NETWORK RESOURCE ALLOCATION FOR EMERGENCY MANAGEMENT BASED ON CLOSED-LOOP ANALYSIS

Guda Blessed[1], Ibrahim Aliyu[2], James Agajo[1], Thiago Lima Sarmento[3], Cleverson Veloso Nahum[3], Lucas Novoa[3], Rebecca Aben-Athar[3], Mariano Moura[3], Lucas Matni[3], Aldebaro Klautau[3], Deena Mukundan[4], Divyani R Achari[4], Mehmet Karaca[5], Doruk Tayli[6], Özge Simay Demirci[5], V. Udaya Sankar[7], Sai Jnaneswar Juvvisetty[7], V.M.V.S. Aditya[7], Abhishek Dandekar[8], Shabnam Sultana[9], Jinsul Kim[2], Vishnu Ram OV[10]

[1]Dept. of Computer Engineering, Federal University of Technology, Minna, Nigeria, [2]Dept. of ICT Convergence System Engineering, Chonnam National University, Gwangju, South Korea, [3]Federal University of Pará (UFPA), Belém, PA, Brazil, [4]Tech Mahindra, India, [5]Dept. of Electrical-Electronics, TED University, Turkey, [6]Q Bio Inc., California, USA, [7]SRM University, Amaravathi, AP, India, [8]TU Berlin, Germany, [9]Highstreet Technologies GmbH, Germany, [10]Independent expert

NOTE: Corresponding author: Ibrahim Aliyu, aliyu@ieee.org

Abstract *– The telecommunication system being a critical pillar of emergency management, intelligent deployment and management of slices in an affected area will help emergency responders. Techniques such as automated management of Machine Learning (ML) pipelines across the edge and emergency responder devices, usage of hierarchical closed-loops, and offloading inference tasks closer to the edge can minimize latencies for first responders in case of emergencies. This study describes the major results from building a Proof of Concept (PoC) for network resource allocation for emergency management using a hierarchical autonomous Artificial Intelligence (AI)/ML-based closed-loops in the mobile network, organized by the Internal Telecommunication Union Focus Group on Autonomous Networks (ITU FG-AN). The background scenario for this PoC included the interaction between a higher closed-loop in the Operations Support System (OSS) and a lower closed-loop in Radio Access Network (RAN) to intelligently share RAN resources between the public and the emergency responder slice. Representation of closed-loop "controllers" in a declarative fashion (intent), triggering "imperative actions" in the "underlay" based on the intent, setup of a data pipeline between various components, and methods of "influencing" lower layer loops using specific logic/models, were some of the essential aspects investigated by various teams. The main conclusions are summarised in this paper, including the significant observations and limitations from the PoC as well as future directions.*

Keywords – AI/ML, closed-loop, emergency, network resource allocation, PoC, RAN

1. INTRODUCTION

With the transformation of digital media and communication technology, the use of mechanism centers on the use of network analytics data and social media scraping as data collected is being examined to detect and respond to emergencies [1]. Emergency Responders (ERs) might use various devices which might need real-time Artificial Intelligence (AI)/Machine Learning (ML) inference and transmission. For instance, a firefighter's helmet mounted camera may use image recognition to detect humans in a burning building and transmit it to an operating center for further analysis and action. Therefore, telecommunication systems are a critical pillar of emergency management. However, due to congestion or damage to infrastructure caused by natural disasters or malicious attacks, the communication systems often face difficulty during emergencies. Therefore, it is a complicated task to manually or statically configure networks to support the influx of emergency responders within a verse geographical location [2].

Detection of emergencies and providing connectivity to emergency responders according to predefined Service Level Agreements (SLAs) remains a challenge for the network operators. Furthermore, Next Generation Networks (NGNs) are also expected to operate and manage a heterogeneous network infrastructure with increased complexity that can cope with a wide and flexible range of services, technologies, verticals, and device requirements [3]. Therefore, to offer the service-level requirements of emergency responders, it is pertinent to ensure that network slices operate dynamically and autonomously. Efforts are being made to enable an automatic orchestration of network resources across different domains with a high Quality of Service (QoS),

leveraging Zero-touch network and Service Management (ZSM) and Network Slicing (NS) techniques [3-6].

Therefore, resource allocation is expected to become more complex as the allocation demand changes more frequently, thus creating the need to optimize the network providers' operational efficacy [7]. This is possible in autonomous networks, which can ensure optimal network resource allocation without human intervention. Intelligent resource management in networks can help emergency responders in the affected area through effective NS, etc. [8]. Inputs from emergency responders can be used to optimize resource allocation using close-loops. Integration of closed-loops helps to monitor, analyze, and optimize network configurations while applying operator-specific policies. The use of hierarchical closed-loops has been studied in [7]. The multi-domain architecture of telecommunication networks makes it possible to integrate hierarchical closed-loops in Radio Access Network (RAN), Core Network (CN), and management planes. The use of closed-loops to optimize resource allocation in networks has been studied in [7, 9].

The ITU FG-AN organized a build-a-thon challenge in 2021 to demonstrate and validate important use cases for autonomous networks, creating PoC implementations and tools in the process relating to emergency management. Interactions between a higher closed-loop in the Operations Support System (OSS) and a lower closed-loop in the RAN to intelligently share RAN resources between the public and emergency responder slice were used as the background scenario for this PoC. This study summarises the outcome of the challenge submitted by the various teams that participated in creating the PoC of the use case. The main outputs of the challenge include: (1) the implementation of a higher closed-loop "controllers" in a declarative fashion (intent), (2) the design and implementation of a lower closed-loop with Cloud Radio Access Network (C-RAN) to trigger "imperative actions" in the "underlay" based on the intent, (3) implementation of a simulation environment for data pipeline between various components; formulation of methods/algorithms for "influencing" lower layer loops using specific logic/models, and (4) the integration of the closed-loops and systems into an Open Radio Access Network (O-RAN)-based software platform, ready to be tested in the 5G Berlin testbed.

In this study, we design and deploy closed-loops to optimize detection and resource allocation in case of emergencies. In particular, a set of hierarchical AI/ML-based closed-loops is proposed to intelligently deploy and manage slices for emergency responders in the affected area. A higher closed-loop in the OSS can detect which area is affected by the emergency and deploy a slice for emergency responders to that area. The higher closed-loop sets a resource arbitration policy for the lower closed-loop in RAN, while the lower loop uses this policy to intelligently share RAN resources between the public and emergency responder slice. Furthermore, the lower loop also manages ML pipelines across the edge and emergency responder devices through either split AI/ML models or offloading inference tasks from the devices to the edge.

The main contributions of this study are summarised as follows:

- We designed and implemented closed-loops using a declarative specification. In the design, the Mobile Network Operators (MNOs) instruct the OSS to detect certain emergencies and provide connectivity to emergency responders according to predefined SLA. The operator input is provided as an intent using Topology and Orchestration Specification for Cloud Applications (TOSCA). The resulting YAML file is parsed, and the resulting components are instantiated in a virtualized environment.
- A network testbed with a C-RAN architecture composed of Remote Radio Units (RRUs), Baseband Unit (BBU) pool, and the core network was designed and implemented. In the architecture, a Software-Defined Network (SDN) and RAN controllers work as information sources and agents that dynamically change the mobile and the computer network. An AI agent performs different actions (e.g., resource allocation) in the testbed according to the application, using the information provided by SDN and RAN controllers to train and execute the test stage neural networks. This study shows that validating and applying closed-loop decisions for prioritizing resource allocation for network slices can significantly increase emergency response efficiency.
- We implemented a simulation environment to generate data for model training and testing purposes and to serve as a simulation underlay for testing. Two simulation cases were considered: a standalone case and a New Radio

(NR) dual connectivity case. The results show that prioritized resource allocation can be simulated in different network topologies. The simulations enable us to study various configurations and analyze them to optimize the allocations. Representation of various configurations using text files helps us to create simulation topologies easily. Thus, the SRC (source) node (generating data corresponding to resource usage) and SINK node (applying various configurations in the form of NED files) are possible in the simulation environment.

- The formulation and implementation of various algorithms for an O-RAN-based controller architecture to verify the resource allocation schemes over various domains is actualized. Two algorithms were investigated to analyze the Physical Resource Block (PRB) utilization in RAN. Results were presented considering the need for resources of each slice can vary over time under dynamic networking conditions. The results show the importance of closed-loop implementations in NS, especially for intelligent management of RAN resources during emergency scenarios.
- Lastly, the paper describes the integration of the algorithms and closed-loops above into an O-RAN-based software platform, ready to be tested in the 5G Berlin testbed [10]. We present the integration of the algorithms in an O-RAN near Real-Time RAN Intelligence Controller (RIC) with the Acumos model repository.

This paper is divided into the following sections: Section 2 describes all the contents related to PoC implementation, Section 2.1 describes the design of closed-loops using a declarative specification, Section 2.2 describes a network testbed with a C-RAN architecture composed of RRUs, a BBU pool, and core network, Section 3 presents the creation of a simulation environment to generate data for model training and testing purposes and serve as a simulation underlay for testing, Section 4 describes the various algorithms which can be integrated with an O-RAN-based controller architecture to verify the resource allocation schemes, Section 5 describes the implementation of the above algorithms in an O-RAN near Real-Time RAN Intelligence Controller (RIC) and its integration with the Acumos model repository, Section 6 describes the integration of these algorithms and closed-loops into O-RAN-based software platform, ready to be tested in the 5G Berlin testbed, Section 7 presents the observations from the implementation of PoC, and Section 8 concludes the paper.

1.1 Background and related studies

This section presents some background on the technology and concepts used in this study. It also reviews some prior studies done on the subjects.

O-RAN is an alliance of the Cloud-RAN (C-RAN) and extensible RAN (xRAN) to merge the goals of the two fora. O-RAN aims to achieve an open set of interfaces driven by virtualization and disaggregated components [11]. Based on the desired requirements, the open interfaces enable easy integration of new services for tuning the network. Due to the network automation and self-organization capabilities it affords, the O-RAN alliance has been significant to the realization of 5G networks and beyond [12].

Closed-loop automation is a management function that utilizes feedback signals to regulate itself towards achieving a specific goal [7]. These closed-loops support autonomous behaviour by achieving their goals without external intervention. Typically, in networks, a closed-loop follows a process of sensing, analysis decision, and action. This can be deployed for dynamic resource allocation, self-optimization, self-healing, and automated service assurance [13].

AI/ML has been applied in several areas in networks to improve users' Quality of Experience (QoE) by enabling self-organizing of the networks. Other functions of AI/ML include configuration and detection of failure of base stations, spectrum deployment, NS, and root cause analysis [14-17]. In addition, newer approaches like federated learning have been applied to minimize conventional ML approaches' latency and communication inefficiencies [15, 18]. Example usage includes spatial reuse [19] and resource allocation [18].

Furthermore, efforts have been made in emergency management networks, xApps in O-RAN, and testbed/simulation of virtualized core and RAN. Related studies in these regards are discussed as follows:

Matracia, et al. [20] provide an overview of challenges in post-disaster communications in the context of 6G, airborne and spaceborne networks for emergency management in the networks. Highlights of related studies, physical and networking issues, and practical guidelines and research directions were provided. [21] describes a solution based on edge computing that reduces the amount of data transmitted in times of disaster. Federated learning has also been proposed to

provide extra resources to the edge dynamically during a disaster [22]. [23] discusses RAN slicing mechanisms for providing an appropriate amount of radio resources for emergency responders in a given use case. [24] discusses a self-planning solution for RAN slicing management based on live network measurements that enables dynamic RAN slice allocations. Similarly, [25] proposes an intent-based automated slicing mechanism for core and RAN, which can be used to provide slices for mission-critical services.

Furthermore, Kułacz and Kliks [26] propose the definition of policies in a file form for Dynamic Spectrum Access (DSA) function in OpenRAN networks in implementing xApps in O-RAN. [27] describes a modular xApp implementation in O-RAN for a traffic steering use case based on the Open networking approach. [28] xApps have been employed on O-RAN defined Near Real-Time Radio Intelligence Controller (NearRT RIC) for RAN optimization at the RAN edge.

Several testbed/simulation implementations have been proposed. For instance, [29] describes Simu5G, a system-level simulator for 5G networks. Py5cheSim, a 5G simulator that supports RAN NS, has been developed [30]. [31] describes design 5G-LENA, an ns-3 module that supports end-to-end NR system-level simulations. [32] describes FlexRIC, a platform for real-time RAN control applications, [33] describes SD-RAN, which provides an O-RAN compliant cloud-native platform that can host RAN control applications, and [34] describes how an AI/ML model workflow can be deployed in O-RAN SC Near-RT RIC platform.

2. THE POC DESIGN AND IMPLEMENTATION

This study proposes the use of analytics in Service Management and Orchestrator (SMO) [35] in combination with predictive resource allocations to specific edge locations based on detected emergencies to implement the PoC. A high-level strategy/policy to reallocate resources among the slices in the non-real-time RAN Intelligence Controller (non-RT RIC), forms the first level of the closed-loop. The decision in the higher level closed-loop in the non-RT RIC to reallocate resources may depend, among other things, on the type of emergency, e.g., a natural disaster such as an earthquake, law and order situation, traffic accidents, etc. A RAN-level may complement this higher-level closed-loop that uses other inputs from emergency responders to arbitrate resources among RAN nodes. Such lower-level closed-loops may be hosted nearer to the edge, e.g., near-RT RIC. The policy input from the higher loop may indicate, among other parameters, the different sources of data for the lower loop, such as system and service data. A RAN level closed-loop might also decide to offload inference tasks from ER devices to either the edge or use a split AI/ML model to run inference tasks on edge and ER devices. This decision might be taken based on the available network and computing resources. Some layers of the AI/ML model may be hosted in the wearable devices of the emergency responders, which will help in locating persons under distress using various inputs such as Global Positioning System (GPS) coordinates.

