A Flexible Heuristic Closed-Loop Algorithm for QoS Assurance in 5G End-to-End Network Slices

Trong Tuan Tran

A Thesis
in
Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Master of Computer Science at
Concordia University
Montréal, Québec, Canada

April 2023

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Trong Tuan Tran**

Entitled: **A Flexible Heuristic Closed-Loop Algorithm for QoS Assurance in 5G End-to-End Network Slices**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Dhrubajyoti Goswami

_____ Examiner
Dr. Abdelhak Bentaleb

_____ Examiner
Dr. Dhrubajyoti Goswami

_____ Thesis Supervisor
Dr. Brigitte Jaumard

_____ Thesis Supervisor
Dr. Tristan Glatard

Approved by _____
Dr. Leila Kosseim, Graduate Program Director

April 06, 2023 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

A Flexible Heuristic Closed-Loop Algorithm for QoS Assurance in 5G End-to-End Network Slices

Trong Tuan Tran

5G networks present new possibilities in communication technology, but they also create challenges in network management due to the incorporation of new complex concepts such as network slicing and virtual network functions (VNF). These challenges make it difficult for network operators to manually ensure that all quality of service (QoS) requirements are met across all network slices while also monitoring resource and energy consumption. To address this, automated network assurance solutions are required. Although machine learning (ML) and deep learning (DL) techniques have shown potential in this field, they come with difficulties such as acquiring real-world labeled training datasets and guaranteeing the quality of ML/DL pipelines during production.

The thesis proposes a time-driven closed-loop algorithm with proactive components that are differentiated by slice type, key performance indicator (KPI) type, and current resource consumption levels to maintain QoS for 5G end-to-end (E2E) network slices. Through simulations, we show that the proposed closed-loop algorithm is effective in resolving KPI violations and reducing network and compute resource usage, as well as allowing for flexible tradeoffs between QoS guarantees and resource consumption for each slice type via slice-specific parameter adjustments. Our solution not only enables network providers to better negotiate with customers, but also has the potential to generate training data for future ML/DL approaches.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Brigitte Jaumard, for her continuous patience, guidance, support, and strict requirements, which were essential for completing this thesis. Her expertise and encouragement were invaluable throughout my master's journey. I would also like to extend my warm appreciation to Dr. Oscar Delgado, who acted as a co-supervisor with his deep technical knowledge in the network domain. His guidance, motivation, and feedback were instrumental in helping me complete this thesis.

I am also grateful to my team mates, Tran Nguyen Phuc, Zhiyi Ding, and Mukesh Kumar Angrish, for their hard work, contributions, and dedication to the project. I learned a lot from their expertise, and their support made the research journey more enjoyable.

I would like to express my gratitude to Fadi Bishay and his team (Hesham Sedky and Logaina Boulos) for providing professional advice on 5G slicing from Ciena, and to Vincent Bissonnette for offering technical assistance from BesLogic, Montreal. Their help was crucial in achieving the research objectives.

From the bottom of my heart, I would like to thank my wife, Vo Ngoc Phuong Anh, for her unconditional love, support, and understanding throughout my challenging times. Her encouragement and motivation kept me focused, even during the pandemic.

I would also like to extend my gratitude to my uncle and his wife for providing me with accommodations in the last year, and my family in Vietnam for their unwavering support.

Lastly, I would like to wish all the best and success to my team and everyone who has been part of my academic journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General background

The fifth-generation (5G) network technology is the latest wireless communication standard designed to meet the increasing demand for mobile connectivity and support emerging technologies such as the Internet of Things (IoT), autonomous vehicles, virtual reality, and augmented reality. 5G promises to deliver faster data speeds, lower latency, higher network capacity, and more reliable connectivity compared to its predecessors. This technology is set to revolutionize the way we communicate and interact with our surroundings, enabling new applications and use cases that were previously impossible.

5G networks operate on higher frequency bands, known as millimeter waves, which enable faster transfer rates and greater data capacity. However, these higher frequencies have a shorter range and can be obstructed by physical barriers. This limitation necessitates the deployment of more cell sites and smaller cell sizes to provide coverage. To overcome this challenge, 5G technology also employs massive MIMO (multiple-input, multiple-output) technology, which utilizes multiple antennas to enhance signal quality and coverage.

One of the most significant benefits of 5G is its low latency, which is the time delay

between sending and receiving data. With latency as low as 1 millisecond, 5G is well-suited for applications that require real-time communication, such as remote surgeries, self-driving cars, and smart factories. 5G's low latency enables faster and more reliable interactions among devices, which is critical for time-sensitive applications. Additionally, 5G networks can support a much larger number of devices simultaneously, enabling a more significant number of IoT devices to connect to the network. This capability is essential for the growth and development of the IoT ecosystem, as it allows for the creation of more connected devices and smarter homes and cities.

The deployment of 5G networks requires substantial investment from telecommunication companies to upgrade their infrastructure, including the construction of new cell sites and installation of new equipment. Despite these costs, the benefits of 5G are expected to outweigh the initial investments, leading to significant advancements in various industries and improving the overall user experience. Ultimately, the deployment of 5G networks will have a transformative impact on society, creating a more connected and accessible world.

As a 5G's key technology, network slicing is a way to create multiple virtual networks on a single physical network by allocating a subset of the available network resources [22]. These virtual networks are tailored to meet the specific requirements of a set of applications and services, improving efficiency, performance, and flexibility, while also creating new revenue opportunities. Network slicing also enables resource and security isolation, ensuring that services in different network slices do not affect each other.

1.2 Quality assurance for 5G network slicing

The advancement of 5G is heavily reliant on technologies like network slicing and virtualization. These new networks face challenges in meeting the service requirements of emerging applications like uRLLC (Ultra Reliable Low Latency Communication), mMTC (Massive Machine Type Communication) and eMBB (Enhanced Mobile Broadband). To

ensure optimal performance, each application is allocated to a specific network slice, either physically or virtually separated from other slices. However, all slices run on the same physical networks, leading to limited resources and posing significant challenges for network operators. Furthermore, each network slice has different Service Level Agreements (SLAs), and network operators need to meet these SLAs through Service Assurance (SA), while a critical part of it are maintaining some key parameters in Quality of Service (QoS). To solve this challenge where traditional monitoring tools designed only for the network layer no longer fit for the purpose in the new 5G environment, there is a must to have closed-loop orchestration algorithm to address and maintain the QoS for 5G networks.

1.3 Contributions

The thesis designs a closed loop algorithm that helps sustain some key characteristics of the quality assurance (QA) aspect for 5G network slices. Main contributions are as follows:

- We formalized the selection process and gathered requirements for a list of Key Performance Indicators (KPIs) that must be met to ensure the maintenance of key aspects of Quality of Service (QoS) for 5G networks.

- We made an improvement to an existing simulation environment by enhancing the original Virtual Network Function (VNF) packet load balancer. Specifically, we replaced the traditional Round Robin mechanism with a weighted Round Robin algorithm to optimize the distribution of packets among different VNF instances within a slice based on their current dynamic resource capacities. This improvement is expected to enhance the performance of KPIs values that are particularly sensitive to compute resources, thus improving the overall efficiency and effectiveness of the simulation environment.

- We developed a closed-loop framework that is capable of retrieving network and

compute resource data from the simulation environment. The framework supports fully dynamic slices, link and Virtual Network Function (VNF) configurations that are automatically synchronized with the simulator's configurations. Additionally, the framework is able to interact with the simulation network to perform network actions, and provides automated closed-loop performance reporting capabilities.

- We developed a historical slice KPI violation and closed-loop action tracking mechanism designed to address new KPI violations using information about past violations and actions. This mechanism utilizes techniques such as repeated violations, violation leveraging, and violation skipping to improve the efficiency and effectiveness of closed-loop actions.

- We designed and implemented a 3-phase closed-loop algorithm that utilizes a two-step approach to quickly solve less complex cases without applying closed-loop actions, followed by a more detailed analysis in the final phase to determine when reactive and/or proactive closed-loop actions should be applied to modify network or compute resources. The criteria for modifying resource amounts are differentiated by action types, slice types, and the current status of network and compute resources. Additionally, we have integrated the ability to add custom rules to the closed loop system, which can be used to check for and execute specific actions designed to address known issues from the past.

- To ensure greater flexibility and control over the closed-loop actions, we also developed a set of configurable parameters that can be customized for each individual slice and resource type across different states of network resources. By modifying these parameters, network operators can control how the closed-loop actions behave for each specific combination of slice, resource type, and resource state. We have provided a guideline to help network operators modify these parameters as needed,

allowing for greater customization and optimization of the closed-loop algorithm.

- In the third phase of the closed loop, we developed a novel algorithm for sorting slice KPI violations, which combines specific 5G domain knowledge with a traditional multi-criteria decision-making (MCDM) method. This algorithm helps prioritize which KPI violations require attention first, especially in cases where multiple violations occur across different slices. By considering various factors such as the severity and impact of the violation on the slice's performance, the sorting algorithm can provide insights on how to allocate resources and apply closed loop actions to address the most critical violations.

- The core component of our closed loop is a set of nine detailed algorithms, each with unique conditions and behaviors that impact two domains: network and compute resources. These algorithms help to maintain the quality assurance (QA) of the network while being mindful of resource consumption. We leverage different techniques, such as threshold-based analysis and trend analysis, to make decisions. Our algorithms are designed to monitor resource usage in time-driven approach and take proactive and reactive measures to maintain network performance.

- Finally, we have developed a set of performance metrics and a scoring scheme to evaluate and compare the results of different scenarios and versions of closed loop algorithms. Our metrics allow for evaluation of KPI violations, resource utilization, and closed loop flexibility. This provides a valuable tool for network operators to identify and implement the most effective closed loop algorithm for their specific needs.

1.4 Plan of the Thesis

The dissertation is divided into five chapters. Current chapter presents high-level context and motivations for the research. Chapter 2 provides a detailed background of the technical aspects and literature review relevant to the research, as well as the challenges encountered during the thesis journey.

The work from Chapter 3 will be submitted as a journal paper summarizing the design, implementation, and results of our closed loop algorithm. In this work, I contributed to the methodology, implementation, experiments, and initial draft. Dr. Jaumard and Dr. Oscar provided academic guidance, technical feedback, and manuscript modification, while Fadi provided technical consultation on 5G technology and quality assurance from industry knowledge.

Chapter 4 discusses additional topics that were not covered in Chapter 3, including the architecture of our closed loop framework and an explanation of the limitations of the dynamic data rate links that significantly impacted the scenario design for evaluating the closed loop.

Finally, Chapter 5 concludes the thesis, discusses some weaknesses that can easily be addressed, and outlines some future directions.

Chapter 2

Background and Problem Statement

2.1 5G and Virtualization

The fifth generation (5G) of broadband cellular networks has been receiving a lot of attentions and expectations after the 4G LTE technology became matured world-wide since 2010s. Such high expectations are because of so many requirements put on 5G, such as: supporting different vertical markets, multi-tenancy and multi-service, supporting multiple end-to-end (E2E) slices with diverse desired performance characteristics, energy efficiency, extreme real time communication, etc.. [12, 39].

