are facilitated through the collaboration of several functional blocks. The *Intent Engine* handles user-initiated intents and, if eligible, forwards them to the *Translation & Execution Verification* block. The Translation block analyzes the intents and devises corresponding configuration and provisioning actions based on the current state of the Infrastructure Layer. Subsequently, the Translation block sends the configuration plan to the Verification block, which verifies its feasibility based on real-time network conditions. If feasible, the configuration plan is sent to the *Policies Fulfillment & Resources Activation* block to be deployed automatically using underlying network management systems (e.g., NFV MANO). Once deployed, the *Monitoring & Intent Assurance* block continuously validates the configuration's effectiveness based on real-time network monitoring data. If validation fails, translation procedures are re-initiated for optimization and remedial solutions.

B. SDN-based Slice Intents

In this work, we aim to integrate an Intent Layer that allows for the deployment of SDN-based network slices with a RASA-based chatbot that allows users to specify their service requirements and connectivity needs using a declarative language. At its core, the framework integrates IBN principles with SDN and NFV technologies to achieve efficient network slice deployment.

More specifically, we refer to a network slice as a set of physical and/or virtual networking and computing resources that are allocated to run a specific service offered by network operators to customers. Those slices are SDN-based, which enhances them with the programmability feature to establish

connectivity, e.g., service chains, among software service and application components in a flexible and scalable manner. SDN-based network slices can be applied in several scenarios that go from Smart factories and Smart cities to Augmented reality and E-health. In such scenarios, SDN-based network slices can suitably support the required services by deploying SDN topologies connected to specific virtual functions (e.g., IoT sensors, wearable devices). More details on those scenarios can be found in [18].

Specifying the users' demands in terms of slice capabilities is not an easy task especially for non-expert users who do not have the necessary skills to provide the required parameters that are then associated to ETSI NFV descriptors such as virtual machine required resources (e.g., CPU, RAM, storage), network configurations, topology settings (e.g., connection points) [19]. For this purpose, we propose an IBN approach that allows users to start by expressing their network slice requirements using high-level intents, such as desired QoS, security policies, resource allocation constraints and Service Level Agreements (SLAs). The Intent Engine then interprets the expressed intents and translates them into actionable instructions for network slice provisioning. This translation process involves identifying the necessary network resources, and determining the appropriate SDN-based configurations to fulfill the requirements. Then, the intent is transformed into a network slice request in terms of fully specified NFV descriptors (i.e., Virtual Network Function Descriptors (VNFDs) and Network Service Descriptor (NSDs)), which are coupled with measurable performance indicators for SDN connectivity.

The Policies Fulfillment & Resources Activation component then uses the obtained descriptors to dynamically create network slices by configuring the underlying network infrastructure. These slices are virtualized, programmable, and isolated portions of the underlying network infrastructure, tailored to meet the requirements specified in the intents. During this operation, our IBN framework seamlessly integrates with existing network management systems such as NFV orchestrators and SDN controllers to dynamically adjust network parameters, such as routing paths, bandwidth allocation, and traffic prioritization, based on real-time network conditions and intent-driven policies. Those operations are allowed through APIs provided for interoperability by the framework.

Once the intents are deployed, the IBN framework manages the entire lifecycle of the network slices, from their creation to their release. It monitors the performance of each slice, automatically scales resources in response to changing demands, and enforces policy compliance. It also verifies whether the deployed network slices meet the intended goals and takes corrective actions if deviations occur. This closed-loop approach ensures that the network remains aligned with the users' intents and adapts to requirements over time.

C. Intent specification for SDN-enabled network slices

In this paragraph, we describe the intent specifications required to capture all the parameters required for the deployment of SDN-enabled slices. More specifically, the intent

request must include: the source and destination, the SDN topology type (e.g., linear, mesh) and size, the required bandwidth and the number and type of VNFs deployed within the slice. Other parameters such as QoS class can be provided and can contribute to improve the deployment process. With respect to the way in which the intent is specified, in this work we rely on a chatbot component that interacts with the users in a friendly manner to be sure that the intent contains all the information necessary for its correct deployment. The chatbot represents an intuitive interface that bridges the gap between the users and the complex processes involved in network slices provisioning. By interacting with the chatbot, users can express their intent in natural language, specifying their requirements for network slices without the need for sophisticated technical knowledge. The chatbot then interprets these intents and forwards them to the Intent Engine to translate them into actionable commands for orchestrating the deployment of network slices. Through conversational interactions, users can provide detailed instructions and receive real-time feedback, facilitating a collaborative and efficient deployment process. Additionally, the chatbot can offer recommendations, insights, and troubleshooting assistance, enhancing the accuracy and reliability of network slice deployment while streamlining the overall workflow.