Workflows for the closed-loops at the different levels are independent of each other. The only interaction between closed-loops is via high-level intents over the inter-loop interface. The closed-loops can create new closed-loops in other network domains without human intervention. Each loop can evolve independently, although loops are deployed hierarchically. It can use different models and ML pipelines as required. Each loop may move up or down the autonomy levels as defined in ITU standard, Recommendation ITU-T Y.3173 [36]. The closed-loops can split and provision AI/ML models to other closed-loops in an automated fashion. In addition, we provide a low orchestration delay, better privacy, and flexibility for verticals (e.g., industrial campus networks) by making closed-loops in the edge domain autonomous. Higher loops can use historical knowledge to optimize and generalize lower loops using high-level intent, resulting in increased efficiency of lower loops while preserving their autonomy (e.g., the higher loop might know certain ML models that are good for cyclone emergency management based on previous cyclones). Fig. 1 presents the workflow sequence for the simulation/testbed.

2.1 Design of closed-loops using a declarative specification

The high complexity of management of future networks, which includes the ability to provide new innovative services using complex network configurations, has led to requirements for autonomous behaviour. To enable low latency response by emergency responders, the use of autonomous networking concepts, including the following factors, were found important: (1) application of intent-based mechanisms to

coordinate closed-loops and (2) translation of intents into decisions and actions. These mechanisms allow seamless design, deployment, and management of emergency resources in the networks using operator-friendly intents.

Several studies have been conducted regarding close-loops. For example, Gomes, et al. [7] presented a method for formulating and managing closed-loops using requirements communicated through intents. They propose new management functions for intent delegation, escalation, and reporting while focusing on how intent management can be integrated into the ZSM framework. Luzar, et al. [37] compared four TOSCA-compliant orchestrators; Opera, Yorc, Indigo, and Cloudify. This comparison was made regarding ease of usage, open-source availability, licenses, and operating systems supported by the orchestrator. Ram O.V, et al. [38] carried out a gap analysis of existing frameworks in autonomous networks. Fig. 2 shows a high-level flow chart of the intent for closed-loops activity, starting with the design of the controllers. An intent is written according to the design of the controllers. This high-level intent is parsed, and appropriate closed-loops are set up to meet the objective of the intent.

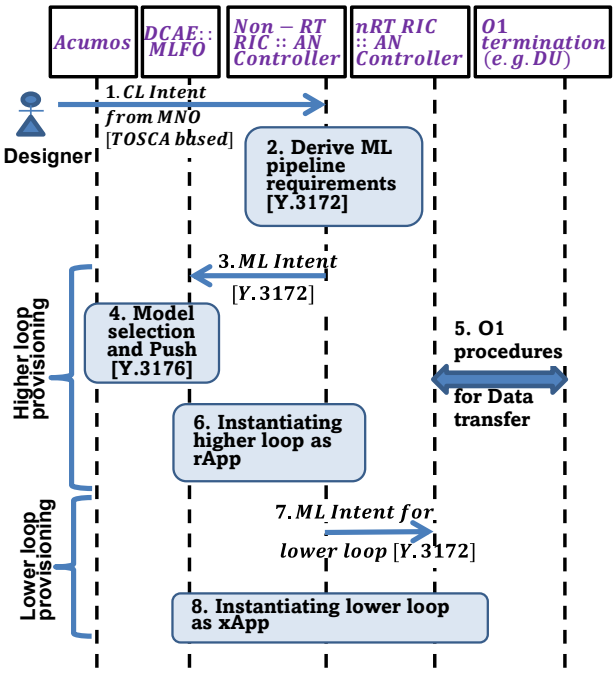


Fig. 1 – Overview of the intent-based design and implementation of hierarchical closed-loops, including simulation and testbed domain

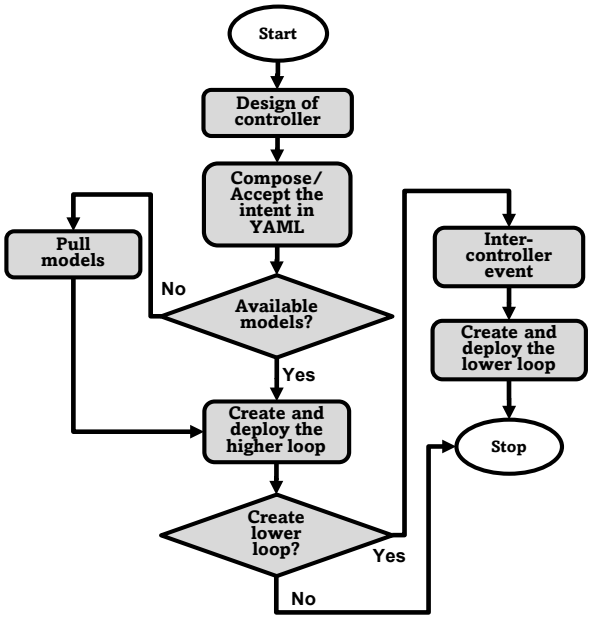


Fig. 2 – The high-level flow chart of the study toward intent for closed-loops

```
model_node:
  derived_from: tosca.nodes.SoftwareComponent
  attributes:
    download_model:
      description: URL to download ML model
      type: string
      default: "http://localhost:8080/model1"
    catalogid:
      description: catalog ID
      type: string
      default: "0"
    revisionid:
      description: revision ID
      type: string
      default: "0"
    solutionid:
      description: solution ID
      type: string
      default: "0"
  interfaces:
    Standard:
      operations:
        create:
          implementation: playbooks/create_model_node.yaml
          inputs:
            link_for_download_2:
              type: string
              default: {get_attribute: [SELF, download_model]}
```

Fig. 3 – Example – Declarative intent in YAML format

Fig. 3 shows an excerpt from the service model showing the definition of the model node. The intent specifies the model node with attributes, including the URL for pulling the ML model from a repository. Additional attributes like catalog ID, revision ID, and solution ID may be used to identify the model. For the implementation, Opensource orchestrator xopera [39] was considered, and simple controller requirements were derived.

Fig. 4 shows the setup considered in this activity demo. The intent is written in TOSCA YAML v1.3. The intent is to create a three-node closed-loop comprising of a source, model, and sink nodes (corresponding to data collection, analysis, and application). This intent is parsed by the xopera orchestrator [39] for the deployment of the closed-

loop. The model metadata and repository are defined based on the standard, Recommendation ITU-T Y.3176 [40].

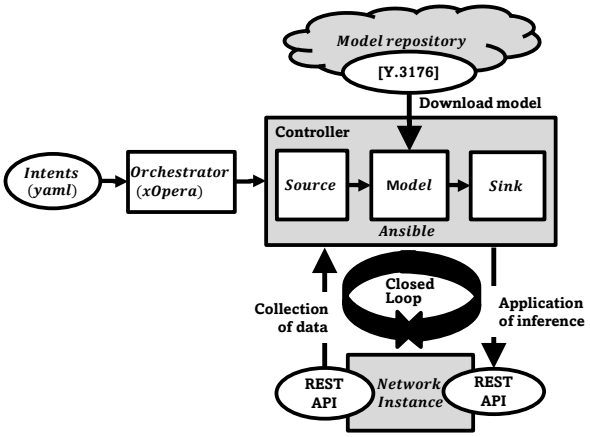


Fig. 4 – Setup design for creation and parsing of example intent

Fig. 5 shows the outputs specified for the three nodes (source, model, and sink). The outputs specified are the attributes of the nodes, which consist of the Application Programming Interface (APIs). Fig. 6 shows the parsing of the service model for the deployment of the three nodes. The APIs in the intents are parsed into three JSON (JavaScript Object Notation) files. Three docker containers are created for implementation, which uses the APIs for data collection, analysis, and adaptation. Dummy data based on the 3Vs (Velocity, Variety, and Volume) and dummy h5 model are downloaded from corresponding repositories according to the specified links. This study shows that a closed-loop can be represented and designed using a standard template demonstrated here using a three-node closed-loop (i.e., SRC node, ML node, and SINK node).

```
outputs:
  output_src_URI:
    description: Rest API to fetch data
    value: { get_attribute: [ source_node, rest_api_attribute ] }
  output_attribute_csv_file:
    description: Sample source data
    value: { get_attribute: [ source_node, my_csv_file_attribute ] }
  output_solution_id:
    description: Solution ID
    value: { get_attribute: [ ml_node, solutionid ] }

  output_catalog_id:
    description: Catalog ID
    value: { get_attribute: [ ml_node, catalogid ] }

  output_revision_id:
    description: Revision ID
    value: { get_attribute: [ ml_node, revisionid ] }

  output_sink_uri:
    description: Sink REST API to set values
    value: { get_attribute: [ sink_node, rest_api_attribute ] }
```

Fig. 5 – Creation and parsing of intent in YAML

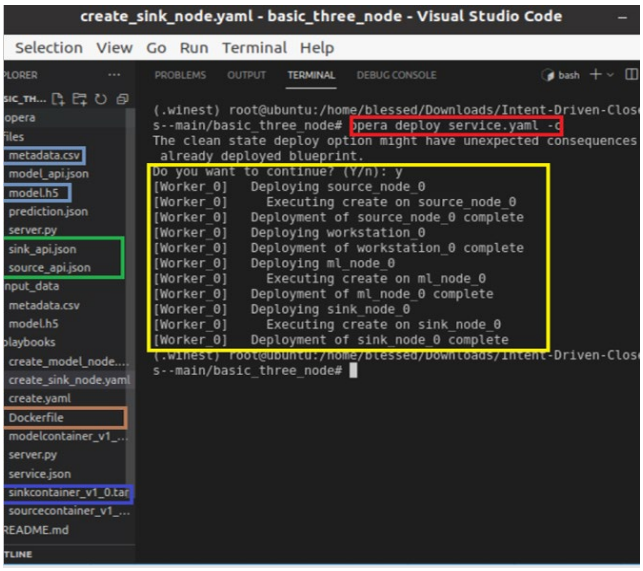


Fig. 6 – Deployment of the nodes

2.2 "Imperative actions" in the "underlay" based on the intent

Validating and applying closed-loop decisions in the network is one of the major challenges in the intelligent allocation of resources for an emergency. The capability to build flexible and realistic AI-based scenarios with different network topologies for 5G and quickly deploy and assess them is important in emergency scenarios. This section describes a network testbed with a C-RAN architecture composed of RRUs, a BBU pool, and a core network. A network testbed called "Connected AI" is described in [41]. The SDN and RAN controllers work as information sources about the network. Furthermore, they work as agents to dynamically change the mobile and the computer network. An AI agent performs different actions in the testbed according to the application using the information provided by SDN and RAN controllers to train and execute in the test stage. The ML workloads are orchestrated along the cluster to provide the AI agent processes.

Results from this study show that the validation and application of closed-loop decisions for prioritizing resource allocation for network slices can significantly increase the efficiency of emergency response. This was demonstrated using priority assigned to an Unmanned Aerial Vehicle (UAV) drone based on a three-node closed-loop, i.e., source (SRC) node, ML node (AI Agent) and sink (SINK) node defined into ITU ML proposed architecture [42].

2.2.1 Connected AI (CAI) network testbed

The CAI testbed deploys a 5G mobile network with a virtualized and orchestrated structure using containers while focusing on integrating AI applications [8]. It uses open-source technologies to deploy and orchestrate the Virtual Network Functions (VNFs) to flexibly create various mobile network scenarios with distinct fronthaul and backhaul topologies. Distinctive features of the testbed are its low cost and the support for using AI to optimize the network performance.

Fig. 7 shows the testbed structure with a C-RAN architecture composed of RRUs, the BBU pool, and the CN. The transport network is emulated by software using Mininet [43], enabling the deployment of different network topologies without real network components (such as switches and routers).

The network contains two main controllers: the RYU SDN controller [44], which is responsible for controlling the transport network emulated by Mininet, and the Open-Air Interface (OAI) FlexRAN controller [45], which is responsible for controlling the base stations deployed in the testbed. Both controllers are connected to the AI agent, which receives network information from controllers and applies commands to change the network operations. No Management and Orchestration (MANO) component was implemented since the main objective of the testbed is to explore focused scenarios which do not include full end-to-end slice support to maintain simplicity and low costs.

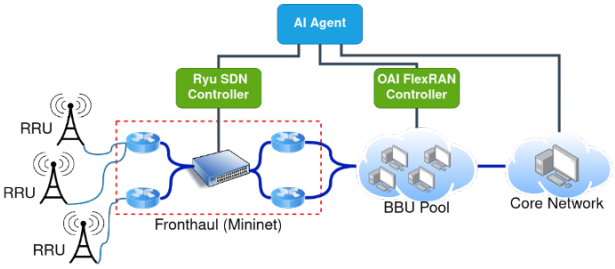


Fig. 7 – Design of the proposed testbed network [41]

To facilitate the deployment of each VNF into containers in different environments and give more flexibility to move these functions to different computers in a cluster, all the testbed components were implemented into container [46] images. The RAN functions and controller were implemented using the OpenAirInterface software [47], while the core network functions were implemented using the Free5GC software [48]. These VNFs, implemented into docker containers, are orchestrated using Kubernetes software [49], enabling the management of the containers as well as the cluster and facilitating the deployment of different mobile network architectures. Fig. 8 shows the VNFs distributed along with the cluster and using a Software-Defined Radio (SDR) to generate Radio Frequency (RF) signals to connect the UE to the mobile network generated by the testbed. The VNFs' location can be defined by scripts as instructed at the testbed repository publicly available [41].

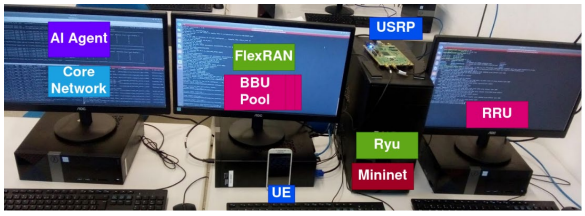


Fig. 8 – Testbed working at LASSE – UFPA lab using a C-RAN architecture [41]

Both the fronthaul (connecting RRU and BBU) and the backhaul (connecting the BBU and core network) are implemented over Ethernet links. Therefore, the transport network complexity usually depends on the network infrastructure available, such as switches, routers, and other network equipment. We implemented the transport network with the Mininet software to decrease costs and increase the flexibility to deploy different transport network scenarios without infrastructure changes. It emulates different topologies with routers and switches with SDN support to make the emulated network management. Then, the transport network topology can be defined in Mininet scripts and different network topologies can be tested without extra network equipment. Our scenario deployment scripts are responsible for forwarding the traffic from fronthaul/backhaul through the emulated network topology. The Mininet also allows to define some network behaviour such as packet loss rate, the bandwidth available in each emulated network link, and latency among each node and other characteristics that give a remarkable amount of flexibility to test algorithms over different network topologies and conditions. Fig. 9 shows a scenario deployed using the testbed where the backhaul link is emulated using a Mininet. A simple network composed of two routers and one switch is emulated, adding a latency of 100 ms between the BBU pool and core network.