According to The Next Generation Mobile Network Board (NGMN)'s vision in the NGMN 5G White Paper: "5G is an end-to-end ecosystem to enable a fully mobile and connected society. It empowers value creation towards customers and partners, through existing and emerging use cases, delivered with consistent experience, and enabled by sustainable business models." [39]. These business models will be developed for the 3 main essential types of services in 5G: Enhanced Mobile Broadband (eMBB), Massive Machine type Communication (mMTC), Ultra-reliable Low-Latency Communications (URLLC) (as shown in Figure 1). Each of those service classes has its own unique requirements in term of data rate, latency, throughput, delay budget, etc.. [46].

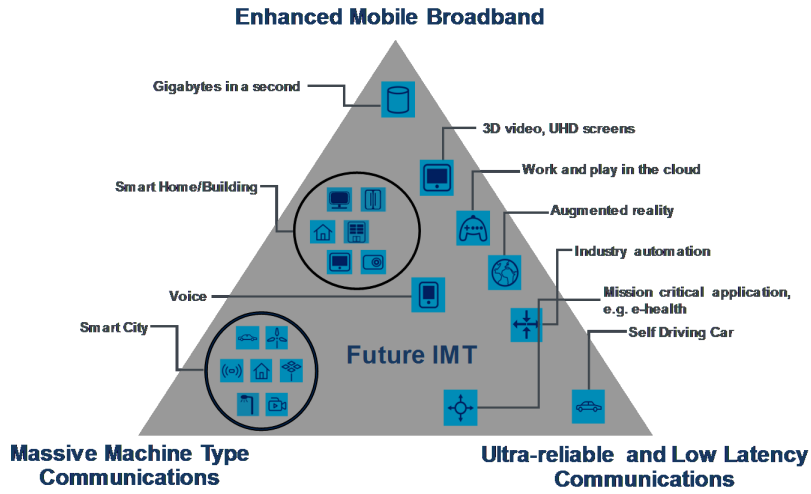


Figure 1: Usage scenarios of IMT for 2020 and beyond (Figure 2 in [46])

From 4G to 5G, there is a huge transformation with a lot of new technologies that need to be developed as the requirements to full-fill the transition. Some of the major ones can be listed here: Software Define Networking (SDN), Network Function Virtualization (NFV), Cloud Computing, Network Slicing, the New Radio (NR) and the Next Generation Core (NG Core), etc..

Software Defined Networking (SDN) has its advantages over traditional networking, due to the separation of control planes and data planes (so that new features development can be independent of the underline hardware), global network view at control plane (SDN control planes with network-wide view allows easier application policies), facilitating management automation (minimize human intervention and lower operational cost), monitoring flexibility, elasticity (resource scaled up or down on-demand), more flexible and easier performance management and network function integration [14, 32].

If SDN was a technology driven approach on networking abstraction born on campus (academic), matured on data centers and was formalized by the Open Networking Foundation (ONF), Network Function Virtualization (NFV) was driven by the Telecom business challenge and formalized by ETSI, with a different approach on service and function abstraction [37]. Together, SDN, NFV and Cloud Computing complete the virtualization

aspect across all network layers and become key enablers for network evolution, specifically 5G [37, 61].

2.2 5G Slicing and its characteristics

2.2.1 Network slicing

To efficiently meet the diverse application and business requirements of different customers, instead of building a single network to fit all, the concept of "network slicing" was introduced for 5G technology. It enables the operation of multiple dedicated networks on a common platform. ([23]).

According to the NGMN 5G White Paper 1 [39], A 5G slice or network slice is a personalized and isolated segment of a 5G network designed to support a particular type of communication service with a specific control and user plane handling. It comprises a collection of 5G network functions and Radio Access Type (RAT) settings that are combined for a specific use case or business model. A 5G slice can cover all aspects of the network and offer only the essential traffic treatment for a particular use case, avoiding any unnecessary functionality. The slice concept's flexibility allows businesses to expand or create new services, while third-party entities can use a suitable API to control aspects of slicing to provide customized services.

2.2.2 The full picture

Figure 2 illustrated the overall architecture of 5G network proposed by 5GPPP [44], with all the pieces brought together. The bottom layer with physical networks and resources infrastructure from three domains: Core/Central Cloud, Transport and Wireless (RAN), shared by the difference slice instances or logical networks are created on top. This requires

continuous monitoring and automation to ensure that customer-centric service level agreements (SLA) are met, and automation is required for managing network slice instances throughout their lifecycles (which is controlled by the "Orchestration" blocks across all areas). The proposed architecture utilizes software-defined, programmable network functions (SDN and NFV) and infrastructure resources, and a recursive structure is used to improve scalability and enable a slice instance to operate on top of resources provided by another slice instance.

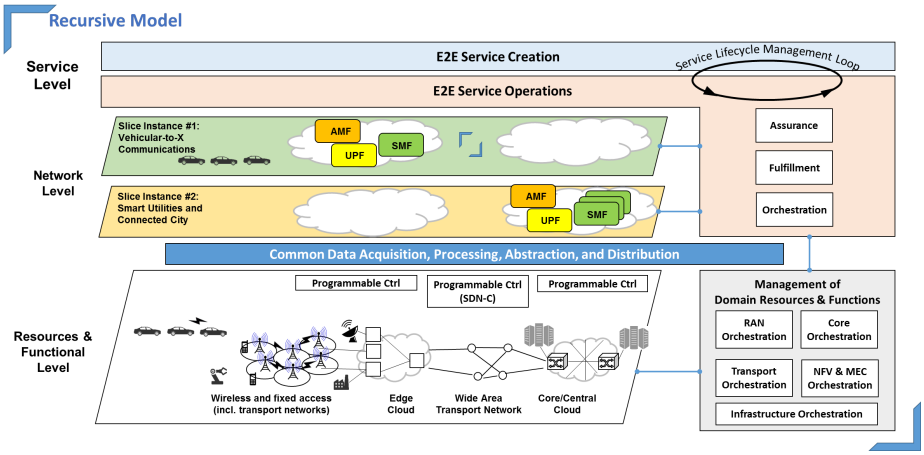


Figure 2: 5G Overall Architecture from 5G-PPP (Figure 2-2 in [44])

In addition, it's worth noting that there are two types of slicing concepts: horizontal slices and vertical slices. Horizontal slices are the end-to-end network slices explained previously, while vertical slices refer to dedicated networks with different requirements for various industries, such as manufacturing, healthcare, and automotive [17]. Our work specifically focuses on the traditional horizontal slices of network slicing.

2.2.3 Service Level Agreement, Service Assurance, QoS and KPIs

Definitions

Service Level Agreement (SLA) is a formal agreement between a service provider and its customers, whether internal or external, that outlines the specific services that will be

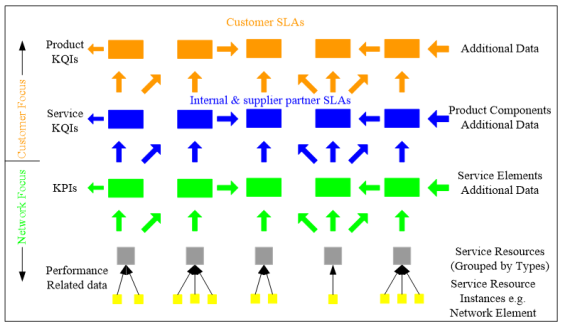
provided and the performance standards that the provider is responsible for meeting [27]. It can be understood as the formal contracts between the two sides where each generic term is detailed by a detailed objective for certain metrics called Service Level Objective (SLO).

In network context, an SLA contract outlines the agreed-upon service level and includes the quality of service (QoS) that the provider must meet. Service assurance (SA) refers to the set of activities and processes that ensure that the service provided meets the agreed-upon SLA and QoS.

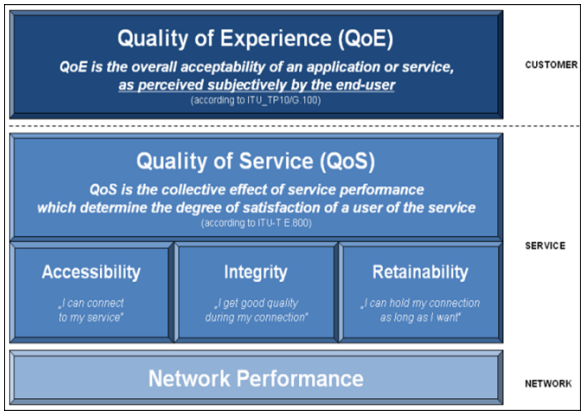
QoS refers to the performance characteristics of a service or network, such as network speed, latency, reliability, availability, and security. QoS is typically measured using Key Performance Indicators (KPIs), which are used to assess the network's performance and ensure that it meets the agreed-upon service level.

In other words, SLA defines the expected QoS for a service, and SA is responsible for monitoring and ensuring that the QoS meets the SLA. KPIs are used to measure and evaluate the network's performance and ensure that it meets the required QoS levels. Overall, SLA, SA, QoS, and KPIs are all closely related and work together to ensure that the network provides high-quality services that meet the needs of customers.

Figure 3 describes those relations of SLAs, KPI and QoS in two sub-figures. Figure 3a on the left shows KPIs at the low level Network focus path to help network providers satisfy SLA requirements with customer. Those KPIs data are measured and collected from different resource instances and network elements. Figure 3b on the right illustrates the QoS with different areas (Accessibility, Integrity, Retain-ability), each area will have corresponding KPIs to be measured and monitored, with the end goal is to statisfy the Quality of Experience (QoE) of customers.



(a) SLAs and KPIs (Source: Fig. 1 in [3])



(b) QoE, QoS and KPIs (Source: Fig. 6.3.1 in [1])

Figure 3: Interrelations between SLAs, QoS and KPIs

SLA and QoS in 5G network slicing context

For 5G network slices, the SLAs between network providers and customer are realized through Generic network Slice Template (GST). A detailed agreement instance using the template with filled values is called e NEtwork Slice Type (NEST). Both GST and NEST are described at great details by GSM Association in [24].

Figure 4 shows an example from [24] of a minimum set of attributes from the GST need to be specified for the High-Performance Machine-Type Communications (HMTC) slice type. The "Attribute" column are the SLAs from the GST, and the agreed values are in the right column. The last SLA attribute "Slice quality of service parameters" define the required QoS characteristics for the HMTC slice type, where all of those 5G QoS parameters and characteristics are well defined by 3GPP in [4].

Mapping between 5G QoS characteristics and End-to-End KPIs

3GPP defined all the network 5G KPIs from different areas (Accessibility, Integrity, Utilization, Retainability, Mobility and Energy Efficient) in [2] with a lot of End-to-End and sub-domain KPIs. In order to satisfy the QoS components of the SLA, network provider

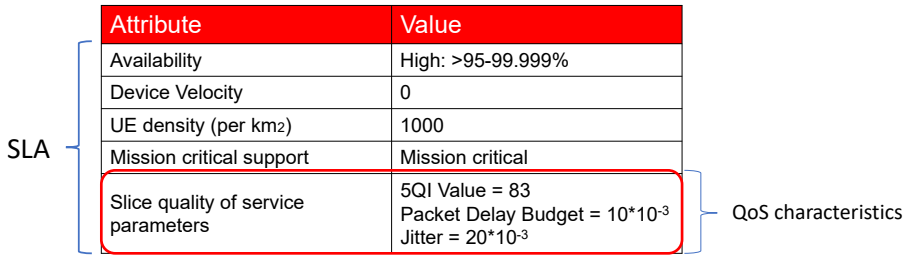


Figure 4: Example of a minimum set of attributes from GST tempate with filled values for High-Performance Machine-Type Communications (HMTC)

must select the proper KPIs to be monitored and map to the specific 5G QoS characteristics. This activity exposes great challenges since there is no official guidance for that, which requires a lot of 5G domain expertise, and to make matter worse, network equipment from different vendors has different types of KPI measurements and not all required KPIs available.