IV. CHATBOT ASSISTANCE FOR USER INTERACTION WITH THE INTENT ENGINE

A. Chatbot Architecture

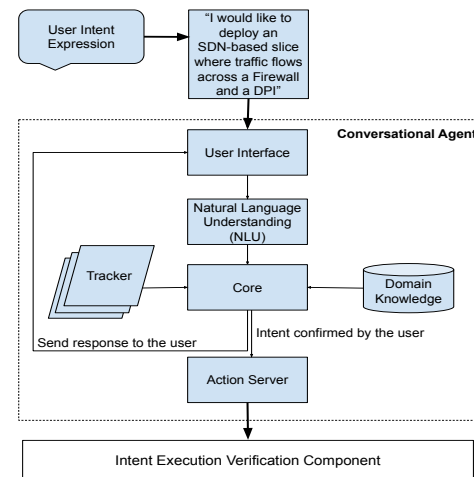


Fig. 2: RASA main components.

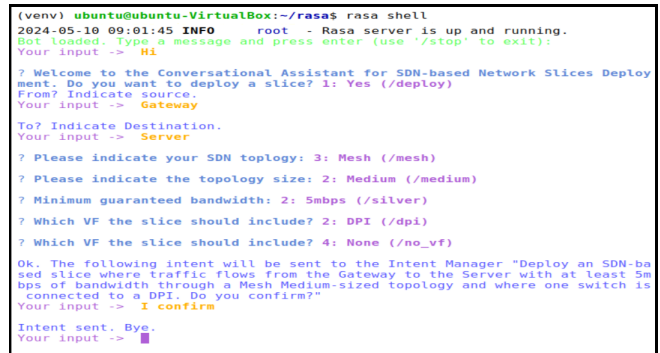
In this work, we implement a chatbot that allows for the classification of the intents sent by the users to the Intent Engine by the recognition of specific entities that are expressed in natural language. The interaction with the chatbot is performed in several steps based on the feedback of the user. Once a decision is reached, the request of the user is then easily mapped to one of the existing descriptors that we have in our catalog and that allow for the deployment of an SDN-based network slice, according to the users' requirements. Our

Policies Fulfillment & Resources Activation component relies then on the RESTful southbound interface to communicate with an NFV MANO and SDN controller to effectively set-up the slice and configure the network. According to the intent specification, the chatbot asks the user to provide the following information: the network endpoints, the SDN topology, QoS in terms of required bandwidth and the type of VNFs to be deployed within the slice. For this purpose, we rely on the RASA open source component [20] since it offers a versatile platform for creating chatbots capable of natural language understanding and dialogue management. With reference to Fig. 2, we provide in the following an overview of its key building blocks:

- **NLU (Natural Language Understanding):** This component processes user input (text) and extracts intents (the user's intention or goal) and entities (i.e., relevant pieces of information) using machine learning techniques. RASA's NLU component can be trained on labeled training data consisting of examples of user input mapped to intents and entities.
- **Core:** handles dialogue management, determining how the chatbot should respond to user input based on the current conversation context and the predicted intent. It leverages a machine learning-based approach, typically using a deep learning model such as a recurrent neural network (RNN) or transformer model, to predict the next action the chatbot should take in the conversation. These actions can be simple messages sent to the user or custom functions executed to respond to the user's intent.
- **Action Server:** handles custom actions that the chatbot can perform, such as querying a database, calling an API, or performing calculations. Custom actions allow the chatbot to perform tasks beyond simple text-based responses, enabling more complex interactions with users. Those custom actions can be defined in a specific file and allow for personalized responses to the user intent.
- **Domain:** defines the universe of possible intents, entities, actions, and responses that the chatbot can understand and generate. It serves as a configuration file for the chatbot, specifying the dialogue flow, action mappings, and other domain-specific settings.
- **Tracker:** keeps track of the conversation state, including the current dialogue context, user input, predicted intents, and entities. It maintains a history of past interactions, allowing the chatbot to maintain context and provide relevant responses throughout the conversation. Each conversation has its own tracker.