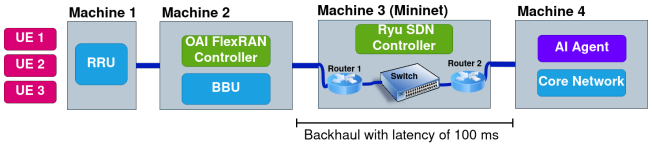


Fig. 9 – RAN slicing scenario with 3 UEs connected to a C-RAN structure with the backhaul virtualized using Mininet [41]

Each component from the Mininet network devices (routers and switches) was connected to an RYU SDN controller that receives information about the transport network, such as the throughput transmitted in each link, dropped packets, and latency. In addition, the SDN controller can apply commands to change transport network operations, such as changing routes and applying congestion control algorithms, thus enabling the usage of external apps to provide transport network management through communication with the SDN controller to promote changes in the network while it is operating. Fig. 10 shows information obtained from the SDN controller about switch 3 in a topology emulated with Mininet [48]. It gives real-time information about the switch operation, such as the number of received and transmitted packets and the port being used.


```
{
    "3": [
        {
            "tx_dropped": 0,
            "rx_packets": 126973,
            "rx_crc_err": 0,
            "tx_bytes": 2045095,
            "rx_dropped": 0,
            "port_no": 1,
            "rx_over_err": 0,
            "rx_frame_err": 0,
            "rx_bytes": 6247861542,
            "tx_errors": 0,
            "duration_nsec": 97000000,
            "collisions": 0,
            "duration_sec": 662,
            "rx_errors": 0,
            "tx_packets": 30943
        }
    ]
}
```


Fig. 10 – Information obtained from SDN controller API about the switches running in the Mininet emulated network [41]

The FlexRAN controller works as an abstraction of the RAN resources and provides an API that enables the service orchestrator entity to dynamically manage the RAN resources to provide information about the mobile network [50]. The FlexRAN protocol [45] defines and implements a software-defined RAN architecture integrated with the OAI platform, which incorporates an API to separate control and data planes for the mobile RAN. This architecture has a master controller represented by the FlexRAN controller in Fig. 9 and a FlexRAN agent corresponding to the OAI eNB instances. Fig. 9 also represents the FlexRAN agent in the OAI BBU instances in a C-RAN scenario. The agents can act as local controllers with a limited network view and handle the functions delegated by the master or coordinated by the master controller.

The FlexRAN agent API separates the control and data plane, allowing the control data to be managed by the FlexRAN controller and the eNB data plane on the opposite side. Fig. 11 shows the information received from a base station using the FlexRAN API, providing information such as the functional split being used, the number of user equipment (UEs) connected, buffer occupancy, and scheduling information. The FlexRAN APIs enable the development of applications related to the control and management of the RAN resources [32], e.g., schedulers, interference, and mobility manager. Moreover, applications related to improvements in the use of RAN resources make more sophisticated decisions [32], such as RAN slicing and adaptative video streaming based on channel quality.


```
{
    "bs_id": 10001,
    "agent_info": [
        {
            "agent_id": 1,
            "ip_port": "192.168.46.176:39024",
            "bs_id": 10001,
            "capabilities": [
                "LOPHY",
                "HIPHY",
                "LOMAC",
                "HIMAC",
                "RLC",
                "PDCP",
                "SDAP",
                "RRC",
                "S1AP"],
            "splits": [
                "nFAPI"]
        }
    ],
```


Fig. 11 – Information obtained from FlexRAN API about the base station running in the testbed [41]

The AI agent is implemented based on the ITU-T Y.3172 [36] architecture that defined a logical interoperable architecture for future networks, which incorporates an ML overlay that operates on top of any specified underlay network technology [50]. This architecture facilitates deploying ML applications in different network scenarios and is adopted in the Connected AI (CAI) testbed. Specifically, ITU-T Y.3172 defined high-level architectural components to integrate ML into the network and a process pipeline [42]. Fig. 12 shows these components, the pipeline, and their respective mapping into the CAI testbed components. This testbed orchestrates the ML

workloads of the AI agent using the Kubeflow tool [51]. Kubeflow works integrated with Kubernetes to orchestrate the ML functions along with the cluster machines. Kubeflow enables the use of pipelines to define the steps of ML processing. Due to the high resource available in the cloud in real scenarios, CAI deploys the AI agent at the cloud location (with the core network) for simplicity.

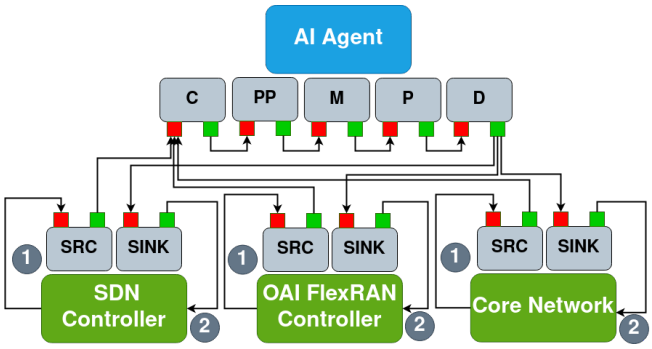


Fig. 12 – ITU-T FG-ML5G ML architecture integrated into the testbed structure [41]

2.2.2 Results and discussions

Some results exploring UAVs in critical missions using the testbed are presented in [52]. This study presents how the AI agents and the network can be adapted to assist mobile network users in Search, Diagnostic and Rescue (SDAR) missions. Fig. 13 shows the results for a scenario with an AI agent controlling the number of radio resources using the RAN slice to prioritize drones in SDAR missions about other UEs connected to the network. In the scenarios without slices, the base station tries to provide an equal amount of radio resources among the UEs without differentiating the applications. When the RAN slice is used and the AI agent set a slice to the drones in the SDAR mission, the AI agent updates the number of radio resources allocated to the drone's slice to guarantee at least 10 Mbps of throughput, the other slice with UEs receives only the remaining radio resources since it has less priority. It shows that a closed-loop can be implemented to control the testbed mobile network using AI methods despite the simplicity of the experiment the AI agent used.

3. SIMULATED UNDERLAY FOR CLOSED-LOOP-BASED RESOURCE ALLOCATION

To complement the testbed described in Section 2.2, this section describes the creation of a simulation environment [29] to generate data for model training and testing purposes and also to serve as a simulation underlay for studying the impact of the

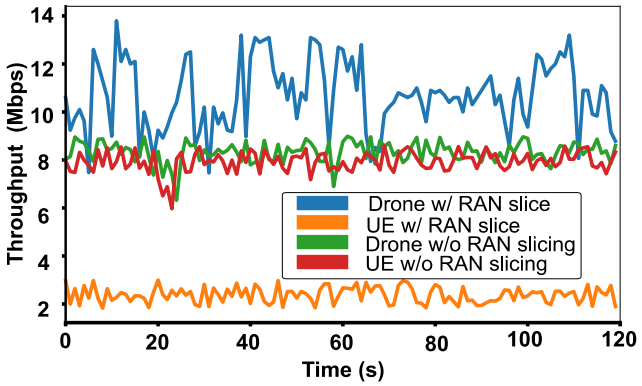


Fig. 13 – Results without RAN slicing and a scenario with RAN slicing using an AI agent

closed-loop on resource allocation scenarios in Medium Access Control (MAC) layer. Simu5g [29] is used to generate output data shown in the results Section 3.2 (e.g., Average served blocks in Downlink/Uplink) based on input parameters given in Table 1 (e.g., Frequency Correction Burst (Fb) Period, target block error probability(BLER), etc.) while simulating the various scenarios. This facilitates studying the machine learning algorithms' impact on resource block allocation by predicting the resource requirement at the UE. Simu5G is based on the OMNET++ simulation framework and incorporates the simulation modules from the INET library [29]. It simulates both the data plane of 5G RAN and the core network.

Two types of simulation scenarios were considered using the Simu5G-standalone case and NR dual connectivity case. In the standalone scenario, gNB is connected to the data network through the core network, while in the NR dual connectivity case, gNB is connected to the eNB through an X2 interface. In addition, the eNB provides access to core and data networks. Sections 3.1 and 3.2 discuss the simulation and the results. The simulator configurations used in this study are simu5g v1.2.0, INET v4.3.2 or above and OMNET++ v6.

3.1 Simulation scenarios

This study defines two simulation scenarios: "single cell with secondary gNB" and "multi-cell with secondary gNB". These two scenarios (called "networks" in the simulator) are defined in the NED (Network Description) file in OMNET++ which the structure of a simulated network can be described. NED enables the user to declare simple modules and connect and assemble them to form compound modules. Compound modules e.g., a "single cell with secondary gNB" and a "multi-cell with secondary gNB" network, can be used as simulation modules.

Parameters can achieve their value from either the NED file or the configuration, i.e., .ini file. Every configuration file has a "General" section that has general parameters like simulation time limit ("sim-time-limit" is the physical time that is set for simulating the network). The "network" keyword is used to flag the network that needs to be simulated.

Fig. 14 shows the two networks which are defined in the NED file: "single cell with secondary gNB" and the "multi-cell with secondary gNB". The main modules that are used in this network are compound modules: carrier aggregation (carrierAggregation), packet gateway (pgw), LTE base station (masterEnb), NR base station (secondaryGnb), and UE. The carrier aggregation module is responsible for assigning multiple frequency blocks. The eNB, which directly connects to CN is called master eNB and the gNB, which is connected to the core via eNB using the X2 interface, is called secondary gNB. The number of UEs is defined using the numUe parameter of the UE module. In the "multi-cell with secondary gNB" case an extra set of eNB which are connected via X2, and an extra set of gNB which is in turn connected to the respective eNBs are shown in Fig. 15.

Fig. 14 – NED file for SingleCell_withSecondaryGnb

Fig. 15 – NED file for MultiCell_withSecondaryGnb

Table 1 – Parameters described in ini file for network simulation

Parameter	Value
eNodeB Transmission Power	40dB
Fb Period	10ms
Target BLER	0.01
BLER Shift	5
#Component Carriers	2
Carrier Frequency of CC1	2GHz
Carrier Frequency of CC2	6GHZ
#UE's	10
UE mobility type	"RandomWaypointMobility"
UE speed	Between 5mps to 15mps
Dual Connectivity	True
# resource blocks for CC1	6
# resource blocks for CC2	6
#UE apps	2
Amount of UDP application on the server (server.numApps)	#UE's * #UE apps =20

The configuration file (also known as an "ini" file) contains network parameters and their corresponding values for each carrier component as shown in Table 1. The number of UEs (numUe) specified in the UE module is set to 10 for this simulation. UE mobility type and UE speed are defined for each UE. Dual connectivity is enabled and each network is configured with uplink and downlink.

Carrier components are part of the carrier aggregation module and have carrier frequency and numerological index. The frequency of each carrier component is defined in Table 1 above. The number of resource blocks is also defined for each carrier component.

3.2 Results of the simulations

This section analyses the output of avgservingblock (average serving blocks are the resource blocks that are utilized at the time of simulation). The result files storing the simulated network's vector values and scalar values are analyzed after simulating the required network configuration. For example, avgservingblocks is a vector quantity because it

varies with the simulation time. Four outputs are analyzed for each network, corresponding to two configurations: uplink and downlink.

The four output results that are obtained include: average served blocks downlink for single-cell with secondary gNB (Fig. 16), average served blocks downlink for multi-cell with secondary gNB (Fig. 17), average served blocks uplink for single-cell with secondary gNB (Fig. 18), and average served blocks uplink for multi-cell with secondary gNB (Fig. 19). The simulation time is variable, and we use a value of 50 s, with resource allocation data being collected every millisecond for each of the four output results discussed above. This gives us enough data points to study the average served blocks for each output. The total number of resource blocks allocated in the results cannot exceed those that are set in the .ini file (specified across different CC, carrier components). The blue and orange coloured line chart represents the avg served blocks for master eNB and secondary gNB, respectively in singleCell_withSecondaryGnb, data flow is downlink in Fig. 16. In contrast, the blue, orange, green, and red coloured line chart represents the avg served blocks for master eNB1, secondary gNB1, master eNB2, and secondary gNB2, respectively, in MultiCell_withSecondaryGnb, and the data flow is downlink in Fig. 17. The blue and orange coloured line chart represents the avg served blocks for master eNB and secondary gNB, respectively, in singleCell_withSecondaryGnb, where the data flow is uplink in Fig. 18. The blue, orange, green and red coloured line chart represents the avg served blocks for master eNB1, secondary gNB1, master eNB2 and secondary gNB2, respectively, in MultiCell_withSecondaryGnb, where the data flow is uplink in Fig. 19.

This study shows that prioritized resource allocation can be simulated in different network topologies. The simulations enable us to study various configurations and analyze them to optimize the allocations. Representation of various configurations using text files defined in [21] enables us to easily create simulation topologies. Therefore, the SRC node (generating data corresponding to resource usage) and SINK node (applying various configurations in the form of NED files) are possible in the simulation environment. Integrated analysis of generated data using AI/ML is for future study.