Most of the QoS characteristics can be found in the Integrity KPIs category. Figure 5 presents an example of our manual mapping efforts to find the relevant delay related KPIs from [2] (blue column headers on the left) with the relevant QoS characteristics from [4] (Green column headers on the right), after picking only delay KPIs belong to “NetworkSliceSubnet” and “NetworkSlice” KPI object levels as recommended by 5G experts from Ciena. All of the grey rows are sub-domain delay KPIs which span between different components in RAN domain, while the yellow rows refer to E2E delay of the whole network slice that links accros three domain Core, Transport and RAN. All of those delay KPIs are most relevant to the "Packet Delay Budget" characteristic of 5G QoS.

KPIs (3GPP TS 28554: MANO 5G E2E KPIs)					5G QoS characteristics/Quality SLAs (NRM)							
KPI type	KPI name (full)	KPI (short)	KPI object *(Data Network of the object instance where the KPI is applicable, including the object where the measurement is made)*	Domain	fiveQIValue (5QI)	Resource Type	Priority Level	Packet Delay Budget	Packet Error Rate	Averaging Window	Maximum Data Burst Volume	jitter
Integrity KPIs	Downlink delay in gNB-DU for a network slice subnet	DLDelay_gNBDU_Nss	NetworkSliceSubnet	RAN				x				x
Integrity KPIs	Downlink delay in gNB-CU-UP for a network slice subnet	DLDelay_gNBCUUP_Nss	NetworkSliceSubnet	RAN				x				x
Integrity KPIs	Uplink delay in gNB-DU for a network slice subnet	ULDelay_gNBDU_Nss	NetworkSliceSubnet	RAN				x				
Integrity KPIs	Uplink delay in gNB-CU-UP for a network slice subnet	ULDelay_gNBCUUP_Nss	NetworkSliceSubnet	RAN				x				
Integrity KPIs	Uplink delay in NG-RAN for a network slice subnet	ULDelay_NR_Nss	NetworkSliceSubnet	RAN				x				
Integrity KPIs	Average e2e uplink delay for a network slice	DelayE2EUlNs	NetworkSlice	E2E (Aggr)				x				
Integrity KPIs	Average e2e downlink delay for a network slice	DelayE2EDlNs	NetworkSlice	E2E (Agg)				x				

Figure 5: Manual mapping between delay related KPIs with QoS characteristics

2.2.4 Virtual Network Functions (VNFs)

NFV concept

As in [9], telecommunication service providers (TSPs) traditionally need a wide variety of hardware appliances to provide network functions, which can be time-consuming, complex, and expensive to install, operate and maintain. The introduction of Network Function Virtualization (NFV) by The European Telecommunications Standards Institute (ETSI) in 2012 decouples network functions from dedicated hardware and provides virtual servers to run these functions as software, which can reduce the need for specialized hardware and lower operational and capital expenses. Figure 6 illustrates that concept of VNFs, from each dedicated hardware serving specific network service (or function) with the support of virtualization and SDN technologies, those functions can be implemented in virtual (software) form sharing the same physical infrastructures.

Figure 7 shows high level architecture of NFV systems, where the virtualization layer from previous figure 6 provides compute (CPU and RAM), storage resources and network resources (example: bandwidth or link capacity) to the virtual functions to serve the network services, where those resources are retrieved from the physical resources layer. In order for the resource distribution and VNFs operate properly and efficiently, there is a must for the present of a management and orchestration system (MANO).

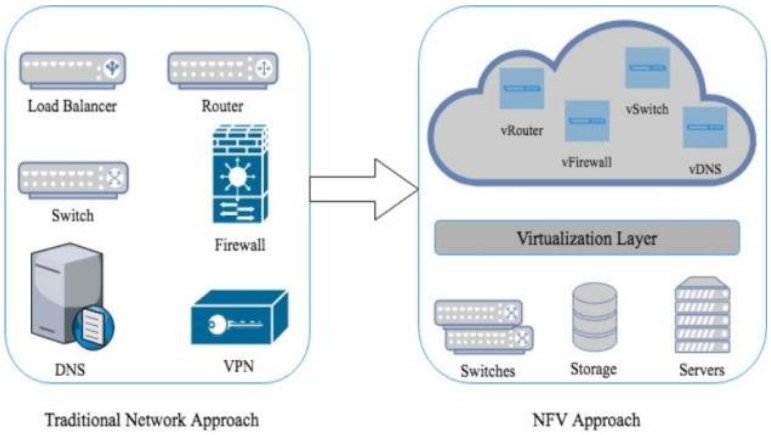


Figure 6: Transformation from traditional network functions to VNFs (Source: Fig.2 in [9])

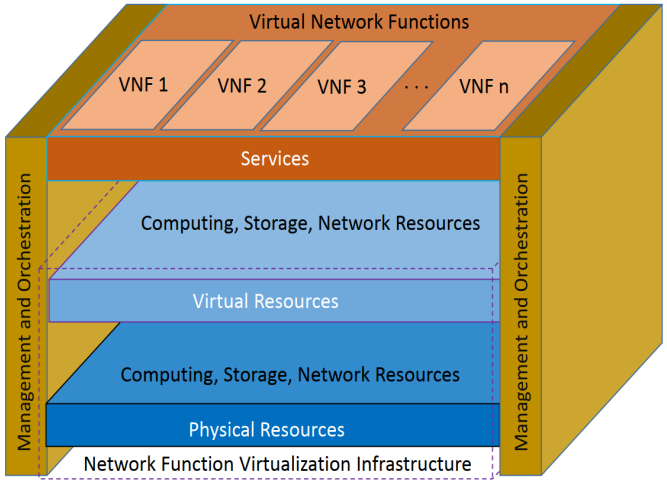


Figure 7: Network Function Virtualization architecture (Source: Fig.4 in [37])

A network service is composed of one or a sequences of both physical and virtual network functions, with is called a service function chain (SFC). There is a need for network providers to find and map each network function to a specific physical location when deploying a SFC, as illustrated in figure 8. However, this is not in the scope of our research.

NFV promises benefits such as independence, flexibility, scalability, and reduced energy consumption by allowing the dynamic allocation and sharing of infrastructure resources to perform various functions.

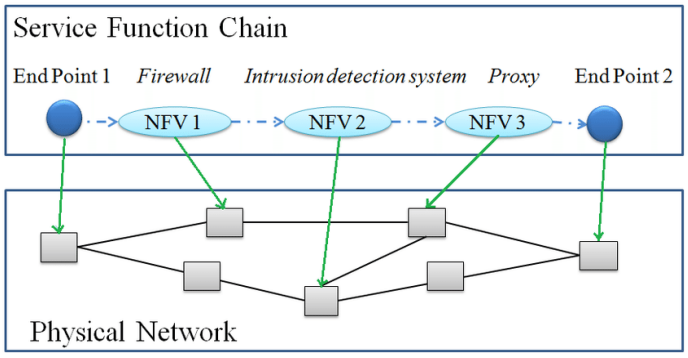


Figure 8: Service Function Chain mapping (Source: Fig.1 in [26])

NFV enables TSPs to deploy network functions faster and more flexibly, and dynamically scale the performance of virtualized network functions with finer granularity. By transforming physical network appliances into virtual ones, NFV facilitates the development of new businesses and creates a new market. This technology can help TSPs to keep up with the increasing demands for launching new network services, and also reduce their environmental impact by reducing energy consumption.

NFV challenges

One of the most relevant challenges from NFV is: according to the authors in [37], despite the existence of a lot of works and NFV solutions from industry, they have primarily focused on pooling resources specific to vendors and hosting them in a cloud. However, they have not provided adequate support for essential NFV requirements such as flexibility, inter-operability, integrated management, orchestration, and service automation.

2.2.5 Closed loop automation

5G networks involve various complex technologies such as Network slicing, NFV, diverse QoS, and orchestration, which make it challenging to ensure seamless network operations and meet SLA demands of different slice type customers using traditional tools and methods. Therefore, to overcome these challenges, a fully automated end-to-end control system

is required to manage all aspects of the 5G network slices, especially the quality assurance and resources consumption areas. There are two approaches to automation, open and closed loop control systems. The open-loop approach is similar to traditional methods in telecommunications networks, where providers manually control specific steps in the process. However, the closed-loop approach aims to achieve full process automation and "zero touch control" by setting goals and supervising system operations. Given the complexity of 5G slices, manual control steps in the open-loop approach are not feasible for operators, and hence, a fully supervised closed-loop system is necessary. Figure 9 provides an illustration of the two control loop approaches.

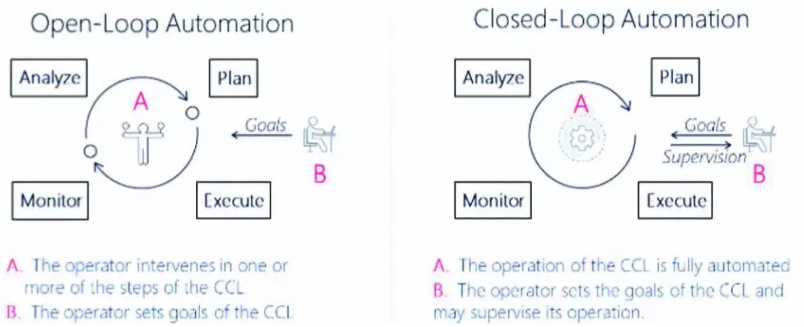


Figure 9: Open-loop automation and Closed-loop automation (Source: Fig.1 in [11])

2.2.6 Trend analysis

Trend analysis is a critical component of our closed-loop algorithm for evaluating KPIs and resource statuses of network slices, and plays a key role in the proactive decision-making process for closed-loop actions. We studied several trend evaluation techniques [53], and finally we have chosen to implement the Mann-Kendall test for its effectiveness in detecting trends in monotonic sequence data without seasonal components, in combination with the Theil-sen's Slope Estimator.

The Mann-Kendall test [36] is a non-parametric test used to determine if there is a trend in a monotonic sequence of data. The test is based on the Mann-Kendall statistic, which is

calculated as the sum of the signs of the differences between all pairs of observations. The null hypothesis of the Mann-Kendall test is that there is no trend in the data. The alternative hypothesis is that there is a trend, either increasing or decreasing.

The test statistic is calculated as:

$$S = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} sign(x_j - x_i) \tag{1}$$

where x_i and x_j are the ith and jth observations, $sign(x_j - x_i)$ is the sign of their difference, and n is the sample size.

The test statistic is then standardized to obtain the test statistic Z:

$$Z = \frac{S - E(S)}{\sqrt{Var(S)}} \tag{2}$$

where $E(S)$ and $Var(S)$ are the expected value and variance of the Mann-Kendall statistic under the null hypothesis.