B. Intent Classification

The user starts by providing its request to the chatbot, under the form of a text message. The NLU component processes the user input and extracts his/her goal in addition to any relevant pieces of information. For this purpose, it begins by tokenizing the user input, splitting the text into individual words or tokens. This process breaks down the input text into its constituent parts, making it easier to analyze. Next, the



```
(venv) ubuntu@ubuntu-VirtualBox:~/rasa$ rasa shell
2024-05-10 09:01:45 INFO root - Rasa server is up and running.
Bot loaded. Type a message and press enter (use /stop to exit):
Your input -> Hi

? Welcome to the Conversational Assistant for SDN-based Network Slices Deploy
ment. Do you want to deploy a slice? 1: Yes (/deploy)
From? Indicate source.
Your input -> Gateway

To? Indicate Destination.
Your input -> Server

? Please indicate your SDN topology: 3: Mesh (/mesh)
? Please indicate the topology size: 2: Medium (/medium)
? Minimum guaranteed bandwidth: 2: 5mbps (/silver)
? Which VF the slice should include? 2: DPI (/dpi)
? Which VF the slice should include? 4: None (/no_vf)

Ok. The following intent will be sent to the Intent Manager "Deploy an SDN-ba
sed slice where traffic flows from the Gateway to the Server with at least 5m
bps of bandwidth through a Mesh Medium-sized topology and where one switch is
connected to a DPI. Do you confirm?"
Your input -> I confirm

Intent sent. Bye.
Your input -> █
```

Fig. 3: Conversation example for the definition of an SDN-based network slice intent.

NLU component extracts features from the tokens to capture important characteristics of the text and classifies the user's intent based on the set of information that we provided. For this purpose, we determined a set of possible intent classes that can be associated to the users' answers and we specified a set of rules that must be considered for each intent class. In addition to classifying the intent, the NLU component also identifies relevant entities within the user input, which are specific pieces of information mentioned in the text that are relevant to the intent.

C. Intent Confirmation

Once the intent and entities are identified, the NLU component maps them to predefined intents and entities defined in the training data. This mapping allows the chatbot to understand the user's request and take appropriate actions in response. Then, based on the classified intent and extracted entities, the NLU component generates a response and sends it back for confirmation to the user. If the user confirms the response, then the chatbot triggers the appropriate action related to it otherwise, further details can be provided by the user in order to refine the intent.

D. Intent Implementation

Once the user confirms the intent by replying "Yes" to the question "The following intent will be sent to the Intent Manager. Do you confirm?", the Action Server communicates with the Intent Execution Component to check whether it is "eligible" considering the current network and resources status. If yes, the intent is mapped by the Policies Fulfillment & Resources Activation component to a set of descriptors (i.e., VNFD and NSD), which are then sent to the underlying NFV MANO platform to deploy the required slice. Once the slice's deployment is done, the Intent Engine informs the user of the successful completion of the intent installation.

V. PROOF-OF-CONCEPT

In this section, we present a preliminary implementation of our proposal. Since some components have been already tested in our previous work [18], in this paper we focus on the conversational assistance feature of the Intent Engine. All our

experiments were performed on machines with 4 core CPU, 8GB of RAM, running Ubuntu 20.04 LTS.

Our testbed consists in the IBN layer we developed, the Open Source MANO (OSM) platform [21] representing the NFV MANO framework and Openstack [22] representing the Virtual Infrastructure Manager (VIM). OSM MANO and Openstack are installed on 2 different virtual machines, while the Intent Layer and the chatbot run on a third VM.

Regarding the chatbot, we rely on the open source framework RASA [23], which in this current implementation has a limited knowledge base and acts more like a searching engine that maps the requirements of the user to the descriptors that are available in the catalogue. As a preliminary implementation, whenever it was possible, we considered a multiple choice answers approach where the user can select the most appropriate answer among the available ones (e.g., SDN topology: single, linear or mesh). In the other cases (e.g., network endpoints), the user has to enter his/her own response. Fig. 3 presents a conversation example between the user and the chatbot where several interactions are performed before reaching a final suitable expression of the intent. At every new feature added to the chatbot or any change performed to the NLU, RASA must be trained again. To do so, the "rasa train" command must be executed to validate the new entered data and generate the new model to be considered by the chatbot during the conversation.

Once the intent is confirmed by the user, an action is sent by the RASA Action server to the Intent Engine to submit the intent. Once submitted, the Intent Engine performs all the required steps to verify the feasibility of the intent, and then sends it to the Fulfillment component, which interacts with the underlying infrastructure. More specifically, it uses the OSM REST API to onboard the slices descriptors and then deploys the requested slice with the help of Openstack. Any slice is composed of one SDN-based VF (which contains the SDN controller and the emulated network topology) and other virtual network functions that are connected to the SDN-based one. Table. I summarizes the average execution time for each step.