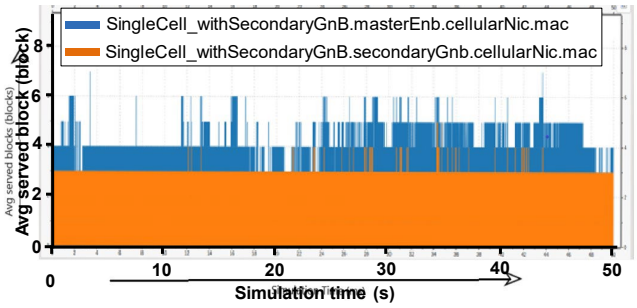


Fig. 16 – Avg served blocks, DL, SingleCell_withSecondaryGnb

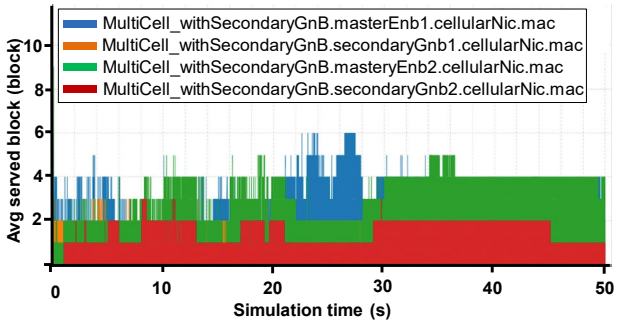


Fig. 17 – Avg served blocks, DL, multiCell_withSecondaryGnb

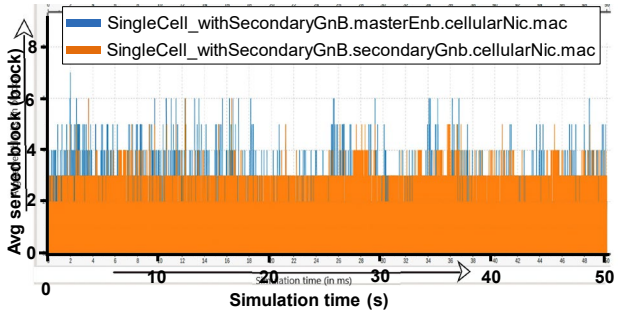


Fig. 18 – Avg served blocks, UL, singleCell_withSecondaryGnb

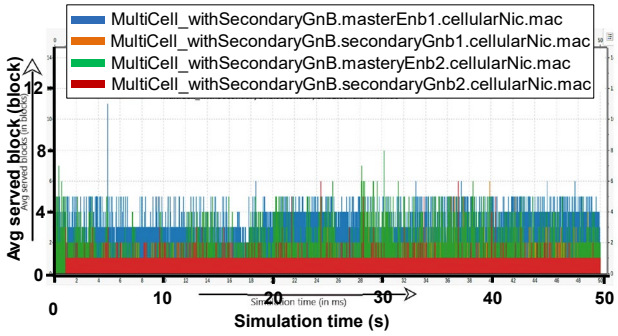


Fig. 19 – Avg served blocks. UL, MultiCell_withSecondaryGnb

4. ALGORITHMS INVESTIGATION FOR THE RESOURCE ALLOCATION IN THE "UNDERLAY"

This section describes the various algorithms that can be plugged into an O-RAN-based software architecture to verify the resource allocation schemes. The non-real-time RIC closed-loop intent is applied to the near real-time RIC lower loop.

The lower-loop monitors RAN resources and makes decisions to achieve the intent. Fig. 20 shows the illustration of the overall process of the system.

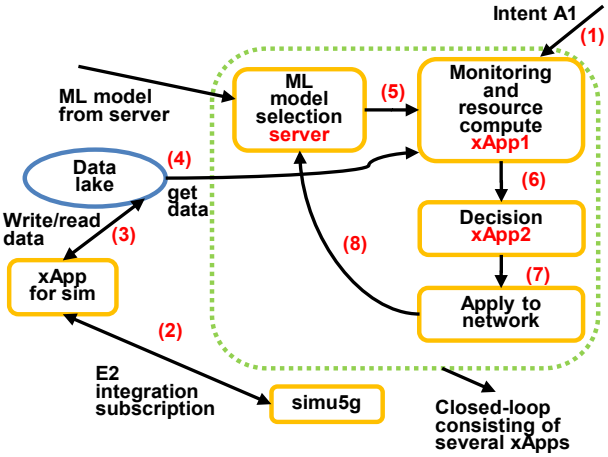


Fig. 20 – Block of closed-loop implementation for an emergency slice

Section 2.1 describes a closed-loop representation and design using a standard template and demonstrates it using a three-node closed-loop (i.e., SRC node, ML node and SINK node). Here, we further enhance this using a model selection service and a complete ML node implementation using two RIC xApps and demonstrate their deployment using docker containers.

ML model selection (server): Different ML models for inference can be available with different complexity and performance. We dynamically select different ML models from a server based on the declarative specification of the ML model, as described in Section 5. The models are implemented as a docker container and selection may be done either periodically or based on an external request. These ML models can be either specific to a particular problem or a general purpose one. The idea is that some ML algorithms might be too costly but can have a good prediction accuracy. On the other hand, there might be cheap ML algorithms with low-quality inference. Depending on the requirements, the best ML model can be selected.

Monitoring and resource compute (xApp 1): Advanced ML algorithms are applied for monitoring RAN resources (i.e., PRB, physical resource block utilization). This paper uses Gaussian Process Regression (GPR) as a non-parametric prediction technique. xApp1 reads data from a data lake either periodically or when needed. Then, it predicts how much resources will be available in the near future using this data. This information is used for other xApps to make resource allocation decisions.

Decision (xApp 2): After receiving the forecasted RAN resource in the near future, xApp 2 makes a resource allocation decision for the current and emergence slices depending on their SLA requirements.

This section designs and studies the closed-loop analysis and decision parts. Communication between xApps is provided through RIC Message Router (RMR) messaging used within the O-RAN software community. The workflow of the implementation shown in Fig. 20 is as follows:

(1) Get intent from a higher loop. It indicates if there is an emergency case and monitoring xapp is triggered.

(2) Subscribe to SRC to get the simulator/testbed data.

(3) Write data to the data lake to be used later for ML training.

(4) Data is sent to the ML node (implemented in xApp1) for model training and inference.

(5) Different ML models can be selected from the server here and sent to xApp1 for inference.

(6) xApp1: Resource monitoring such as PRB utilization. Here, the ML model is used, which can be fetched from our local repository. It also analyses whether there is an overutilization/ underutilization.

(7) Result obtained on (6) is sent to xApp2, which will make the final decision. It decides whether there is a need to allocate more resources on RAN for an emergency slice. Then it applies the decision to the real network (allocate more PRB for the emergency slice, E2 CONTROL).

4.1 The system implementation

A low-level closed-loop needs to be instantiated that monitors and computes RAN resources and makes a resource allocation decision for emergency cases based on the high-level intent.

The xApp1 monitors RAN resources and makes forecasts for the future PRB usage of the network. It also computes the available resources in the RAN domain. The forecasting and resource information is sent to xApp2, which is our decision xApp, through the RMR. RMR is developed by the O-RAN Software Community (SC), and we also utilize this messaging protocol in our implementation. xApp2 receives the necessary information from xApp1 and solves the problem P2 (or P1), which are given next, to find out the necessary PRB resources needed for

ES and make the resource allocation. The output of xApp2 will be sent to the real network to be applied through the E2 interface of O-RAN when the real integration starts.

We implement a docker container that acts as a web server where we keep different ML models to be used for monitoring or any other activities to test the model selection. model_handler.py implements an ML selection/pulling task in which we select ML dynamically depending on the performance of the current ML, thus enabling us to use another ML that may have a better performance than the current model. Fig. 21 shows the implementation details of this activity.

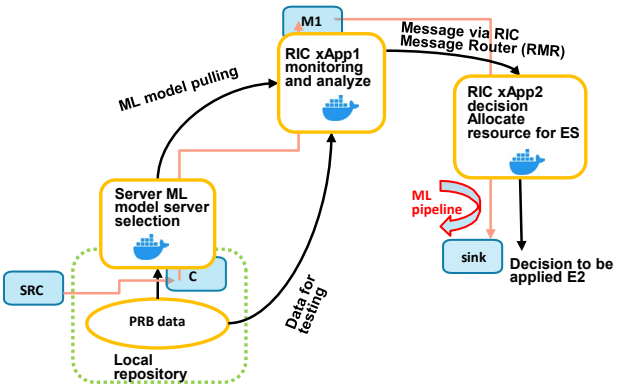


Fig. 21 – Workflow of the implementation

4.2 Results and analysis

4.2.1 Time-series forecasting of traffic for monitoring using Gaussian Process Regression

This section studies NS implementation in the network to have dedicated network resources over various domains. For example, the operator can allocate dedicated frequency resources (PRBs) to each slice at the RAN domain. Furthermore, different slices may have different SLA requirements on latency, bandwidth, reliability, etc. Even though each slice's need for resources can vary over time under dynamic networking conditions, the operator needs to ensure that the underlying infrastructure SLAs for each slice is guaranteed. For example, an operator can deploy a separate slice for video streaming. It needs to allocate additional resources to meet the SLA requirements on the slice during peak hours of the day. In case of an emergency, a new slice, Emergency Slice (ES), must be deployed by operators to handle the traffic in the emergency area, and the necessary amount of resources must also be allocated to the ES. In this study, NS with ES is a dynamic resource allocation problem in the RAN domain. When an emergency occurs, the ES is deployed and the resources needed for the ES are maintained autonomously.

To achieve autonomous resource management, traffic prediction of each slice is critical to gather information on the minimum amount of resources needed for the SLA requirements. It is complex to capture the dynamics through linear models due to the highly dynamic and non-linear patterns exhibited by wireless traffic. Artificial Neural Networks (ANNs), also known as deep neural networks or recurrent neural networks are commonly applied for traffic prediction. However, NN has well-known training challenges, and it is complicated to interpret the outcome of the NN prediction. Comparatively, Gaussian Process Regression (GPR) has continuously gained attention due to its interpretability and prediction accuracy. In addition, GPR can also provide information on the uncertainty of prediction, which is important when making resource allocation. In this study, PRB usage measurement is used to reflect the traffic characteristics and a time-series forecasting problem is formulated in which PRB utilization is predicted using GPR.

We study the use of GPR for the prediction of traffic. The PRB usage characterizes traffic which enables us to predict PRB utilization in the RAN domain. Real-world data from [53] in an urban area shows PRB utilization measured over a Long-Term Evolution (LTE) network for a user and collected and reported at every 500 ms. Fig. 22 shows 1000 samples of PRB utilization data.

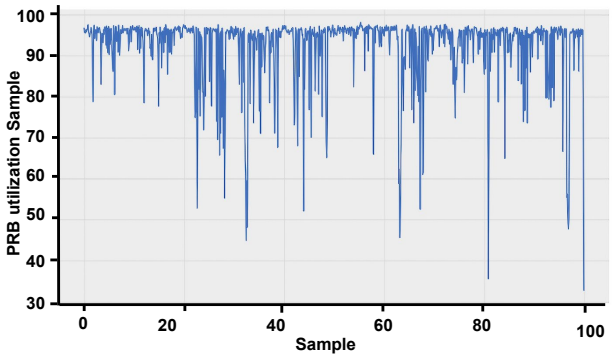


Fig. 22 – PRB utilisation over 1000 sample points

Understanding the characteristics of the data is important in selecting the best kernel for GPR. Periodic and varying data characteristics observed in Fig. 23 and Fig. 24 enables us to determine a good kernel for PRB prediction with GPR, representing both types of characteristics.

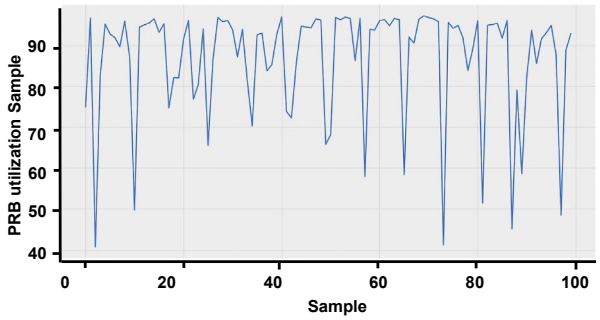


Fig. 23 – Periodic-type characteristics over a period of 100 time-step

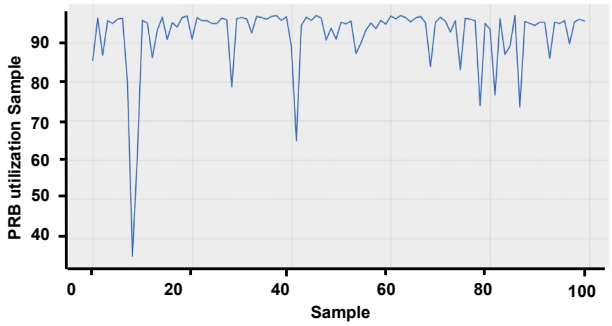


Fig. 24 – Constant-type characteristics over a period of 100 time- time

Forecasting with GPR

Fig. 25 shows PRB forecasting with GPR. The GPR model is trained with the last 100 samples and the chosen kernel as described above. We note that more data may need to be used for training depending on the application. Then, the trained GPR is used to make predictions for the future 50 samples. It can be concluded that the prediction with GPR is good enough to make efficient resource allocation proactively, as shown in Fig. 25. Note that GPR also provides information on the uncertainty of these predictions as we point to the upper bound when making predictions for the next 50 points. These upper bounds on the predictions can be utilized when making resource allocation to ensure that the correct amount of PRBs is allocated while satisfying the SLA requirements.

The PRB data is stored in a local repository. It is also possible to use different ML models for inference. An example implementation for O-RAN integration is implemented as a separate xApp and prediction_xapp.py creates a docker container for the inference implementation as a micro-service to be used for O-RAN.

We evaluate the performance of GPR prediction. We train with 1000 data points and evaluate the performance in terms of Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The MAE and RMSE for future 4000 points are 0.077 and 0.147 for entity reference of 0.046 and 0.081, respectively.

4.2.2 Resource allocation at RAN for an emergency slice

After the predicted traffic is obtained through GPR, the next step is to determine how many resources the ES should allocate. Therefore, we need to consider the SLAs of other slices in the network. The SLAs of other slices may degrade if we allocate more resources than the ES needs. The emergency case cannot be handled if we allocate fewer resources for the ES than it needs.