Under the null hypothesis, the test statistic Z follows a standard normal distribution. The p-value can then be calculated as the probability of observing a value of Z as extreme or more extreme than the observed value, given the null hypothesis. If the p-value is less than the chosen significance level (e.g. 0.05), then the null hypothesis is rejected and there is evidence of a trend in the data.

The Sen slope estimator can also be calculated as:

$$b = \frac{\text{median}(x_j - x_i)}{\text{median}(t_j - t_i)} \tag{3}$$

where t_i and t_j are the time intervals between the ith and jth observations. The Sen slope estimator provides an estimate of the magnitude and direction of the trend, with positive values indicating an increasing trend and negative values indicating a decreasing trend.

2.3 Literature review

2.3.1 5G Slicing and Orchestration

Rokui *et al.* [45] provide a comprehensive overview of the concept of 5G network slicing. They describe the current efforts by multiple Standard Development Organizations (SDOs) and industry players to achieve an end-to-end slice management solution, which includes core and RAN network slicing from 3GPP, transport network slicing from IETF, and a combined solution from ETSI's ZSM ISG (Zero Touch Network and Service Management Industry Specification Group). The latter aims to introduce a model-driven solution to transport slicing from IETF. The authors also discuss the ONAP project's (Open Network Automation Platform) demonstrative implementation work, which was planned for release in the second half of 2020.

Barakabitze *et al.* [11] conducted a comprehensive survey on various aspects of 5G network slicing using SDN and NFV, including architectures, management, collaborative effort and the associated challenges. While SDN/NFV-based 5G networks provide numerous benefits like flexibility and agility, they also present challenges such as the need for for dynamic resource allocation efficiency and Network functions (NFs) placement. To manage the complexity of SDN/NFV-enabled 5G network slices, Management and Orchestration (MANO) of NFV is essential, with orchestration covering a single or multiple domains, along with E2E slice orchestration management. However, a significant challenge in this area is the method to realize high-level service description to the concrete and actual network slice. Additionally, diverse slice type SLAs require sophisticated orchestration plans and adaptive solutions that can help make appropriate decision on resources management based on current or future state of system and user demand.

2.3.2 Closed loop frameworks

In [51], Vaishnavi and Ciavaglia explain how open and closed control loops are crucial for achieving greater levels of network automation. Furthermore, they highlights the obstacles that operators encounter when implementing control loops in real-life scenarios, and suggests initial solutions based on standards. The authors first introduced well known generic closed loop architectures such as MAPE-K and OODA with similar concepts and the same key components: (1) Information acquisition, 2) Information analysis, 3) Decision making, 4) Action execution, and Knowledge [K]). Then some telecom specific closed loop frameworks are presented such as FOCALE architecture or SON (self-organizing networks). The SON architectures proposed by 3GPP has three basic use case categories: self-configuration, self-optimization and self-healing. There were also more recent works such as the Unified Management Framework (UMF) which aims for multi-vendor CCLs, and even more cognitive (self-learning) approaches with integration of AI/ML into the closed-loops. Finally the author discuss some challenges of closed control loops deployment in different areas, such as types of deployment, how to control and supervise, trust in CCL's operation, operational policies and conflicts management between multi-CCLs interactions. In conclusion, the paper stated that the standards, industry and academic bodies need to collaborate to efficiently realize the design and validation of CCLs in practice.

2.3.3 Service assurance

Xie *et al.* [55,57,60] and Kim *et al.* [31] studied and proposed a series of service assurance (SA) architectures for 5G network slices. First, in [31], the authors proposed the extended version of NFV MANO framework from ETSI to interface with the E2E slices assurance architecture, with a four-layer (i.e., Infrastructure, VNF, Network Service and Slice Assurance) hierarchical SA architecture. In [60], the authors presents some key ideas for the SA architecture: A SA system should be able to interact with the slice orchestration and

management (SOM) at all network layers and domains, and requires the combination of monitoring and analysis of network and resources data to assure the network services. The SA function at each layer should be connected and aware of other SA functions of other layers and domains (cross-layer awareness). Those concepts are demonstrated through some simulation with a four-layer network and shows that the scenario which enabling all of the mentioned ideas achieved the best results with optimal balance between delay and cost. In [57], the author propose the detailed architecture for SA framework in a specific network layer as well as the overall SA architecture that allow interaction between slice assurance of different service providers, from the 5G-VINNI project. The overall SA architecture is called "hierarchical SA" that provide assurance for all 5 network layers: Infrastructure-SA, Network Function (NF)-SA, Network Service (NS)-SA, E2E Slice Assurance and customer facing service assurance (CFSA). The interaction between different SA functions requires different types of closed loops: internal closed loop, external closed loop as well as cross-layer closed loop with the some challenges to to implement them such as interfaces and information models, coordination, policy management, monitoring and isolation. In [55], the authors propose the benefits, architecture and method to design cross-closed loop systems that allow SA systems from different network providers to collaborate with each other to serve end customer's needs, through a proposed workflow and intent and policy management.

In their paper [35], Mai *et al.* propose a novel optimization-based distributed algorithm to autonomously meet end-to-end quality of service (E2E QoS) requirements for core and access 5G/6G networks. The author formulates the problem as a routing optimization problem that takes into account the traffic class (i.e. traffic of a specific slice type) to minimize resources cost at each local domain network (also known as subsystem), while still satisfying the global constraint, which is the thresholds that the E2E KPIs cannot be exceeded. Compared to previous distributed algorithms, the proposed algorithm has two

main advantages. First, it uses dynamic negotiation for KPI budgets instead of static local budgets for each domain network. Second, instead of sharing local decision variables with all other subsystems, each domain network only needs to exchange its estimated values global constraint functions, which reduces the heavy signaling overheads of previous methods. To demonstrate the effectiveness of the algorithm, the authors conducted a numerical study on a small network with two traffic classes, one core network (CN) and two access networks (ANs), each AN having only two links. The optimization problem aimed to minimize the total costs of each of the three networks while guarantee different average E2E delay thresholds for each traffic class. The cost of each AN includes cost of reserving links bandwidth and the cost dependent on experienced average traffic delays, while costs of the CN is total cost of provisioning bandwidth and delay cost function for both traffic classes. Although numerical results suggest that the algorithm can converge to the optimal solution after a small number of iterations, the scalability of the algorithm to much larger networks in real-life scenarios remains uncertain. Additionally, the study does not cover virtual network function (VNF) concepts, raising questions about how to incorporate VNF resource models into the algorithm.

In a separate area of research, Nhu *et al.* [40] propose a dynamic VNF scaling algorithm for the core network. The authors employ deep learning techniques to develop a multivariate time series forecasting model that predicts future CPU and RAM resources based on various VM performance metrics. Using upper and lower thresholds, the system can then adjust the number of VM instances to optimize compute resource consumption. Specifically, the authors use an Auto Encoder-Decoder framework augmented by an attention layer, with the encoder using a Bidirectional-LSTM model and the decoder using a traditional LSTM model. They train the model on an open-source dataset containing performance metrics data of 527 data center VMs. The authors also design an Automated Resource Configuration System based on the Open Source MANO framework (OSM) and

the VIM module of the OpenStack framework to enable VNF instance scaling. The prediction algorithm and system are tested on a small test bed with three computers and two core network slices, and the model's forecast ability is evaluated using metrics such as MSE, RMSE, MAE, and R^2 across three scenarios: short, medium, and long-term predictions (predicting 1, 5, and 10 steps ahead). The results show that the proposed model outperforms other deep learning and classical models in terms of metrics scores and resource scaling effectiveness. However, the paper mostly focuses on VNF resource optimization in the core, with limited discussion on QoS assurance and E2E KPI thresholds. While the authors mention reducing SLA violation costs, they do not explicitly address the QoS assurance aspect.

In conclusion, our literature review shows that most of the related works focus on the service assurance architecture aspect without providing a detailed closed-loop algorithm. Moreover, none of the proposed algorithms can fully satisfy E2E QoS goals for 5G network slices while taking into account both network and compute (VNF) resources. Therefore, there is a need for further research to develop a comprehensive closed-loop algorithm that can both guarantee E2E QoS goals and optimize all domains' resource utilization for 5G E2E network slices.

2.4 Challenges

There are a lot of challenges we encountered during this research. Some of the major ones:

- The first and most critical challenge is the lack of a real 5G network lab and equipment or even an actual 5G test bed, especially due to the advent of the pandemic that made access to Ciena's lab become impossible. This obstacle limits access to actual 5G networks and traffic data (which is also difficult to find both from public and industry with a lot of legal concerns). That's why we need to used a self developed 5G

traffic generator [20] to help generate slices traffic data with real life characteristics.

- We used OMNeT++ [52] as our network simulator to generate data and test our closed-loop algorithm. OMNeT++ is a packet-level simulator that offers high granularity, making it a viable option for obtaining near-realistic slice-specific traffic data. However, the simulator has some major disadvantages, such as single CPU usage and slow simulation time. Despite these limitations, we were able to design a network with four slices that generated high levels of KPI and VNF traffic during peak times. To improve simulation speed, we implemented several workarounds, such as scaling down traffic and using the largest possible time resolution while maintaining KPI data measurement accuracy at 0.1ms. However, due to the simulator's limitations, running the full 24 hours of simulation would require at least 18 to 20 real-time hours. Despite these challenges, OMNeT++ remains the most suitable simulator for our needs.

- Another challenge is the lack of single sources to formulate KPI and QoS goals. 5G network knowledge domain and service assurance is so broad, so at the beginning of the research we need to spend significant amounts of time to clarify the problem statements, requirements, defining scopes, finding 5G network slices' QoS characteristics and select KPI goals which are scattered in so many sources, then combine into a full set of slice KPI requirements.

- Moreover, most papers and academic works focus on the Service Assurance architecture rather than the actual algorithm. Thus, there is no existing heuristic algorithm with similar goal that we can use to benchmark. Most of traditional optimization techniques also do not scale well with big data [41], making it even more difficult to find an already working algorithm.

- There is no framework that can directly work with the selected simulation environment. Current trends in network orchestration field leverage existing open source platforms such as ONAP as Kubernetes, but they are not easy to integrate into OMNeT++ and requires powerful hardware [42] that our team cannot afford to build a network of multiple nodes that is reasonably sized and not too small. That is the reason why besides the core algorithm, we must develop the closed loop from scratch and implement all of the network actions to interact with the simulator with huge amount of time and efforts.

- Currently there are further known challenges of the simulation environment that make it difficult to perform the proper network dimensioning aspect in order for the closed loop to work, such as the VNF delays model, VNF queue duration fine-tuning. A few examples: First, we spent a lot of time to find out the limitation of current dynamic data rate link implementation that make the simulation crash due to a bottle neck condition, then searched for a minimal condition that the simulation can run without crash to demonstrate the impact of our closed loop. Second, we did find it struggle to search for the proper resource ranges and limits that the closed loop can perform both vertical and horizontal scaling actions.