TABLE I: Average Execution Times.

Step	Execution Time [s]
RASA Action Submission	0.24
Intent parsing	0.42
Descriptors Onboarding	3.99
Network Service Instantiation	58.9
Slice Deployment	210.8

VI. CONCLUSIONS

In this paper, we presented an IBN framework enhanced with a RASA-based chatbot that allows for the provisioning of SDN-based network slices. By leveraging RASA, a powerful open-source conversational AI platform, we developed an interface for the request of SDN-based network slices through an intent layer. We also demonstrated the feasibility of using

natural language understanding techniques to enable intuitive interaction between users and network management systems, abstracting away the complexities of SDN configurations and offering users a friendly interface to express their intent. As a future work, we envision further improvements of the conversational agent to add new features and make it more reactive to the intent requests provided by the users.

ACKNOWLEDGEMENT

This work was carried out within the Framework of Department of Excellence in Robotics & AI funded by MUR to Scuola Superiore Sant'Anna.

REFERENCES

- [1] X. Li *et al.*, "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Computing*, vol. 21, no. 5, pp. 20–27, 2017.
- [2] S. Zhang, "An overview of network slicing for 5G," *IEEE Wireless Communications*, vol. 26, no. 3, 2019.
- [3] A. Clemm *et al.*, "Intent-based networking - concepts and overview," in [online] Available: <https://bit.ly/2ImukBF>, January 2020.
- [4] A. Leivadreas *et al.*, "A survey on intent-based networking," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, 2023.
- [5] S. Kusal *et al.*, "AI-based conversational agents: A scoping review from technologies to future directions," *IEEE Access*, vol. 10, 2022.
- [6] D. Z. Alotaibi, "Isolation model for network slicing in 5G network," *Arab Journal for Scientific Publishing*, pp. 29–45, 2020.
- [7] T. A. Khan *et al.*, "Intent-based orchestration of network slices and resource assurance using machine learning," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2020.
- [8] L. Contreras *et al.*, "Transport slice intent," in <https://tools.ietf.org/html/draft-contreras-nmrg-transport-slice-intent-00>.
- [9] K. Abbas *et al.*, "Network slice lifecycle management for 5G mobile networks: An intent-based networking approach," *IEEE Access*, 2021.
- [10] L. Manik *et al.*, "Out-of-scope intent detection on a knowledge-based chatbot," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 5, 2021.
- [11] B. Orlandi *et al.*, "Intent-based network management with user-friendly interfaces and natural language processing," in *27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, 2024.
- [12] N. Ali *et al.*, "Chatbot: A conversational agent employed with named entity recognition model using artificial neural network," *arXiv:2007.04248*, 2020.
- [13] A. Boutouchent *et al.*, "Amanos: An intent-driven management and orchestration system for next-generation cloud-native networks," *IEEE Communications Magazine*, pp. 1–7, 2023.
- [14] A. Jacobs *et al.*, "Hey, lumi! using natural language for intent-based network management," in *USENIX Annual Technical Conference*, 2021.
- [15] A.-R. Meijer *et al.*, "User-driven path control through intent-based networking," in *IEEE/ACM International Workshop on Innovating the Network for Data-Intensive Science (INDIS)*, 2022, pp. 9–19.
- [16] C. Cesila *et al.*, "Chat-ibn-rasa: Building an intent translator for packet-optical networks based on rasa," in *IEEE 9th International Conference on Network Softwarization (NetSoft)*, 2023, pp. 534–538.
- [17] E. El-Rif *et al.*, "Intent expression through natural language processing in an enterprise network," in *IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, 2023, pp. 1–6.
- [18] B. Martini *et al.*, "Intent-based network slicing for SDN vertical services with assurance: Context, design and preliminary experiments," *Future Generation Computer Systems*, vol. 142, pp. 101–116, 2023.
- [19] R. Sharma *et al.*, "An analytical study and review of open source chatbot framework, rasa," *International Journal of Engineering Research Technology (IJERT)*, vol. 9, no. 6, 2020.
- [20] M. Joshi *et al.*, "An analytical study and review of open source chatbot framework, rasa," *International Journal of Engineering Research Technology (IJERT)*, vol. 9, no. 06, 2020.
- [21] "Open source MANO," <https://osm.etsi.org/>.
- [22] "Openstack official site," <https://www.openstack.org>.
- [23] "Rasa open source," <https://github.com/RasaHQ/rasa>.