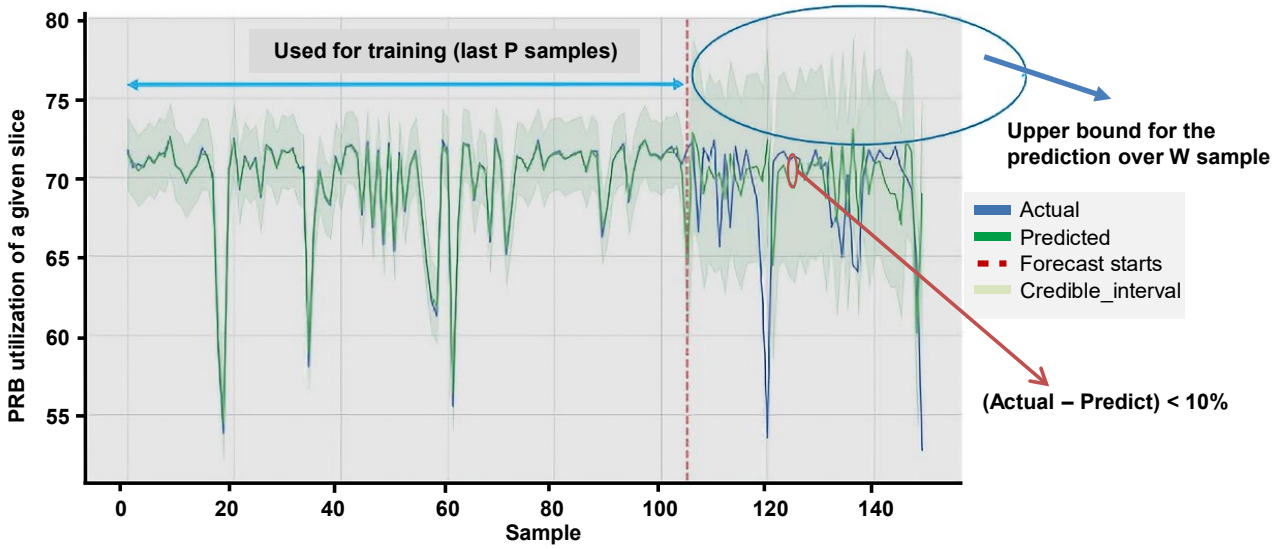


Fig. 25 – Time-series forecasting of PRB utilization with GPR

We assume that RAN resources are given in terms of either frequency resources or PRBs. Different slices allocated to different amounts of PRBs can be determined and fixed by the operator. However, it is not always efficient because not every slice is active all the time and uses its PRBs with 100% utilization although this strategy is good enough to have a dedicated network. This means that there can be some leftover PRBs that are not used by the corresponding slices [54], [55].

We consider two cases:

- Case-1: The ES does not have any dedicated PRB allocated, but it can only use the unused PRBs from other slices. The advantage of this strategy is it guarantees the SLAs of other slices, but ES can have significant degradation because it can only use the leftover PRBs, and after some time the leftover PRBs may not be large enough to support the emergency case.
- Case-2: We dynamically borrow PRBs from other slices to support the emergency case to minimize the degradation of the SLAs of other slices. In this strategy, priority is given to the ES and we guarantee that the emergency case is solved successfully while also minimizing the negative impact of borrowing PRBs on the SLAs of other slices.

We develop two algorithms to implement these two strategies:

Leftover GPR-based PRB allocation to ES algorithm (ALG1)

ALG 1 implements the first strategy in which only the leftover PRBs from other slices are allocated to the ES. The details of ALG1 are given in ALGORITHM 1.

To illustrate the operations of ALG1, let us consider two slices and the allocated PRBS to these slices: T1 = 40 PRBs and T2 = 60 PRBs and in total, the system has T = 100 PRBs. Let us also assume that the PRB utilization of these slices is 80% and 90%, respectively. That means the first slice uses only 40*0.8=32 PRBs and the second slice uses only 60*0.9=54 PRBs. Hence, 40-32= 8 PRBs from the first slice and 60-54=6 PRBs from the second slice (in total 14 PRBs) can be allocated to the ES with ALG1 for this example.

Priority GPR-based PRB allocation to ES algorithm (ALG2)

ALGORITHM 1: *Leftover GPR-based PRB allocation to ES algorithm (ALG1)*

Input: T = total available PRBs of the system (i.e., For LTE, 100 PRBs).
W = Prediction window (i.e., next prediction time, 500 ms).
P = Past training window (the last 100 samples).
o = Compensation.
N = number of slices in the network.
T_n = Amount of PRBs allocated to slice n.
D_n = PRB utilization time-series data for each slice.

Output: Allocate PRBs to ES: P_{ES}

For *each other slice* n
  ***Step 1*:** Train GPR with the latest P training data.
  ***Step 2*:** Forecast PRB utilization over the next W samples with GPR : U_n.
  ***Step 3*:** Calculate maximum possible PRB utilization using upper bound: $C_n = U_n + o_n$.
  ***Step 4*:** Calculate the forecasted PRB usage of all other slices over next W samples : $B_n = T_n C_n$
End

***Step 5*:** Calculate available PRBs for Emergency Slice:

$$P_{ES} = T - \sum_{n=1}^{N} T_n C_n$$

***Step 6*:** Allocate PRBs to ES: P_{ES}

ALG2 implements the second strategy in which we borrow PRBs from the other slice while minimizing the negative impact on both of them. We assume the ES needs an E amount of PRBs. First, we allocate the available leftover PRBs to the ES. If it is not enough, we borrow PRBs from other slices by minimizing their performance degradation. The details of ALG2 are given in ALGORITHM 2.

PRBs are borrowed from other slices to meet the requirement of ES and also minimize the resource shortage of other slices. Thus, the P1 optimization problem is formulated in ALG 2

Since it involves a non-linear operation with a max [56] operator, the problem is difficult to solve. However, we use an auxiliary trick and convert this problem to an easily solvable integer program. This problem is transformed into a solvable integer problem using the auxiliary variable u_n as shown P2 in ALG 2.

The importance of P2 is to decide how many PRBs are to be taken from each of the other slices and allocated to the ES. These two algorithms are implemented, and decision_xapp.py creates a docker file to run this implementation as a microservice to be ready for use in the O-RAN platform.

ALGORITHM 2: *Priority GPR-based PRB allocation to ES algorithm (ALG 2)*

Input: T = Total available PRBs of the system.
T_n = Amount of PRBs allocated to slice n.
W = Prediction window.
P = Past training window.
o = Compensation.
N = number of slices in the network.
D_n = PRB utilization time-series data for each slice
E = amount of PRBs needed for an emergency slice

Output: $x_n(t)$ = amount of PRBs needed for slice n at time t
$y_n(t)$ = amount of PRB taken from slice n at time t

P1:

$$min \sum_{t=1}^{W} \sum_{n}^{N} max\ \{0, x_n(t) - (T_n - y_n(t))\}$$

$$\text{s.t. } 0 \le y_n(t) \le T_n\ \forall n$$
$$\sum_{n}^{N} y_n(t) \ge E$$

– $x_n(t)$: PRB usage for slice n at time t

– T_n : Total PRBs given to slice n

– $y_n(t)$: Number of PRBs taken from slice n at time t to be used for emergency slice

P2:

$$\min \sum_{t-1}^{W} \sum_{n}^{N} u_n(t)$$

$$\tilde{x}_n(t) + o_n(t) - \left(T_n - y_n(t)\right) \le u_n(t)$$

$$0 \le y_n(t) \le T_n\ \forall n$$

$$\sum_{n}^{N} y_n(t) \ge E$$

$$u_n(t) \ge 0$$

where $x_n(t) = \tilde{x}_n(t) + o_n(t)$

$x_n(t)$: Actual PRB usage at time t in future. This cannot be known in advance.

$\tilde{x}_n(t)$: Estimated PRB usage with GPR

$o_n(t)$: Estimation error. Upper bound provided by GPR can be used.

In a simulation scenario, we assumed that we have two slices with different PRB requirements:

- T1: number of PRBs assigned to Slice 1 by the operator (e.g., T1 = 40)
- T2: number of PRBs assigned to Slice 2 by the operator(e.g., T2 = 60)
- T: total number of PRBs in the system (e.g., T = 100 PRBs)
- Time series PRB utilization for each slice, in percentage, with a granularity of 100ms and 200 ms

The performance of ALG2 was studied under the scenario that there are two other slices and T1 = 40 and T2 = 60 PRBs allocated to them. By applying ALG2, we borrow PRBs to satisfy the requirement of the ES when those slices do not need the resources. We assume the ES needs 20 PRBs. Fig. 26 shows the number of PRBs taken from other slices over 45 time instants. Depending on the predicted PRB usage of other slices, ALG2 takes 11 or 12 PRBs and 8 or 9 PRBs from the first and second slices, respectively, and 20 PRBs in total are ready to be used by the ES simultaneously.

This section studies NS implementation in the network to have dynamic resource allocation over various domains. To analyze the PRB utilization in RAN, two algorithms were studied. Results are presented considering the need for resources of each slice which can vary over time under dynamic networking conditions. The results show the importance of closed-loop implementations in NS, especially for intelligent management of RAN resources during emergency scenarios.

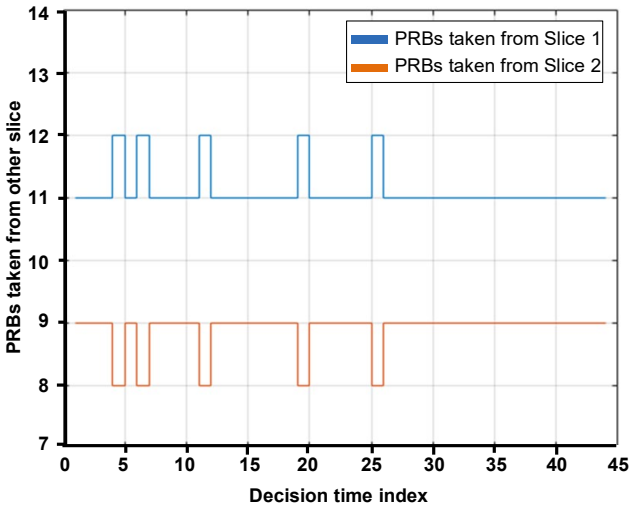


Fig. 26 – PRB allocation for ES

5. O-RAN CONTROL-LOOP INSTANTIATION

This section describes the implementation of the algorithms described in Section 4 in an O-RAN near Real-Time RAN Intelligence Controller (RIC) [57] and its integration with the Acumos [58] model repository. The model description is included in the declarative specification of closed-loop as discussed in Section 2.1. In this study, a pretrained model is fetched from Acumos based on the given description and deployed as xApp [59] in the O-RAN platform (See Fig. 27 for details).

5.1 A solution workflow

The following workflow is used for the implementation:

- RAN (E2-SIM [60] is used) is registered and associated with O-RAN near RT RIC.
- RIC receives policy updates from A1 for triggering closed-loop PRB allocation.
- An ML model is fetched based on the A1 policy details.
- PRB utilization is predicted based on the analysis of test data used instead of actual data from E2.
- The PRB to be allocated is computed and an E2 control message is sent based on the inference. PRBs are always reserved for the emergency slice and additional resources can be reallocated based on situational considerations.
- The allocation decision is continuously monitored, evaluated, and improved upon.

The workflow steps are further explained in Fig. 27 and discussed as follows:

- Points 1 and 2 show that E2 SIM is up and that association with RIC is set up.
- Point 3 shows the nRT RIC receives the A1 policy update to trigger closed-loop monitoring.
- Point 4 shows the A1-mediator sends A1 Policy REQ to the "*prbpred*" xApp.
- Points 5a and 5b show the model is fetched from the model store as per policy guidelines and "*prbpred*" instructs DataMon/Alloc xApp to start monitoring the data.
- Points 6,7, and 8 show the messaging done for subscribing to E2 for data.

E2 Indication is for future reference; currently, data is not monitored through RIC indication. In future, data needs to be monitored and sent to predict xApp.

- Point 9 shows data reception from the E2 node. The received metrics are stored in metrics DB as in Point 10.
- Upon timer expiry as in Point 11, a request for prediction is sent to "prbpred" xApp as in Point 12.
- "*prbpred*" uses the ML model to predict future utilization. Retraining may be done based on the new data model. The predicted values may be sent to DataMon/Alloc xApp as in Point 13 and Point 14.
- DataMon/Alloc xApp computes the PRB to be allocated and sends the E2 control message towards E2 as in Point 15.

5.2 Resulting implementation

This section presents the implementation of the algorithms in Section 4. The algorithms can be instantiated in the O-RAN-RIC platform and prediction based on the xApp onboarding/deployment process and RIC platform components can be achieved.