Chapter 3

5G Slices Service Assurance Closed-Loop Algorithm

3.1 Introduction

Novel wireless mobile networks like 5G and beyond rely more an more on technologies such as network virtualization and network slicing. Additionally, the emerging applications such as uRLLC (Ultra Reliable Low Latency Communication), mMTC (Massive Machine Type Communication) and eMBB (Enhanced Mobile Broadband) have stringent service requirements that challenges current network management systems. Each of those applications will be allocated and running on different types of network slices which are physically or virtually separated to ensure each slice does not impact on each other's performance. However, those slices run on the same physical infrastructure with limited network and compute resources, which is one of the most critical challenges for network operators [48].

Virtualization techniques, such as Network Function Virtualization (NFV) and software-defined networking (SDN), are key enabling technologies for 5G networks. Many traditional and new network functions have been virtualized and can be deployed on commodity computing infrastructure, such as virtual machines (VMs), data centers, or the cloud.

This approach enables network operators to easily add or reduce network and compute resources, or add and remove compute instances, which provides more flexibility for network management. However, it also introduces difficulties in optimizing compute resource utilization. Efficient resource consumption is crucial for minimizing energy consumption and costs, while ensuring the network is not impacted by over-reduction of resources. SDN can help to optimize resource utilization by allowing for dynamic allocation of resources and traffic management through centralized controllers. Thus, all of these difficulties require a "zero-touch" closed loop orchestration algorithm to control and manage the network slices and resources effectively [15].

Additionally, each network slice type has different Service Level Agreements (SLAs) that network operators must meet for customers through service assurance (SA). Although SLAs or SA encompass various aspects, our work focuses on maintaining key quality parameters of 5G Service Assurance: We guarantee specific QoS parameters outlined in the 3GPP 5G specifications [4] by sustaining threshold-based goals for selected Key Performance Indicators (KPIs) [2]. This approach ensures that the network provides the required QoS for different types of services in a network slice, such as latency, throughput, reliability.

Our contributions are the following:

- We developed a historical slice KPI violation and closed-loop action tracking mechanism designed to address new KPI violations using information about past violations and actions. This mechanism utilizes techniques such as repeated violations, violation leveraging, and violation skipping to improve the efficiency and effectiveness of closed-loop actions.

- We designed and implemented a 3-phase closed-loop algorithm that utilizes a two-step approach to quickly solve less complex cases without applying closed-loop actions, followed by a more detailed analysis in the final phase to determine when

reactive and/or proactive closed-loop actions should be applied to modify network or compute resources. The criteria for modifying resource amounts are differentiated by action types, slice types, and the current status of network and compute resources. Additionally, we have integrated the ability to add custom rules to the closed loop system, which can be used to check for and execute specific actions designed to address known issues from the past.

- To ensure greater flexibility and control over the closed-loop actions, we also developed a set of configurable parameters that can be customized for each individual slice and resource type across different states of network resources. By modifying these parameters, network operators can control how the closed-loop actions behave for each specific combination of slice, resource type, and resource state. We have provided a guideline to help network operators modify these parameters as needed, allowing for greater customization and optimization of the closed-loop algorithm.

- We developed a novel algorithm for sorting slice KPI violations, which combines specific 5G domain knowledge with a traditional multi-criteria decision-making (MCDM) method. This algorithm helps prioritize which KPI violations require attention first, especially in cases where multiple violations occur across different slices. By considering various factors such as the severity and impact of the violation on the slice's performance, the sorting algorithm can provide insights on how to allocate resources and apply closed loop actions to address the most critical violations.

- The core component of our closed loop is a set of nine detailed algorithms, each with unique conditions and behaviors that impact two domains: network and compute resources. These algorithms help to maintain the quality assurance (QA) of the network while being mindful of resource consumption. We leverage different techniques, such as threshold-based analysis and trend analysis, to make decisions. Our

algorithms are designed to monitor resource usage in time-driven approach and take proactive and reactive measures to maintain network performance.

- Finally, we have developed a set of performance metrics and a scoring scheme to evaluate and compare the results of different scenarios and versions of closed loop algorithms. Our metrics allow for evaluation of KPI violations, resource utilization, and closed loop flexibility. This provides a valuable tool for network operators to identify and implement the most effective closed loop algorithm for their specific needs.

The rest of the paper is organized as follows: In Section 3.2 we discuss the most relevant related works. In Section 3.3, we describe the architecture and some key parts of the proposed telemetry close loop algorithm. In Section 3.4 we outline the network setup and the analysis of the simulation results. Finally, Section 3.5 briefly concludes the paper.

3.2 Background

The need to design an automatic service assurance mechanism for end-to-end network management has been highlighted in the previous section. Several approaches to accomplish 5G network management and service assurance are proposed from literature.

In [19] Sousa *et al.* proposed a closed-loop management framework based on high-level policies that decouples the service management from the service deployment but offered no experiments to validate their proposal.

The authors of [56] [59] proposed a high level service assurance architecture but left many challenges unsolved as the policy management still needs to be developed.

Another interesting paper written by Xie *et al.* [58] investigates the impact of securely expose management capabilities of the networks and services to authorized customers or third party companies.

Naik *et al.* [38] showcases zero-touch provisioning and closed-loop orchestration only for core network slices with the aid of AI-based operational analytic for anomaly detection, root-cause analysis and automated remediation.

A supervised learning based service assurance architecture for 5G was proposed in [63]. This solution learns and triggers automatic actions to mitigate detected anomalies, nevertheless it was tested in an open-loop network so it is hard to see if the recommended actions effectively mitigate the root problems.

The authors of [43] describe a closed-loop ML framework that showcase anomalous detection, and traffic load and SMF (Session Management Function) resource usage prediction. Although the authors describe a flexible framework they do not develop a service assurance algorithm.

Another approach, that partially tackles service assurance, was developed in [33]. The proposed solution describes an automatic UPF fail-over procedure, executed using the Open Network Automation Platform (ONAP).

In [28] the authors developed a reinforcement learning agent that focuses on O-RAN automation, and tested its applicability by optimizing power allocation.

A dynamic VNF scaling for the 5G core was proposed in [40]. The authors used a predictive algorithm that forecast resource consumption (CPU, memory) and recommend VNF scaling. However, it focuses on the forecasting aspect, and not in the decision making logic, not to mention than the experiments are mainly concern with the 5G core.

An intent based platform that automates and manages the network slice life-cycle was developed in [8]. The authors tested their proposed solution by creating multiple slices of core and RAN, However, their results were limited to test throughput in an small time scale.

In [35] the authors proposed a distributed algorithm that enables autonomously managed access and core 5G networks to cooperate with each other to meet the E2E KPI goals.

Although this approach handles better the domain KPI budgets, it was tested in a limited scenario (down-link with 2 slices).

From literature review, it is apparent that no existing model has achieved satisfactory results in an end-to-end 5G network or specifically addressed the modulation of network and/or compute resources with actual quality assurance in the network slicing context. As a result, the need for network operators to reduce the cost of SLA violations and sub-optimal energy use remains an open research question. Therefore, we propose a novel model based on thresholds and trends that can dynamically manage network and compute resources to satisfy specific network requirements. This model will ensure that SLA requirements are met by minimizing KPI violations and allow for more efficient resource utilization.

3.3 Design

3.3.1 Assumptions / Scope

While there are many challenges to work with a real 5G network, such as limited access to an end-to-end slice enabled 5G test bed, it is necessary to define the scope and apply specific assumptions in the design of our closed-loop algorithm.

Scope

- Among the many aspects of Service Assurance (SA), in this work, we specifically focus on assuring the QoS by monitoring and setting objectives for selected network KPIs (Key Performance Indicators) outlined in the 5G 3GPP KPIs spec [2].

- A given KPI can be related to an specific domain: Radio Access Network or RAN (R), Transport (T), Core (C), or it could be an end-to-end (E2E) KPI, such as Delay, Jitter or Packet Loss, whose impact might be distributed among multiple domains.

Due to the limitation of the current simulation environment that is yet to allow access to sub-domain KPIs, we decided to only monitor and assure the E2E KPIs.

Main goals

In the first version of our QoS assurance closed loop, the first priority is to minimize the number of KPI violations on 5G slices, caused by temporary lack of network or compute resources, the second priority is to minimize the total operation cost derived from the use of resources and the closed loop actions performed on the network. Moreover, the project does not try to cover root cause analysis (RCA) and fault localization when a KPI violation is detected.

Network state assumption

We assume that the maximum number of VNFs and slices that can be allocated have already been optimized based on the network capacity, the known types of traffic and the traffic distribution, so the number of network slices is fixed and there would be enough resource capacity (bare metal capacity) for the incoming traffic demand. Additionally, the proposed closed loop algorithm does not deal with the admission control problem.

Assumptions on VNFs

There are more than one type of VNFs: some are dedicated VNF types that can only serve one slice at a time, such as UPF, SMF, NEF, etc., while some other types can be used by multiple slice instances, such as AMF [29]. Our current closed loop only works with dedicated VNFs type, specifically for simplicity we use only one VNF type, which is the UPF (User Plan Function), where the mapping relationship between a slice and the VNFs is one-to-many (a slice can be served by multiple VNF instances, but a VNF instance can only be dedicated to a single slice). We also don't take service function chains (SFC, a series of

connected VNFs of different types to serve some specific purpose) into consideration.

We also assume the ideal case in which the execution time of the closed loop actions is negligible, in contrast with resource scaling features of current cloud providers, where a resource scaling or virtual machine (VM) scaling might require the restart of those VM instances and running applications.

Resource Scaling behaviour

Regarding closed loop actions, specifically link capacity and compute resources vertical scaling, we apply continuous scaling, meaning that the system accepts any decimal value of scaling step of network data rate capacity or compute resource limits, in contrast with the discrete scaling approach where the scaling amounts are fixed by a pre-determined step value.

Heuristic vs Machine Learning

With the increasing potential of artificial intelligence (AI), specifically machine learning/deep learning (ML/DL), academics and industrial researchers are turning to these techniques to achieve better results in complex tasks where traditional algorithms designed by humans fall short. Applying ML/DL to difficult areas such as the 5G service assurance context seems obvious and correct; however, the absence of a public dataset containing quality assurance data from a real 5G network poses a challenge. The lack of this dataset means that collecting data of QoS assurance actions decided by human experts to use as training labels is both time and cost consuming. Moreover, integrating and maintaining ML pipelines in production poses significant challenges [50]. Therefore, our team designed a heuristic re-active closed loop algorithm that applies some proactive components, without using actual prediction of future network states, as a simpler and more feasible approach[1].

[1] Inspired by rule #1 in [64]

This heuristic closed loop algorithm can also be used to generate initial training data with labels for ML/DL approaches, or even act as the baseline method for the application of reinforcement learning (RL) in the future, as seen in the "learning from baseline" approach from [34]. Given that our closed loop algorithm already timely records all necessary input and output information, generating training data with labels from the closed loop is straightforward.

3.3.2 5G Network description

The 5G network used in this paper is based on the simulator developed in [20]. This simulator enables network virtualization and network slicing, and is capable of handling multiple types of traffic, as shown in Fig. 10.

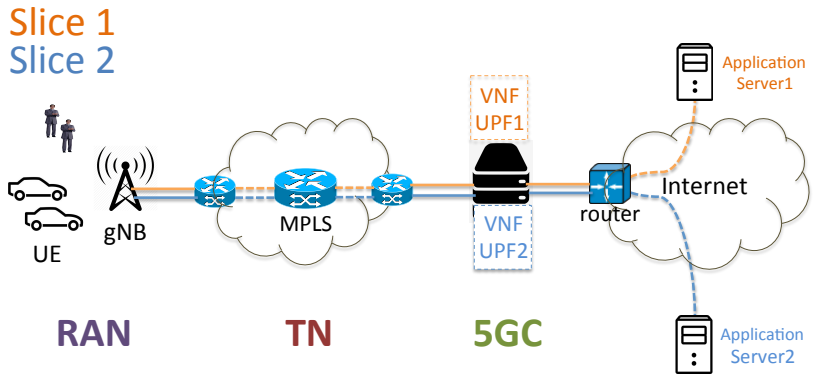


Figure 10: 5G Network slicing simulation model (user plane). Source: Figure 1 in [20]

The 5GC (5G Core) enables the dynamic creation of virtual UPFs (User Plane Function), which could be composed of one or more VNF (Virtual Network Function) instances, allowing measurement of compute resources consumption (CPU, RAM, Storage) and providing the capability to scale those instances vertically and horizontally. To handle multiple VNFs serving a network slice, we enhanced our previous simulator's load balancing function with a weighted round-robin mechanism. This mechanism distributes packets to VNF instances based on their relative resource capacity compared to the total capacity of all

VNFs serving the slice[2]. With the traditional Round Robin load balancer, when there is a high gap between compute resource limits among the VNFs utilized by a slice, if the packets are just sent alternatively to each VNF one by one, the VNF(s) with lower resource limits could easily get overloaded (which causes extra delay since the over-sent packets need to be queued in line or even worse get dropped leading to packet loss), while the ones with higher limit become under-loaded, hence the compute resource moderation task for the closed loop will be more challenging and difficult. With the weighted round robin mechanism, the VNFs in the long run will receive the workload (incoming packets) corresponding to their capacities (resource limits), thus the resource utilization will be more optimized and helps to yield better overall E2E KPI performance.

One of the most important features of the simulator is the capability of handling the following network actions during simulation time:

- Link scale Up/Down: Increase/Decrease slice's E2E link data rate capacity
- VNF Scale Up/Down (or VNF vertical scaling): Increase/decrease one or all of the compute resources (CPU, RAM, Storage) of a VNF instance.
- VNF Scale Out/In (or VNF horizontal scaling): Increase/Decrease a VNF instance.
- Slice assignment to Edge/Core: Re-assign a slice to the UPF at the core or edge location of the network

Despite the simulator's capabilities, the closed loop only implemented the first three types of actions (link capacity scaling, VNF vertical and horizontal scaling). Table 1 summarizes all the nine closed-loop actions implemented from those 3 action types. Each action has its unique algorithm (decision criteria and scaling formula) but has a few common types

[2] 1) Resource limits of serving VNF instances are normalized based on total resource limits of each resource type; 2) The dominant factor of each VNF instance is determined by selecting the maximum value among the 3 normalized resource types. 3) The VNF's probability to be selected is determined by summing the dominant factors of all VNFs and normalizing the values.

of parameters, such as action resource threshold limits, resources and KPI trends, etc.. which will be described later in section 3.3.4. The key difference between a re-active action and a pro-active one is: while a re-active action are specified (assigned) to solved a specific slice KPI violation and only applied to the violated slice, the pro-active actions are the way our closed loop attempts to look at current relevant resources/KPI status and trends to pro-actively allocate more or reduce resources, avoiding to lack or to have over-abundant resources for ALL slices. It's also be noticed that "VNF resources" and "compute resources" will be used interchangeably, and the term "links" used in our context refers to the E2E virtual links, where each link is associated with an E2E network slice.

Table 1: Implemented Closed loop actions

Area - Domain	**No.**	**Action**	**Proactive?** [Y/N]
Network resource (Links data rate limits) - Transport	1	Link data rate scale up	N
	2	Link data rate scale up proactive	Y
	3	Link data rate scale down	Y
VNF Compute resource (CPU/RAM/STORAGE) - Core	4	VNF scale up	N
	5	VNF scale up proactive	Y
	6	VNF scale down	Y
	7	VNF scale out	N
	8	VNF scale out proactive	Y
	9	VNF scale in	Y

Any resource has a maximum physical capacity (phy) and a configured capacity (cap), which is the "virtual" dynamic capacity assigned to that resource ($cap \leq phy$) and can be adjusted by closed loop actions, together with a current resource utilization (ratio between max resource usage over Tw and current configured capacity). Especially, for throughput resource (or links' data rate), each slice has its own virtual E2E link with a "hard" limit (Th^{phy}), while all of the slices share one physical link with a physical data rate limit (T^{total}) that the total sum of configured capacities of all individual links cannot exceed.

3.3.3 Orchestration

Our QoS assurance approach employs a closed-loop orchestrator tested over the simulated 5G network with slice components containerized as virtual network functions. The closed-loop aims to automate network actions mentioned in the previous section in a zero-touch manner, with a particular focus on 5G network orchestration in the Core domain (modulate VNF compute resources) and transport domain (Link capacity resources). The closed-loop algorithm handles these actions, as depicted at a high level in Figure 11.

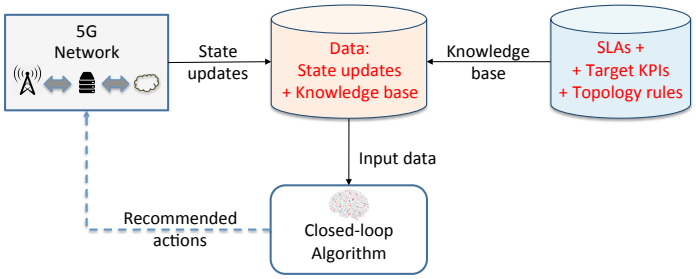


Figure 11: Closed-loop orchestration

3.3.4 Closed-Loop Algorithm

KPI violation status tracking

Although the closed-loop algorithm can be designed in event-driven mode, we have chosen a time-driven model for simplicity. The closed-loop is executed every time window (Tw), where multiple KPI readings or samples of the same KPI can happen inside Tw. A KPI violation is determined when the KPI's average value and/or max value over Tw exceeds a given KPI threshold specified for each slice type. To improve our ability to monitor KPI violations, we implemented a violation status tracking system that includes the following information related to KPI violations:

- **NEW:** KPI violation that occurs in the current time window Tw.

- **PENDING:** KPI violations that were not solved in previous Tw(s) (a violation either not having any closed loop action assigned yet, or a violation already having action(s) assigned in the past but still repeated again in the current monitored Tw).
- **LEVERAGED:** KPI violations that does not need to be directly solve, but they will be solved together with another higher prioritized NEW or PENDING violation. So those not-urgent violations are considered leveraged on that higher- priority violation.
- **ACTION_GIVEN:** KPI violations that had an action assigned and need to be monitored for some pre-defined Tw(s) to be fully closed if they are not violated again.
- **SKIPPED:** KPI violations that do not need to be solved yet (due to meeting certain skipping criteria, depending on whether they are NEW or PENDING, but still need to be monitored for some Tw(s) with the expect that they will not likely be violated again in the near future. The purpose to "skip" a violation is to avoid over-reaction from the closed loop to save actions and resources costs.
- **DONE:** KPI violations that have been confirmed to be solved successfully (not violated again during monitoring in some specific Tws).
- **INVALID:** SKIPPED violations that have not occurred again during monitoring Tw(s).

Violations with status **DONE** and **INVALID** are outside of the tracking scope of the closed loop, while the ones with remaining statuses are considered active (on-going) violations.

Figure 12 explains the logic of **SKIPPED** and **INVALID** violation by giving an example of how a newly occurred violation is skipped and later updated to INVALID. The left plot shows the max delay KPI on the down-link (DL) direction of a slice for the whole simulation duration (24 hours), and the right plot is the zoomed-in section of the simulation period from the 18^{th} to 20^{th} hours. The first violation occurs in the 4^{th} time window

from the 18^{th} hour: max delay in the Tw, 104ms $>$ delay threshold limit of that slice type (60ms). This violation meets the 'SKIP_NEW' criteria:

- There are no other on-going violation on the slice and direction.
- The violation was only a high peak with 1 violated data point over the 5-point Tw (20%), which meets the violation duration threshold of below 40%.
- The ratio between its max over average value in the Tw is higher than a specific threshold value X (X = 5).

Thus, it is updated as **SKIPPED** at the end of the violation occurring Tw. 3 time windows later, since that violation does not happen again, it will be closed as an **INVALID** violation (the monitoring period has expired). The second violation occurs later is outside of the monitoring period, so it would be considered a NEW violation and will be addressed independently. Another note is that the skipping logic for a **PENDING** violation ('SKIP_PENDING' criteria) would be much more complicated than skipping a new case that just occurred, as a PENDING violation might not have a short tall peak as the NEW one and the closed loop needs to look at a combination of other factors such as the accumulated trend, several ratio and outlier tests.

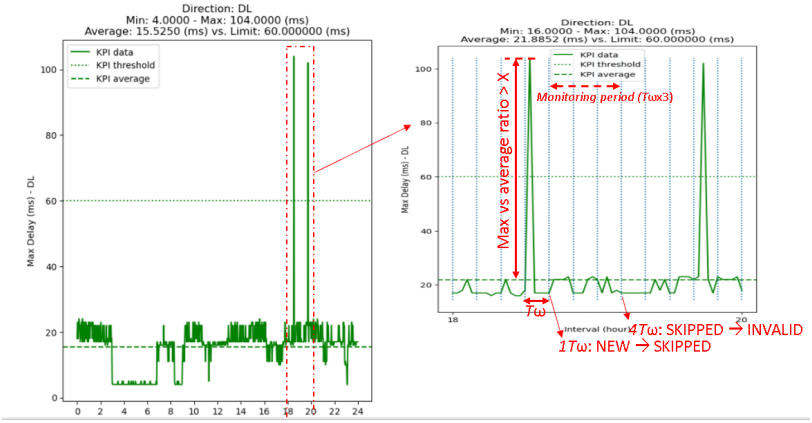


Figure 12: An example of a SKIPPED and then INVALID violation (coming from a NEW violation)

Besides those main statuses, we introduce an additional field called "*Repeat*" to capture re-current violations. When there is a new violation occurred on the same KPI, slice and direction with an on-going active violation, instead of capturing that new violation as a NEW case, we mark the on-going violation as "repeated" to indicate that it is a re-current violation and to keep track of the repeated count, which is an important factor in violations ranking and skipping decisions taken later on.

High level closed-loop description

Algorithm 1 (ProAct-Adaptive-CL) describes the high level closed-loop algorithm. $vTrack$ is the master KPI violation tracking table. Key information of $vTrack$ are listed on Table 2. For short, the term "*case*" is used interchangeably with "(slice KPI) violation".

Table 2: $vTrack$ table

Id	KPI	Slice	Direction	Status	Tw	LeveragedId	Repeat
*	*	*	*	*	*	*	*

where

- **Id:** Identity key assigned to any NEW entry (case).
- **KPI:** Key Performance Indicator, referring to the E2E KPI names. A KPI type can have more than one KPI, for example: KPI type 'Delay' has 2 different KPIs named 'max-Delay' and 'average-Delay'
- **Slice:** Identity key assigned to each Slice.
- **Direction:** Traffic direction, up-link (UL) or down-link (DL).
- **Status:** Violation status information mentioned above.
- $\boldsymbol{Tw}$: The time window that the violation first occurs.