The xApps are developed based on the xApp Framework for Python. Separate xApp-descriptor files were defined detailing the configuration, RX & TX messages supported:

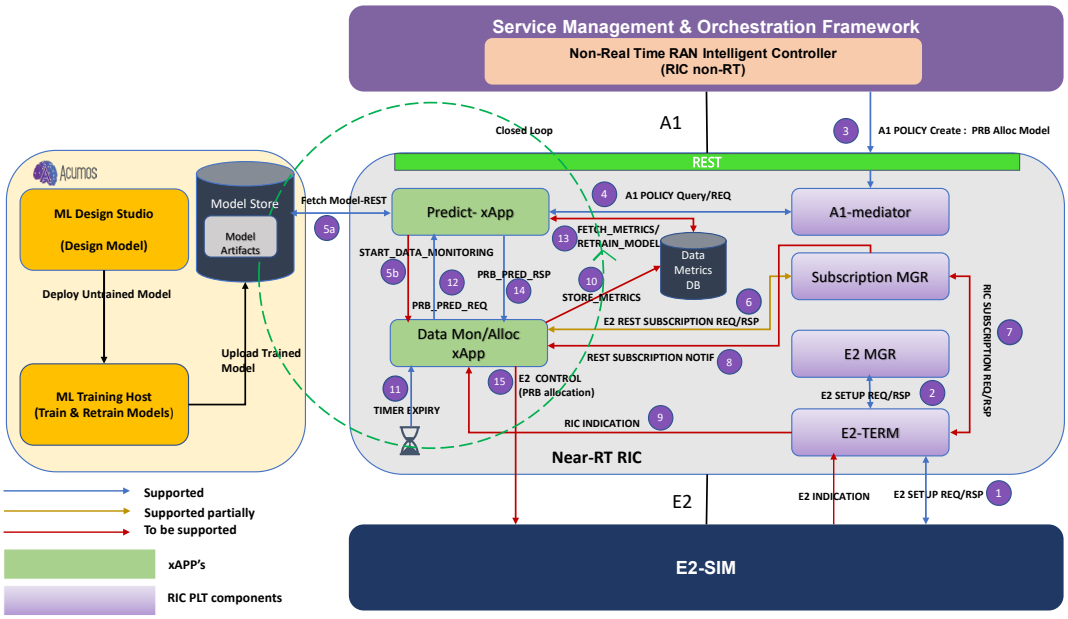
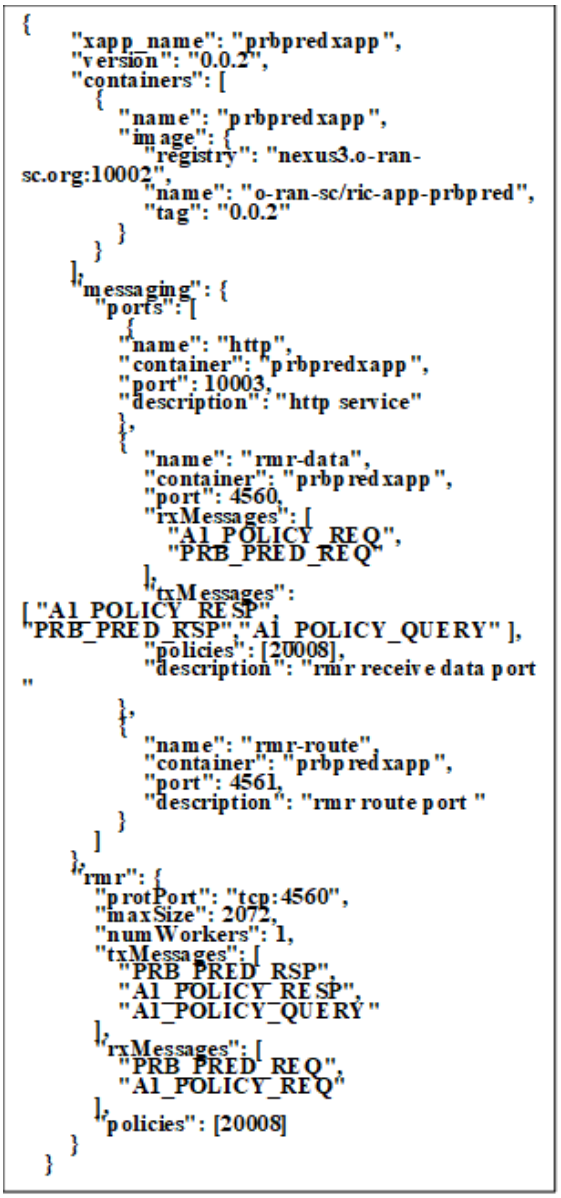
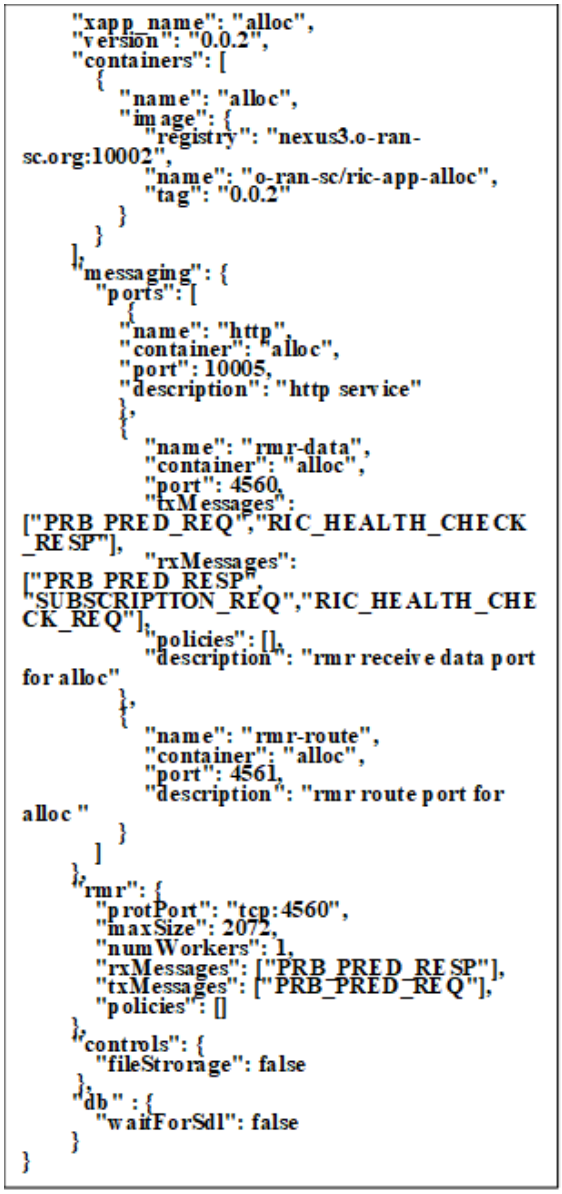


Fig. 27 – Modified xApp process model



```
{
    "xapp_name": "prbpredxapp",
    "version": "0.0.2",
    "containers": [
      {
        "name": "prbpredxapp",
        "image": {
          "registry": "nexus3.o-ran-
sc.org:10002",
          "name": "o-ran-sc/ric-app-prbpred",
          "tag": "0.0.2"
        }
      }
    ],
    "messaging": {
      "ports": [
        {
        "name": "http",
        "container": "prbpredxapp",
        "port": 10003,
        "description": "http service"
        },
        {
          "name": "rmr-data",
          "container": "prbpredxapp",
          "port": 4560,
          "rxMessages": [
            "A1_POLICY_REQ",
            "PRB_PRED_REQ"
          ],
          "txMessages":
[ "A1_POLICY_RESP"
"PRB_PRED_RSP","A1_POLICY_QUERY" ],
          "policies": [20008],
          "description": "rmr receive data port
"
        },
        {
          "name": "rmr-route",
          "container": "prbpredxapp",
          "port": 4561,
          "description": "rmr route port "
        }
      ]
    },
    "rmr": {
      "protPort": "tcp:4560",
      "maxSize": 2072,
      "numWorkers": 1,
      "txMessages": [
        "PRB_PRED_RSP",
        "A1_POLICY_RESP",
        "A1_POLICY_QUERY"
      ],
      "rxMessages": [
        "PRB_PRED_REQ",
        "A1_POLICY_REQ"
      ],
      "policies": [20008]
    }
  }
```


Fig. 28 – xApp-descriptor file for prbpred xApp



```
    "xapp_name": "alloc",
    "version": "0.0.2",
    "containers": [
      {
        "name": "alloc",
        "image": {
          "registry": "nexus3.o-ran-
sc.org:10002",
          "name": "o-ran-sc/ric-app-alloc",
          "tag": "0.0.2"
        }
      }
    ],
    "messaging": {
      "ports": [
        {
        "name": "http",
        "container": "alloc",
        "port": 10005,
        "description": "http service"
        },
        {
          "name": "rmr-data",
          "container": "alloc",
          "port": 4560,
          "txMessages":
["PRB_PRED_REQ","RIC_HEALTH_CHECK
_RESP"],
          "rxMessages":
["PRB_PRED_RESP",
"SUBSCRIPTION_REQ","RIC_HEALTH_CHE
CK_REQ"],
          "policies": [],
          "description": "rmr receive data port
for alloc"
        },
        {
          "name": "rmr-route",
          "container": "alloc",
          "port": 4561,
          "description": "rmr route port for
alloc "
        }
      ]
    },
    "rmr": {
      "protPort": "tcp:4560",
      "maxSize": 2072,
      "numWorkers": 1,
      "rxMessages": ["PRB_PRED_RESP"],
      "txMessages": ["PRB_PRED_REQ"],
      "policies": []
    },
    "controls": {
      "fileStrorage": false
    },
    "db" : {
      "waitForSdl": false
    }
}
```


Fig. 29 – xApp-descriptor file for allocator xApp

- *prbpred* xApp: Initially, this xApp registers for PRB_PRED_REQ (PRB Prediction Request) and A1_POLICY_REQ (A1 Policy Request), and queries A1-mediator to get the policy details. A specific policy was created which gives model information and model version information to be used. This xApp is responsible for receiving *A1_POLICY_REQ* and saving the policy details. Fetch the model from the *modelStore* and save it. Predict the future PRB utilization and respond to *alloc* xApp for further processing based on the timer trigger. Upon reception of *PRB_PRED_REQ*, the xApp predicts PRB utilization for each slice and sends a response to *Alloc* xApp based on the model fetched. Fig. 28 shows the xApp-descriptor file for the *prbpred* xApp.

- Allocator xApp: Initially, this xApp registers with the subscription manager for E2 information and starts a timer to trigger PRB_PRED_REQ periodically. PRB is allocated for an emergency slice based on predicted future PRB utilization. A simple algorithm for PRB allocation in Section 4 is used here. In addition, some PRBs are shown as reserved for emergency/high priority events.

The assumption taken is the total number of PRBs in the system is 100. Slice #1 and Slice#2 were configured with 35 PRBs each. 30 PRBs were reserved for emergency/high priority events.

The actual Value of PRB utilized is computed based on the predicted PRB utilization received for each slice.

Utilised_PRB_slice1=PRB_ALLOC_SLICE1*(slice1_utilisation/100)

Utilised_PRB_slice2=PRB_ALLOC_SLICE2*(slice2_utilisation/100)

total_prb_avail = Total_PRB – (Utilised_PRB_slice1 + Utilised_PRB_slice2)

The reserved PRBs are also made available because this is an emergency event. Alloc xApp sends the E2 control message to allocate the available PRBs from the calculation. Fig. 29 shows the xApp-descriptor file for allocator xApp.

Successful communication between the xApp and other RIC platform components was achieved as part of this. A model store was developed to mimic Acumos and have access to the pretrained model. E2 SIM setup was registered with the E2 component in the RIC platform.

In the Dawn release, the creation of the A1 policy instance doesn't trigger the A1 policy to send a message towards the xApp [61]. The workflow was modified to send a timer-based event from alloc XApp to trigger PRB prediction. When the policy instance is created (CREATE/UPDATE messages are sent to xApp by A1 mediator), the prbpred xApp can store the model information and perform prediction based on the trigger.

6. INTEGRATION OF THE POC

This section describes the integration of the above implementation of closed-loops into O-RAN-based software platform ready to be tested in the 5G Berlin testbed [62]. The operator inputs the declarative intent to the Service Management Orchestrator (SMO)/Non-RT RIC, which describes the use case to detect emergencies and maintain the required SLA as described in Section 2.1. Similar to the mechanism described in 2.1, SMO/Non-RT RIC then creates a higher loop that monitors various parameters like network activities, input from emergency responders (ER), social media trends, etc. to detect and locate the emergency (e.g., fire in a building). This can be realized using either a hosted model in Acumos or Open Network Automation Platform data collection analytics engine (ONAP DCAE) or O-RAN rApp, as discussed in Sections 2.2 and 4. Once the emergency is detected, the higher loop sends an intent over the A1 interface to the Near-RT RIC, instructing it to handle the increased load for the corresponding RAN node. Real-time ML/AI inference might be needed by some of the ERs' devices; for firefighters a helmet-mounted camera may use image recognition to detect humans in a burning building. However, the devices might not have enough computing and might need to offload the task to the network edge or use split AI/ML models for inference. The Near-RT RIC receives the intent and creates a closed-loop which can monitor the network and compute resources of the edge and the ER device and maintains the SLA/QoS (quality of service) of the inference task as discussed in Section 3 above. This loop can be realized using xApp. Fig. 30 shows the simulator-based sequence for the integration of the activities.

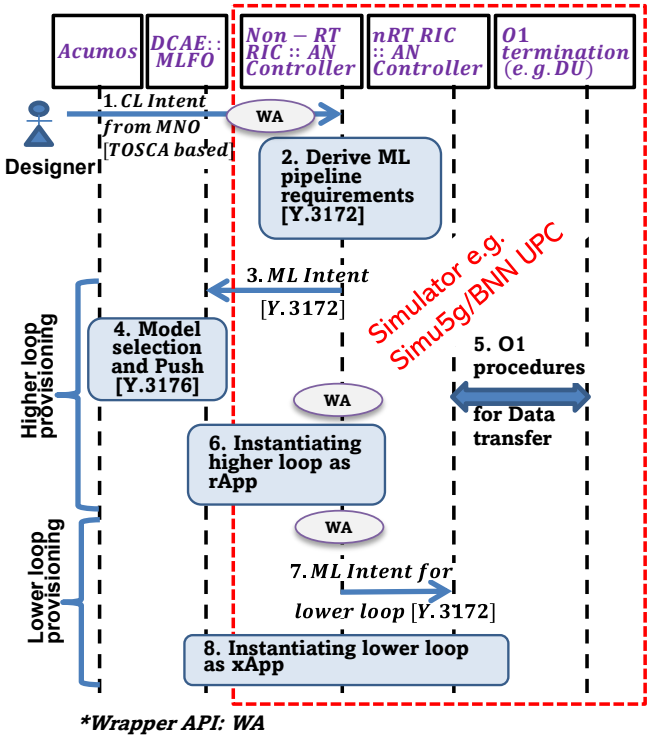


Fig. 30 – Simulator-based sequence for the integration of the activities.