- **LeveragedId:** Id of the KPI violation that the current violation is leveraged on.
- **Repeat:** (True/False) indicate recurrent KPI violation (on the same slice and direction) over multiple $\boldsymbol{Tw}$s.

Algorithm 1 High level Closed-loop (ProAct-Adaptive-CL)

Initialize: $vTrack \leftarrow \emptyset$
for every Tw **do**
  Input: KPIs, resources & thresholds, $vTrack$ **Output:** Closed-loop Action(s)
  Check for new KPI violations on slices in current Tw
  Phase 1:
    Check repeated cases from current **NEW** violations
    Check to skip **NEW** cases ▷ criteria SKIP_NEW
  Phase 2:
    Check to skip **PENDING** cases ▷ criteria SKIP_PENDING
    Close **ACTION_GIVEN** cases as **DONE** if passing monitoring periods without repeat
    Close **SKIPPED** cases as **INVALID** if passing monitoring periods without repeat
  Phase 3:
    Sort & rank $vTrack$ ▷ Sorting Alg.2
    Remaining violations of every slice are set to be **LEVERAGED** on each slice's top ranked case
    Perform link data rate & VNF resource analyses on all slices using data from (Tw - 3) to Tw periods and also in Tw
    Pre-assess and recommend re-active actions for the top violation of each slice based on previous analyses ▷ Recommendation Alg.3
    Evaluate and perform recommended actions if detailed conditions met ▷ Section 3.3.4
    Evaluate and perform pro-active actions on all suitable slices (slices having no violation or no specific action type(s) performed during Tw)
    Evaluate and perform specific action(s) for known historical issue(s) ▷ Alg.6

Updating violations list (Closed loop phases 1 & 2)

In the first two phases, we find the new KPI violations and register them in $vTrack$ with status **NEW** if they are not repeated cases. Else, any *Repeat* violation will have its status changed back to **PENDING** to be able to be evaluated again in current Tw. Then we update the status of the remaining existing violations, by checking if they can be LEVERAGED (i.e. if there are both average and max delay violations on a slice and direction, the maxDelay violation will be leveraged on the average-Delay case) or SKIPPED (if meeting SKIP_NEW or SKIP_PENDING criteria), and determining if a violation being monitored was successfully solved (DONE or INVALID) so that it can be removed from the table (by checking if they have any repeated occurrence during the monitoring period of 3 Tws), or if it needs further actions.

Sorting violations (Algo.1 line 13)

When multiple KPI violations occur, a challenge of the closed loop is to determine the appropriate sequence of actions to address them. This is because each slice type has its own priority, and KPI priorities can vary between them. To further complicate matters, when multiple slices require additional bandwidth capacity or VNF instances, prioritization is necessary due to the shared total physical link capacity or limited available VNF instances. For example, when deciding which slice should receive a link capacity increase or a VNF scale out, it is essential to prioritize the slice with the most urgent bandwidth-related violation or VNF resource-related violation. To address these challenges, our approach proposes to sort the KPI violations and determine which violations need to be addressed first, ensuring that the necessary actions are taken in the proper sequence.

The three key factors used to sort $vTrack$ are:

- **Slice priority (P_S):** Priority of each slice type (the smaller number the higher priority), or the type of traffic in a given slice. These priority numbers are selected from relevant application types from the 3GPP 5QI table (Table 5.7.4-1) in [4]).

- **KPI priority (P_K):** Indicates the priority given to each KPI type of the same slice type (the bigger number the higher priority). It works in conjunction with the slice priority. These priorities are also application specific and were partially selected from 5GPPP's 5G-XHaul project (Figure 3.2 in [6]). Table 3 summarizes all of the Slice and KPI priorities.

- **Risk Factor** ($\mathcal{F}_R$): The overall estimated risk of each KPI violation, based on several violation tracking attributes.

The sorting and ranking principles are described in Algorithm 2 ($PrioritiesAndRiskFactors$)

Table 3: Slice and KPI priorities

Slice		KPI		
Slice type	Priority	Delay	Jitter	Pkt. Loss
eMBB	50	2	2	4
mMTC	19	2	1	4
URLLC	30	5	4	4
VoIP	20	5	5	5

Algorithm 2 Sort $vTrack$: $PrioritiesAndRiskFactors$

Input: $vTrack$ **Output:** $vTrack_{sorted}$
Find $V \in \{vTrack : vTrack[status] \in \{\textbf{NEW, PENDING}\}\}$
Sort V by $\{P_K$ (descending), P_S (ascending)$\}$
$R_{KPI} \leftarrow \text{Rank}_{\text{dense-descending}}(P_K)$ over $\{P_S, Slice\}$
Calculate $\mathcal{F}_R$ for all $v \in V$ ▷ Risk factor, Eq.4
$R_{Risk} \leftarrow \text{Rank}_{\text{first-descending}}(\mathcal{F}_R)$ over $\{P_S, P_K, Slice\}$
$R_{Cycle} \leftarrow \text{Rank}_{\text{first}}(Slice)$ over $\{Slice\}$
$vTrack_{sorted} \leftarrow$ Sort V by $\{R_{Cycle}, R_{KPI}, P_S, R_{Risk}\}$

Only active **PENDING** and **NEW** violations from $vTrack$ are selected for sorting. The final result of the ranking is: when there are multiple KPI violations on multiple slices, directions and different KPI types, all of the most critical KPIs of each slice will be on top of the table ordered according to their slice and KPI priorities. In case a slice have multiple violations of the same KPI type, or when there is more than one slice with the same slice type and KPI violation, the risk factor $\mathcal{F}_R$ is used to determine which violation has higher priority. The ranking attribute R_{Cycle} will indicate which violations of each slice should be targeted to be solved in this Tw, while the rest of the violations with lower ranking will be leveraged on the top violation of its corresponding slice. By using a sort and rank technique and combining domain expertise, Slice and KPI priorities, with a risk estimation scheme through a risk factor, our approach ensures that all slices and their KPI violations are addressed in a reasonable order in the current Tw.

The risk factor is calculated based on several key attributes of KPI violations:

- **KPI intensity ratio (R_I):** Max value of the KPI measurement over its KPI threshold.

- **KPI duration ratio (R_D):** Accumulated total number of times (data points) a KPI was violated over the number of available data points from its first occurrence Tw until current Tw.
- **Number of leveraged violations (N):** Count of all cases being leveraged on the considered violation.
- **Period count (O):** Number of Tws from $vTrack$[status] = NEW to current Tw.
- **Repeat count (C_R):** Number of recurrences the violation happened again (without fully closing) from $vTrack$[status] = NEW to current Tw.

We utilize all of these attributes to calculate the risk factor using Equation 4 and the weighted product model (WPM), which is a widely used technique for solving multi-criteria decision-making (MCDM) problems [49] due to its simplicity and effectiveness:

$$\mathcal{F}_R = \prod_{a \in \{R_I, R_D, N, O, C_R\}} a^{w_a} \tag{4}$$

where a is the normalized value of the given metric (using min-max scaling within the range 1 - 10), and w_a is the exponent coefficient (or product weight) defined to modulate the relative importance of each metric. These weights can be chosen based on the operator's opinion about which metrics among the five are more critical. If the operator is unsure, they can select a default value of 1 for all coefficients.

Re-active actions recommendation (Algo.1 line 16)

Reactive actions to solve KPI violations are prioritized over proactive actions, as the former addresses immediate issues, while the latter is focused on preventing potential lack of resources or conserving resources. In phase 3, since it is assumed that only E2E KPI information is accessible and the primary goal is to minimize KPI violations resulting from

network or compute resource shortages, the preliminary assessment involves conducting initial analyses of current domain-specific resource utilization (such as links' throughput resource on Transport and VNF resources on Core). Based on these analyses, relevant domain action(s) from Table 1 are recommended for further evaluation and potential execution later, provided they meet the detailed criteria of each action type.

Algorithm 3 shows the recommendation logic for reactive actions (link scale up, VNF scale up and VNF scale out) to solve the top violation of each slices. For link capacity scale up, a threshold comparison is checked (compare the ratios between current max and average throughput over current link capacity configured of violated slice direction vs. two link scale up thresholds $\gamma_{\text{l\_up\_max}}$ and $\gamma_{\text{l\_up\_avg}}$). For VNF scaling, the decision of scaling up or scaling out for a slice is based on the combination of three factors: the need to increase capacity for a specific VNF resource due to nearly reaching the current configured capacity (n_R_{jk}), whether that resource is nearly reaching its physical capacity or not (c_R_{jk}) and whether the total resource cap of a resource type k on the slice reaches its total physical limit or not (r_R_k).

It is important to note that since the resource types of the two domains are independent, there is no dependency or constraint between the two domain action types. This means link scaling up and VNF scaling up/out can occur simultaneously on the same slice without any concern. However, when it comes to VNF scaling, if certain resources of a VNF instance need to scale up while others require scaling out, the decision to scale out will take priority over scaling up.

The recommendations for proactive scale-up actions are similar to reactive ones, with the main difference being that they are applied to all slices following the slice priority order, regardless of whether there is a KPI violation that triggers the action.

Algorithm 3 Recommend reactive closed loop actions

Input: $vTrack_{sorted}$, LinksAnalysis, VnfsAnalysis **Output:** Re-active Action(s) recommendation
Find $vT' \in \{vTrack_{sorted} : vTrack_{sorted}[R_{Cycle}] = 1\}$
for each v in vT' **do**
  for each domain $\in$ {Transport (T), Core (C)} **do**
    if domain $= T$ **then**
      if $avg(th) > \gamma_{\text{l\_up\_avg}}.th^{cap}$ & $\max(th) > \gamma_{\text{l\_up\_max}}.th^{cap}$ **then**
        Recommend checking link scale up on slice
    else if domain $= C$ **then**
      $n\_R_{jk} \leftarrow \max(R_{jk}) < \gamma_{\text{Vnf\_up}}.R_k^{cap}$
      $c\_R_{jk} \leftarrow \max(R_{jk}).(1 + \alpha_{\text{Vnf\_up}}) > R_k^{phy}$
      $r\_R_k \leftarrow \left(\sum_{l=1}^{N_i} R_{lk}^{cap} < \gamma_{\text{Vnf\_total\_up}}.\sum_{j=1}^{N_i} R_k^{phy}\right)$
      if $\exists r \in \{R_{jk} : n\_R_{jk}$ & c_R_{jk} & $r\_R_k\}$ **then**
        Recommend checking scale out on slice
      else if $\exists r \in \{R_{jk} : n\_R_{jk}$ & $\sim c\_R_{jk}$ & $\sim r\_R_k\}$ **then**
        Recommend checking scale up on slice

Where:

- th: throughput of the impacted slice direction
- $R_{jk,j=1..N_i,k\in\{\text{CPU, RAM, Storage}\}}$: Resource k of VNF$_j$ in N_i VNFs serving impacted slice s_i
- $\gamma_{\text{l\_up\_avg}}, \gamma_{\text{l\_up\_max}}, \gamma_{\text{Vnf\_up}}, \gamma_{\text{Vnf\_total\_up}} < 1$: constant ratio check thresholds ($\gamma_{\text{Vnf\_total\_up}} = 0.9$, for other 3 terms: details provided later in section 3.3.4)
- $\alpha_{\text{Vnf\_up}} < 1$: Default minimum VNF scale up step

Actions criteria check and execution

After all re-active action recommendations are assigned, the actions are not executed immediately but are checked by their own sub-algorithm for further constraints and to determine specific execution behaviors. All pro-active actions will be only checked and executed after all the re-active ones from the same period Tw have been executed.