Fig. 31 shows the extensions to the sections above to integrate the implementation of the algorithms described in Section 4 in an O-RAN- RIC and its integration with the Acumos [58] model repository. The first addition was the A1 poller which pulls the A1 mediator at regular intervals and converts the A1 policy to a TOSCA template described in Section 2.1. It uses HTTP-based interfaces to communicate with the A1 mediator and the orchestrator. The dms_cli tool provided by the O-RAN-SC was used to enable the orchestrator to orchestrate the xapps as specified in the A1 policy. Playbooks (workflows) described in Section 2.1 were updated to integrate relevant command line (dms_cli) commands. These commands are used to onboard and install corresponding xApps.

The overall flow of the final integrated solution (see Fig. 31) is as follows:

1. The human user or a higher loop applies an A1 policy to the near-RT RIC. This policy is received by the A1 mediator.

2. The A1 poller gets the policy, translates it into the TOSCA template and sends it to the orchestrator.

3. The orchestrator manages the RIC xapps according to the TOSCA template using dms_cli.

4. The newly orchestrated xapps pull the necessary models from the model repository server.

5. *Pred* xapp makes a time-series prediction for future traffic in the network and how much resources (PRB) will be available for an emergency slice in the near future.

6. *Alloc* xapp sends an RMR request to *pred* to get the prediction and allocates PRBs to the emergency slice based on that.

7. *Alloc* xapp then sends a message over the E2 interface to the RAN. Slice allocation messages are verified from the console.

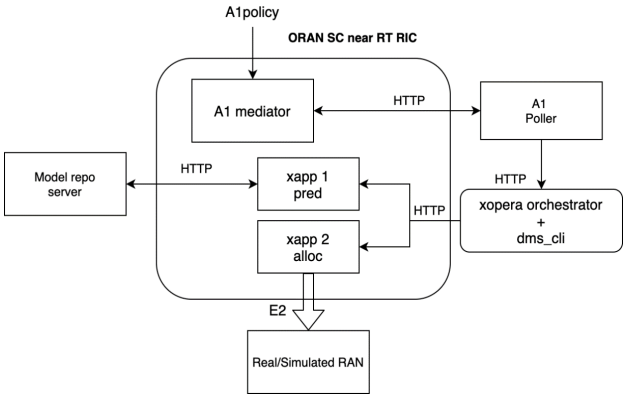


Fig. 31 – Overall flow of the final integrated PoC

7. OBSERVATIONS FROM THE POC

Abstraction of nodes allows the service template to select concrete nodes that best match the requirements of the abstract nodes during deployment. The concrete nodes can be provided in a repository known to the orchestrator. Abstract requirements can be achieved in TOSCA YAML using the *node_filter* feature. However, this study found that abstraction features like *node_filter* and *substitution* are not supported by certain implementations of orchestrators. A feature-based comparison of orchestrators concerning TOSCA compliance may be made as part of future studies.

Besides the RAN slicing experiments exploring a closed-loop using the FlexRAN controller, other AI applications can interact with the SDN controller and the VNF placement functions to attend to different network requirements. An End-to-End (E2E) network slice cannot be completely implemented in the testbed because a MANO implementation was not used to avoid computational costs and network complexity in this first phase and focus on AI integration. Future studies may include integrating the software developed in our testbed into ONAP software, a popular MANO implementation, to provide E2E NS with a centralized closed-loop. The verification and validation of resource allocation during simulation in line with the traffic pattern (e.g., full buffer) when simulating the scenarios, e.g., dual connectivity, is an essential future step as we broaden the simulation into more scenarios.

Creating a closed-loop with several modules brings communication and computation problems. Overall integration, including A1/O1/E1 interface integrations, is critical and which parts of this integration can be realized autonomously can be explored. The real-time system performance will have to be tested to ensure compliance with closed-loop specifications. Integration issues with platforms highlight the importance of close coordination with underlays, as mentioned in Sections 2.2 and 5.

8. CONCLUSION AND FUTURE RESEARCH

This is a collaborative study where we developed and implemented a hierarchical closed-loop that autonomously handles an emergency case. The study focused on intent parsing, traffic monitoring, resource computing, and allocation autonomously. The closed-loops were implemented with several micro-services deployed as docker containers with specific functions such as monitoring, computing, ML selection, and resource allocation. Future activities will focus on enhancing the attributes of the nodes in the template, e.g., data parameters in the SRC [e.g., 3xVs: velocity, variety and volume], Model metadata (as defined in ITU-T Y.3176), and SINK parameters [e.g., underlay specific APIs]. After integration, data pulling, model pulling, and adaptation can be demonstrated based on such enhanced attributes. The machine learning agent presented in the Connected AI study will be implemented for the built environment with ONAP and Acumos integration for future activities. Enhancing the simulator to include inputs from an

intent and integration with the SRC, ML and SINK nodes to form the closed-loop in the simulation domain is also an important future step. Apart from advanced algorithms studied, e.g., multivariant time-series models with monitored data and arriving at intelligent inference, resource reservation for emergencies and resource reallocation from lower priority services should be explored. Easy onboarding of xApps and the triggering of policy towards lower closed-loops and supporting visualizations can increase usability. In addition, extending the solution to self-learning closed-loops with continuous collection, analytics, decision and actuation and model performance detection needs further study. With the self-learning close-loops, the network could trigger a switchover to another better performing model, analyze and trigger a different set of data/measurements for data analysis and perform synchronization and management across the edge and emergency responder devices.

ACKNOWLEDGEMENT

ITU FG-AN, the International Telecommunication Union Focus Group on Autonomous Networks, organized a “build-a-thon challenge” in 2021 to demonstrate and validate important use cases for autonomous networks, creating Proof of Concept (PoC) implementations and tools in the process. The majority of the works in this study were done under the Build-a-thon Challenge.

REFERENCES

[1] E. Selerio Jr, J. A. Caladcad, M. R. Catamco, E. M. Capinpin, and L. Ocampo, "Emergency preparedness during the COVID-19 pandemic: Modelling the roles of social media with fuzzy DEMATEL and analytic network process", *Socio-economic planning sciences,* p. 101217, 2021.

[2] K. Carlberg, E. W. Burger, and R. P. Jover, "Dynamic 5G Network Slicing for First Responders", in *2019 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 2019: IEEE, pp. 1-4.

[3] J. Gallego-Madrid, R. Sanchez-Iborra, P. M. Ruiz, and A. F. Skarmeta, "Machine learning-based zero-touch network and service management: A survey", *Digital Communications and Networks,* 2021.

[4] V. Sciancalepore, F. Z. Yousaf, and X. Costa-Perez, "z-TORCH: An automated NFV orchestration and monitoring solution", *IEEE Transactions on Network and Service Management,* vol. 15, no. 4, pp. 1292-1306, 2018.

[5] R. Wen *et al.*, "On robustness of network slicing for next-generation mobile networks", *IEEE Transactions on Communications,* vol. 67, no. 1, pp. 430-444, 2018.

[6] Q. Wang *et al.*, "Enable advanced QoS-aware network slicing in 5G networks for slice-based media use cases", *IEEE transactions on broadcasting,* vol. 65, no. 2, pp. 444-453, 2019.

[7] P. H. Gomes, M. Buhrgard, J. Harmatos, S. K. Mohalik, D. Roeland, and J. Niemöller, "Intent-driven closed loops for autonomous networks", *Journal of ICT Standardization,* pp. 257–290-257–290, 2021.

[8] S. Singh, P. K. Sharma, B. Yoon, M. Shojafar, G. H. Cho, and I.-H. Ra, "Convergence of blockchain and artificial intelligence in IoT network for the sustainable smart city", *Sustainable Cities and Society,* vol. 63, p. 102364, 2020.

[9] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks", *IEEE Communications Magazine,* vol. 59, no. 10, pp. 21-27, 2021.

[10] S. Sultana and A. Mittermaier. "Build-a-Thon (PoC) – 5G Berlin Test Network Functional Specification", https://extranet.itu.int/sites/itu-t/focusgroups/an/input/FGAN-I-093.zip (accessed 07/06, 2022).

[11] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges", *arXiv preprint arXiv:2202.01032,* 2022.

[12] L. Gavrilovska, V. Rakovic, and D. Denkovski, "From cloud RAN to open RAN", *Wireless Personal Communications,* vol. 113, no. 3, pp. 1523-1539, 2020.

[13] G. ETSI, "Zero-touch network and Service Management (ZSM); Reference Architecture", Tech. Rep, 2019.

[14] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, "Knowledge-based 5G radio access network planning and optimization", in *2016 International Symposium on Wireless Communication Systems (ISWCS)*, 2016: IEEE, pp. 359-365.

[15] X. You, C. Zhang, X. Tan, S. Jin, and H. Wu, "AI for 5G: research directions and paradigms", *Science China Information Sciences,* vol. 62, no. 2, pp. 1-13, 2019.

[16] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial", *IEEE Communications Surveys & Tutorials,* 2021.

[17] X. Wang, X. Li, and V. C. Leung, "Artificial intelligence-based techniques for emerging heterogeneous network: State of the arts, opportunities, and challenges", *IEEE Access,* vol. 3, pp. 1379-1391, 2015.

[18] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey", *IEEE Communications Surveys & Tutorials,* vol. 22, no. 3, pp. 2031-2063, 2020.

[19] A. Bardou, T. Begin, and A. Busson, "Improving the spatial reuse in ieee 802.11 ax wlans: A multi-armed bandit approach", in *Proceedings of the 24th International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2021, pp. 135-144.

[20] M. Matracia, N. Saeed, M. A. Kishk, and M.-S. Alouini, "Post-Disaster Communications: Enabling Technologies, Architectures, and Open Challenges", *arXiv preprint arXiv:2203.13621,* 2022.

[21] F. Liu, Y. Guo, Z. Cai, N. Xiao, and Z. Zhao, "Edge-enabled disaster rescue: a case study of searching for missing people", *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 10, no. 6, pp. 1-21, 2019.

[22] R. F. Hussain, M. A. Salehi, A. Kovalenko, Y. Feng, and O. Semiari, "Federated edge computing for disaster management in remote smart oil fields", in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019: IEEE, pp. 929-936.

[23] J. Pérez-Romero *et al.*, "Supporting mission critical services through radio access network slicing", in *2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, 2019: IEEE, pp. 1-8.

[24] R. Ferrús, O. Sallent, J. Pérez-Romero, and R. Agusti, "On the automation of RAN slicing provisioning and cell planning in NG-RAN", in *2018 European Conference on Networks and Communications (EuCNC)*, 2018: IEEE, pp. 37-42.

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

[26] Ł. Kułacz and A. Kliks, "Dynamic Spectrum Allocation Using Multi-Source Context Information in OpenRAN Networks", *Sensors,* vol. 22, no. 9, p. 3515, 2022.

[27] M. Dryjański, Ł. Kułacz, and A. Kliks, "Toward Modular and Flexible Open RAN Implementations in 6G Networks: Traffic Steering Use Case and O-RAN xApps", *Sensors,* vol. 21, no. 24, p. 8173, 2021.

[28] H. Kumar, V. Sapru, and S. K. Jaisawal, "O-RAN based proactive ANR optimization", in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020: IEEE, pp. 1-4.

[29] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis, "Simu5G–An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks", *IEEE Access,* vol. 8, pp. 181176-181191, 2020.

[30] G. Pereyra, C. Rattaro, and P. Belzarena, "Py5cheSim: a 5G Multi-Slice Cell Capacity Simulator", in *2021 XLVII Latin American Computing Conference (CLEI)*, 2021: IEEE, pp. 1-8.

[31] 5G-LENA Team. "5G-LENA simulator", https://5g-lena.cttc.es/ (accessed 05/13, 2022).

[32] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks", in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427-441.

[33] SD-RAN. "Introduction – SD-RAN Docs 1.1.0 documentation", https://docs.sd-ran.org/sdran-1.1/introduction.html (accessed 5/13, 2022).

[34] H. Lee, J. Cha, D. Kwon, M. Jeong, and I. Park, "Hosting ai/ml workflows on o-ran ric platform", in *2020 IEEE Globecom Workshops (GC Wkshps*, 2020: IEEE, pp. 1-6.

[35] O-RAN. "O-RAN Architecture Overview", https://docs.o-ran-sc.org/en/latest/architecture/architecture.html (accessed 03/09, 2022).

[36] ITU - T. "Framework for evaluating intelligence levels of future networks including IMT-2020", https://www.itu.int/rec/T-REC-Y.3173 (accessed 30/05, 2020).

[37] A. Luzar, S. Stanovnik, and M. Cankar, "Examination and Comparison of TOSCA Orchestration Tools", in *European Conference on Software Architecture*, 2020: Springer, pp. 247-259.

[38] V. Ram O.V *et al.* "Proposal for a "Build-a-thon" for ITU AI/ML in 5G Challenge (second edition, 2021), aligned with FGAN WG3", ITU Focus Group on Autonomous Network (FG-AN). https://extranet.itu.int/sites/itu-t/focusgroups/an/input/FGAN-I-170-R1.docx (accessed 02/23, 2022).

[39] *OASIS TOSCA Simple Profile in YAML v1.3*, xopera, 2021. [Online]. Available: https://xlab-si.github.io/xopera-docs/

[40] ITU - T. "Machine learning marketplace integration in future networks including IMT-2020", https://www.itu.int/rec/T-REC-Y.3176 (accessed 30/05, 2022).

[41] C. V. Nahum *et al.*, "Testbed for 5G connected artificial intelligence on virtualized networks", *IEEE Access,* vol. 8, pp. 223202-223213, 2020.

[42] ITU - T. "Architectural framework for machine learning in future networks including IMT - 2020", https://www.itu.int/rec/T-REC-Y.3172-201906-i/en (accessed 03/12, 2022).

[43] I. Aliyu, M. C. Feliciano, S. Van Engelenburg, D. O. Kim, and C. G. Lim, "A blockchain-based federated forest for SDN-enabled in-vehicle network intrusion detection system", *IEEE Access,* vol. 9, pp. 102593-102608, 2021.

[44] Ryu. "Ryu API Reference", https://ryu.readthedocs.io/en/latest/api_ref.html (accessed 01/25, 2022).

[45] Flexran. "Mosaic5G", https://mosaic5g.io/flexran/ (accessed 01/25, 2022).

[46] *Empowering App Development for Developers | Docker*, Docker. [Online]. Available: https://www.docker.com/

[47] O. OpenAirInterface. Accessed: Sep. 13. "5G Software Alliance for Democratising Wireless Innovation", http://www.openairinterface.org/ (accessed 01/25, 2022).

[48] Free5GC. "Free5GC: Open-Source 5GC", https://www.free5gc.org/ (accessed 01/25, 2022).

[49] Kubernetes. "Kubernetes", (accessed 01/25, 2022).

[50] 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.

[51] E. Bisong, "*Kubeflow and kubeflow pipelines*", *in Building Machine Learning and Deep Learning Models on Google Cloud Platform.* . Berkeley, CA, USA: Springer, 2019.

[52] S. Lins *et al.*, "Artificial Intelligence for Enhanced Mobility and 5G Connectivity in UAV-Based Critical Missions", *IEEE Access,* vol. 9, pp. 111792-111801, 2021.

[53] V. Raida, P. Svoboda, and M. Rupp, "Real World Performance of LTE Downlink in a Static Dense Urban Scenario-An Open Dataset", in *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 2020: IEEE, pp. 1-6.

[54] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: RAN slicing for a flexible and cost-effective multi-service mobile network architecture", in *Proceedings of the 23rd annual international conference on mobile computing and networking*, 2017, pp. 127-140.

[55] A. Okic, L. Zanzi, V. Sciancalepore, A. Redondi, and X. Costa-Pérez, "π-ROAD: A learn-as-you-go framework for on-demand emergency slices in V2X scenarios", in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021: IEEE, pp. 1-10.

[56] *TOSCA Simple Profile in YAML Version 1.3*, C. L. OASIS Committee Specification 01..Edited by Matt Rutkowski, Claude Noshpitz, 2019. [Online]. Available: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/TOSCA-SimpleProfile-YAML-v1.3.html

[57] K. Kristiansen. "Near Realtime RIC ", https://wiki.o-ran-sc.org/display/GS/Near+Realtime+RIC+Installation (accessed 03/09, 2022).

[58] S. Zhao, M. Talasila, G. Jacobson, C. Borcea, S. A. Aftab, and J. F. Murray, "Packaging and sharing machine learning models via the acumos ai open platform", in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018: IEEE, pp. 841-846.

[59] Z. Huang. "App Writing Guide", https://wiki.o-ran-sc.org/display/ORANSDK/App+Writing+Guide (accessed 03/09, 2022).

[60] O-RAN. "O-RAN Software Community", https://github.com/o-ran-sc (accessed 03/09, 2022).

[61] Z. Huang. "Traffic Steering Flows", https://wiki.o-ran-sc.org/display/IAT/Traffic+Steering+Flows?focusedCommentId=41456537#comment-41456537 (accessed).

[62] S. Sultana and A. Mittermaier. "Updates on the 5G test network, Plans on intent based-network slicing", ITU-T FGAN. https://extranet.itu.int/sites/itu-t/focusgroups/an/input/FGAN-I-197.zip (accessed 03/12, 2022).

AUTHORS

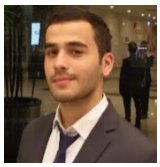
Guda Blessed received his bachelor's degree in computer engineering from the Federal University of Technology, Minna, 2021. He made several contributions to the ITU Focus Group (FG) on ML for 5G and Autonomous Networks (AN). He has research interests in AI for NLP, network security, 5G and autonomous networks and embedded systems. He is a mentor with WINEST Research group and founder of AI4Africa Research group. He received the Mentors Encouragement award from ITU AI/ML in 5G Challenge, 2021. He is currently an AI engineer at Prunny Technologies and also mentors student research projects with ITU FG-AN.

Ibrahim Aliyu received his PhD in computer science and engineering from Chonnam National University, South Korea, in 2022. He also holds a B.Eng and M.Eng degree in computer engineering at the Federal University of Technology, Minna, Nigeria, in 2014 and 2018, respectively. He is currently a postdoc researcher at Hyper Intelligence Media Network Platform Lab, Dept. of ICT Convergence System Engineering, Chonnam National University. In addition, he is contributing to the ITU Focus Group on Autonomous Networks. His current research interest is on source routing-based in-network computing for the XR/metaverse applications and the development of zone adaptive network structure for large scale metaverse deployment. His other research interests include federated learning, data privacy, network SECURITY and AI for the autonomous networks. He received the Mentors Encouragement award from ITU AI/ML in 5G Challenge, 2021, and the 2017 Korean Government Scholarship Program Award.

James Agajo received a Bachelor of Engineering (B.Eng) degree in electrical and computer engineering from the Federal University of Technology Minna, and a Master's of Engineering (M.Eng.) degree in electronics and telecommunication engineering from Nnamdi Azikiwe University, with a PhD in telecommunication and computer engineering from Nnamdi Azikiwe University. A former HOD, acting director and postgraduate coordinator, Dr James Agajo is presently an associate professor and the Head of the Department of Computer Engineering with the Federal University of Technology Minna, School of Electrical Engineering and Technology. He has published over 130 articles and received many awards, including IBM AI Analyst Master's Award, IBM AI Analyst Award, IBM IoT Cloud Developer Award, IBM Blockchain Developer Award, and IBM Telecommunications Insights & Solutions. Presently, he is also a visiting professor at I.C.T University U.S.A, Nile University Abuja, Baze University Abuja, Kebbi State University of Science and Technology Kebbi, University of Pretoria, Federal University of Petroleum Resources Effurun.

Thiago Lima Sarmento received a B.Sc. degree in computer engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil, in 2017. He received his Master's in 2019 and is pursuing his Doctor's degree in electrical engineering with emphasis on telecommunications in the Electrical Engineering Graduate Program at UFPA. He has been part of the Research and Development Center for Telecommunications, Automation and Electronics (LASSE) since 2014. His current research focuses on machine learning for telecommunications.

Cleverson Veloso Nahum received a B.Sc. degree in computer engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil, in 2019. He received his Master's and is pursuing his Doctor's degree in electrical engineering with emphasis on telecommunications in the Electrical Engineering Graduate Program at UFPA, in 2021. He is part of the Research and Development Center for Telecommunications, Automation and Electronics (LASSE) since 2016. His current research interests include network slicing, radio resource management, and artificial intelligence applied to mobile communication systems.



Lucas Novoa is a Brazilian, with a degree in electrical engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil. He is currently part of the Research and Development Center for Telecommunications, Automation and Electronics (LASSE), and he is initiating an Electrical Engineering Master's degree in the signal processing area. He has experience in the following subjects: 2G/4G/5G mobile networks, community networks, mobile networks and the transport layer.

Rebecca Aben-Athar is an electrical engineering student at the Federal University of Pará (UFPA), Belém, Pará, Brazil, since 2019. She is part of the Research and Development Center for Telecommunications, automation and Electronics (LASSE) since 2021. Her current research focuses on machine learning for telecommunications.

Mariano Moura is currently a graduate student of electrical engineering at Universidade Federal do Pará (UFPA), Belém, Pará, Brazil. He is part of a research group that focuses on 5G network automation at the Research and Development Center for Telecommunications, Automation and Electronics (LASSE).

Lucas Matni is currently a graduate student of computer engineering at Universidade Estácio de Sá, Belém, Pará, Brazil. He is part of the research group focused on the automation of 5G networks at the Center for Research and Development in Telecommunications, Automation and Electronics (LASSE).

Aldebaro Klautau received a bachelor's (Universidade Federal do Pará, UFPA, 1990), M. Sc. (Universidade Federal de Santa Catarina, UFSC, 1993) and Ph. D. degrees (University of California at San Diego, UCSD, 2003) in electrical engineering. Since 1996, he has been with UFPA and is now a full professor, at the ITU Focal Point, and directs LASSE. He was a visiting scholar at Stockholm University, UCSD and The University of Texas at Austin. He is a researcher of the Brazilian National Council of Scientific and Technological Development (CNPq), a Senior Member of the IEEE and a senior member of the Brazilian Telecommunications Society (SBrT). His work focuses on machine learning and signal processing for communications and embedded systems.

Deena Mukundan received a bachelor's degree in electrical and electronics engineering from the University of Kerala (India) in 1999, and has 20 years of experience in software engineering in the telecommunications domain. During this period, she has worked on various wireline and wireless technologies including IP-DSLAM, M2M, 3G/4G protocol stack development, and most recently in Automation. In addition to this, she has received a postgraduate diploma in AI and machine learning and is currently exploring the application of AI in the telecommunications domain.

Divyani R Achari is affiliated with Tech Mahindra, India.

Mehmet Karaca received my B.S. degree in telecommunication engineering in 2006 from Istanbul Technical University, Turkey. He received M.S. and PhD degrees in electronics engineering from Sabanci University, Turkey, in August 2008 and January 2013, respectively. After working two years as a system and research engineer at AirTies Wireless Networks, Istanbul, Turkey, he did his postdoctoral study at Lund University between 2015 and 2017. Then, he joined Ericsson AB, Lund as a system developer focusing on massive and MU-MIMO systems for Ericsson 5G base stations. Currently, he is an assistant professor at TED University, Ankara, Turkey.

Doruk Tayli works at Q Bio Inc., California, USA, as a lead computational software engineer.

Özge Simay Demirci is an undergraduate student at the Dept. of Electrical-Electronics Eng., TED University, Ankara, Turkey.

V. Udaya Sankar completed his PhD from the ECE Department at Indian Institute of Science, Bangalore, in August 2017 and did his M.Tech from IIT, Roorkee in 2002-04. His thesis is related to the design of a distributed resource allocation algorithm to mitigate interference between femtocells using game theory. He has 4+ years of industrial work experience. He was AOTS Scholar through Hindu-Hitachi Technical Training

Programme, Japan, from July 2008 to December 2008 at Hitachi Communication Technologies, Hitachi Limited, where he underwent training on standardization activity in enterprise and strategy establishment activity. He was an executive council member for IETE, Bangalore for 2008-09, an Execom member for IEEE-IISc Student Branch for 2010-2012 and Chairperson for IEEE-IISc Student Branch for 2012 and 2013. He also represented IEEE-IISc Student Branch in IEEE-UPP (University Partnership Programme) Leaders' Summit during 18-21 October 2012 at Seattle, Washington, USA. He was leading the smart city initiative by the IEEE Bangalore section from 2015 to 2016. He was a founding member of the SIAM Student chapter of IISc Bangalore, IEEE-IISc Nanotechnology council. He is an IEEE Senior Member and IEEE HKN Member. Currently, he is working as an assistant professor in ECE Department, SRM University, Amaravati, AP. His research interests include game theory and optimization, machine learning algorithms, cognitive decision making systems design, baseband signal processing for advanced wireless communications, small cell networks, self-organizing networks design using evolutionary biology concept, cooperative communications, design of algorithms for vehicular to vehicular communication, machine to machine communications, information theory and coding, IoT and smart grid.

Sai Jnaneswar Juvvisetty is currently pursuing his bachelor's in electronics and communication engineering at SRM University AP. He works as an intern in Analog Devices. His areas of interest include machine learning and deep learning. His areas of specialization also include IoT, embedded systems and 5G architecture. He also participated in the ITU AI ML 2021 and worked on the topic "Network resource allocation for emergency management based on closed-loop analysis".

V.M.V.S. Aditya is currently an undergraduate student in electronics and communication engineering at SRM University Andhra Pradesh, India. He has done several projects and has conference papers in the domain of Internet of Things, Wireless Sensor Networks, and Neural Networks. He also participated in the ITU AI ML Challenge 2021 and worked on the topic "Network resource allocation for emergency management based on closed-loop analysis". His current research interests include the various machine learning algorithms in the context of 5G and wireless networks.

Abhishek Dandekar received his bachelor's degree in telecommunication engineering from the University of Mumbai in 2015. He is currently pursuing his master's degree in ICT innovation at TU Berlin and writing his master's thesis at Fraunhofer HHI. Before this, Abhishek was with the information networking lab at IIT Bombay, where he worked on developing 5G solutions for rural India using softwarised WLAN networks. He has contributed to IEEE and ITU standardization working groups and holds a patent for controlling SDN-based multi-RAT networks. He won the judges' prize in ITU AI/ML in the 5G 2020 challenge. His current research interests include autonomous networks, distributed ML orchestration and industrial 5G.

Shabnam Sultana is affiliated with Highstreet Technologies GmbH, Germany.

Jinsul Kim received a B.S. degree in computer science from the University of Utah, Salt Lake City, Utah, USA, in 1998, and the M.S. and PhD degrees in digital media engineering, dept. of information and communications from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2004 and 2008. He worked as a researcher in IPTV Infrastructure Technology Research Laboratory, Broadcasting/Telecommunications Convergence Research Division, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea from 2004 to 2009. He worked as a professor at Korea Nazarene University, Cheonan, Korea from 2009 to 2011. Currently, he is a professor at Chonnam National University, Gwangju, Korea. He has been invited to review for IEEE Trans. Multimedia since 2008 as an IEEE Member. He has been invited for TPC (Technical Program Committee) for IWITMA2009/2010, PC (Program Chair) for ICCCT2011, IWMWT2013/2014/2015, and General Chair for ICMWT2014. His research interests include QoS/QoE, cloud computing, edge computing, AI, energy AI, multimedia communication and new media (VR, AR, metaverse etc.).

Vishnu Ram OV has 24 years of hands-on experience in the field of telecommunications industry, developing and implementing standards, holding 13 international granted patents, published many papers and was appointed as a Scientific Advisory Board Associate (SABA) member of Motorola Networks. He is currently serving as an independent consultant, vice chair of the ITU-T focus group on Autonomous Networks (ITU-T FG-AN), and was co-editor of the recently published ITU-T focus group ML5G specifications on AI/ML which led to many Recommendations such as ITU-T Y.3172. He is a Senior Member of IEEE. His current passion includes coordinating global standards in ITU-T, liaison with other SDOs like ETSI and IRTF, mentoring student projects, and coordinating global challenges in AI/ML in 5G.