Algorithm 4 illustrates a high level template of criteria checking and executing logic for scale up or scale out actions. The input sequence of slice checking for action conditions is executed either one by one followed by a sorted KPI violation in the re-active scenario, or by a sorted slice priority in the proactive scenario. The common checks performed for each slice / action (detailed criteria is different per action type) is as follows:

- If it is a proactive scenario, there should not be any active violation on the slice
- $NoPreviousActionsOnSlice$: Check if any related action has been performed on

a slice. For example, if a re-active scale up has been executed on a slice, the corresponding scale up or scale out action on that slice will not need to be in the same Tw.

- $ScalingThresholds$: threshold based comparisons to determine the max and average resource level against current resource cap during Tw would trigger the action or not (similar to the checks made in the recommendation step above, by comparing relevant max and average resource levels with the ratios $\gamma_{*\_up\_@}$ where $*$ is a wildcard for l [link] or Vnf and @ can be avg or max)

- $Trends$: Check the resource trend, either for the current Tw's data or the accumulated data over the last 3 periods or both (depending on action type). The purpose of using accumulated data in 3 Tws is to increase the sample size in the trend hypothesis test, thus to have a more confident level on the confirmed trends. Moreover, for link proactive scale up, since there can be up to 2 directions (UL/DL) the trend analysis should determine the dominant trend among both directions (for example for scaling up actions, the major concern is whether a resource in a given direction is uptrend or not).

All of the trend and slope analyses used in our closed loop leveraged the Mann-Kendall trend hypothesis test on monotonic sequence data (no seasonal and non-parametric version) with the Sen slope estimator [25, 36].

Analyses for both threshold levels, ratios, and trend information have already been performed at the beginning of phase 3 after the KPI sorting step for all slices (as shown in the high level algorithm 1), and will play a part in determining the scaling up amounts of the considered actions (details later in section 3.3.4

Algorithm 4 Scaling Up/ Scaling Out actions template

Input: Performed actions in Tw, $vTrack_{sorted}$, slice $s_i \in S_{sorted}$ (sorted by P_S) in consideration, $proactive$ (True/False)
Output: scaling up / scaling out action(s)
if ($proactive$ & $\nexists v \in \{vTrack_{sorted} : vTrack_{sorted}[slice] = s_i\}$)
or ($\sim proactive$ & $NoPreviousActionsOnSlice$) **then**
    if $ScalingThresholds$ check passed & $Trends$ check passed **then**
        if domain $= T$ & recommended_action$[s_i]$ = "link scale up" **then**
            Calculate new link capacity $t_i^{'cap}$ for s_i
            if $t^{'cap} \leq t^{phy}$ & new total link cap $\leq Th^{total}$ **then**
                Send link scale up request ▷ Scale up step calculation: Eq.8
        else if domain $= C$ **then**
            if recommended_action$[s_i]$ = "VNF scale out" **then**
                Select the next available (inactivated) VNF instance for scale out
            else
                Calculate scaling step(s) for scale up request & execute ▷ Eqs.5, 7

Scale Down/ Scale In actions

As the closed loop prioritizes minimizing violations over resource consumption, the conditions for triggering scale up / scale out actions are simpler than the rest. But for scaling down or scaling in actions, more precautions should be taken and the actions would behave differently according to different levels of resource utilization. Here is the principle (details explained later in this section):

- If a slice's current resource level is within the average range, i.e. not too high but not too low compared to the currently configured capacities, a resource scale-down may occur if the current KPI status on that slice allows it. The scale-down amounts are moderated based on whether there is any KPI in an uptrend and nearly reaching its threshold ratio.

- When resources are in abundance and their levels are significantly lower than the configured capacities, the closed loop can reduce relatively large amounts of resources without caring about the current slices' KPI status. This approach minimizes the number of actions required by the closed loop to optimize the resource levels.

Therefore, based on the principle described above, we have chosen to implement a two-threshold level check (high and low) for each resource type. Algorithm 5 describes a high level template for scale down/ scale in actions for all slices. The first and most important

condition for reducing resources is to check whether there are any active violations occurring on the slice. Additionally, the action under consideration must not conflict with any previous actions taken during the current time window (e.g., if a VNF scale-up has already occurred, a VNF scale-in or scale-out should not be allowed), which is checked using the $NoOppositeActionsOnSlice$ condition. Finally, a combination of three tests, including a two-level threshold and a trending test, is used to determine the appropriate scaling step for link or VNF scale-down as follows:

- $HighScalingThresholds$ & $LowScalingThresholds$ checks: similar to $ScalingThresholds$ check for scaling up/out, but applied to both high and low threshold levels following the above mentioned principle (comparing average and max of link or compute resources to low thresholds $\gamma_{*\_down\_@\_l}$ and high thresholds $\gamma_{*\_down\_@\_h}$ where $*$ replaces l or Vnf and @ replaces avg or max).

- $Trends$ check: check on resource trends on current Tw and also on the last 3 accumulated Tws (similar to scaling up/out). The generic rule is: if the max and average resource ratios are below their low thresholds, then no need to care about those resource trends (for link data rate) or $SliceKpiStatus$ check (for compute resource), but if the ratios are below the high threshold then they must not uptrend in both current and from the accumulated last 3 Tws.

- $SliceKpisStatus$ check: check on slice's, KPIs' levels and trends excluding throughput, each action type has different criteria details (combinations of 2 types of threshold levels and trends) and applies on compute resources scaling only. The thresholds used for slices' KPI checks are $\gamma_{\text{Kpi\_*\_max\_h}}$ and $\gamma_{\text{Kpi\_*\_max\_l}}$, where $*$ can be in or $down$. The rule for for slice KPI check is summarized as follows: i), if all KPIs not exceeded low thresholds or ii), if all KPI values are between low and high thresholds and no KPI is uptrend from the last 3 Tw (if uptrend, their estimated max increase

value should not reach the KPI limit) or iii), if any KPI exceeded high threshold and it is uptrend, its estimated max increase value should not reach the KPI limit.

The thresholds check combined with the trend check will also determine the default scaling step to be used to calculate the resource scaling amounts.

Algorithm 5 Scaling Down/ Scaling In actions template

Input: Performed actions in Tw, $vTrack_{sorted}$ **Output:** scaling down / scaling in action(s)
for all slice s_i in slices S (sorted by P_S) **do**
  if ($\nexists v \in \{vTrack_{sorted} : vTrack_{sorted}[slice] = s_i\}$) & $NoOppositeActionsOnSlice$ check passed **then**
    if $HighScalingThresholds$ check passed & $Trends$ check passed & $SliceKpisStatus$ check passed **then**
      if domain $= T$ **then**
        Perform link scale down with default low or medium scaling step depending on resource trend ▷ Following Eq. 9
      else if domain $= C$ **then**
        if $ScaleInResource$ check passed **then** ▷ Section 3.3.4
          Find the proper VNF instance suitable for VNF scale in
        else
          Perform VNF scale down on relevant VNF instances with proper (low or medium) scaling step following resource trend ▷ Eq. 9
    if $LowScalingThreshold$ check passed **then**
      Perform link scale down / VNF scale down action without caring about resource trend, using default high scaling step ▷ Eq. 9

VNF scale in resource check

The logic of $ScaleInResource$ check in algorithm 5 is slightly different than scale down checks. Specifically:

- The slice under consideration for scale in must have at least 2 VNF instances.
- Instead of comparing to fixed thresholds, each max compute resource k of all N_i VNFs serving the slice s_i are compared against their *relative ratio* X_{jk} of the scale in total threshold $\gamma_{\text{Vnf\_in\_total\_max}}$:

$$\max r_{jk} \leq X_{jk}.\gamma_{\text{Vnf\_in\_total\_max}}.r_{jk}^{cap} \; \forall k \in \{C, R, S\}$$

  where $X_{jk} = \frac{r_{jk}^{cap}}{\sum_{l=1}^{N_i} r_{lj}^{cap}}$ is the relative ratio between VNF$_j$'s current resource limit to the total resource limit of all N_i VNFs, and $\{C, R, S\}$ are 3 compute resources CPU, RAM and Storage.

- Besides those threshold checks, another test is performed to make sure that removing a VNF instance from a slice does not cause any lacking resource afterwards: The list of N_i VNFs is sorted such that the VNF that has the maximum number of resource types and is currently having lowest utilization among them goes on top; Then repeat the test starting from the first VNF instance until finding a VNF$_j$ satisfying: if removing the VNF$_j$ instance from the slice, the remaining total physical resources should be larger than $(1 + \alpha_{\text{Vnf\_up}})$ times of the current total resources, for all resource types $k \in \{CPU, RAM, Storage\}$:

$$\sum_{l=1, l\neq j}^{N_i - 1} r_{lk}^{phy} \geq (1 + \alpha_{\text{Vnf\_up}}). \sum_{m=1}^{N_i} r_{mk}$$

  where the factor $(1 + \alpha_{\text{Vnf\_up}})$ is a buffer defined to make sure the total physical resource after removing VNF$_j$ can still handle the resources needed, even if there might be some small increase of resource levels in the next Tw

Scaling steps calculation

Calculation schemes are different between action types and pro-activeness.

Re-active VNF vertical scaling steps For re-active VNF scale up, though there is no direct correlation between network E2E KPI data and VNF resources, in order not to have a too naive approach as just picking some fix fraction number as scaling step, we decided that the VNF resource scale up amounts would be formulated based on the changes or movement of the violated KPI. So, to calculate the compute resource k's total scale up amounts for all N_i VNFs serving an impacted slice s_i we use:

$$Q_{ik} = \left[\Gamma_i . \beta_i . \frac{n_v}{n_t} + (1 - \Gamma_i) \right] . x_k \tag{5}$$

where,

$$\Gamma_i = \begin{cases} 1, & \text{if } R_I\text{'s trend increases} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

and β_i is the Sen's slope (Mann-Kendall trend test that captures the rate of change of the violated KPI intensity ratios R_I, i.e. normalized ratio between KPI values in Tw over KPI threshold of the violated KPI) when the trend is found, n_v is the number of violated data points, n_t is the total number of time slots since violation appeared (so that the fraction $\frac{n_v}{nt} \leq 1$ takes into account the duration of violated data points over the whole violation period), x_k is the default scale up amount for resource k:

$$x_k = \alpha_{\text{Vnf\_up}}.(\sum_{j=1}^{N_i} r_{jk}^{phy} - \sum_{j=1}^{N_i} r_{jk}^{cap})$$

It should be noticed that all the terms Γ_i, β_i, n_v and n_t in equation 5 refers to data of the violated KPI on slice s_i as we tried to quantify that KPI's latest behavior, while x_k determines the remaining compute compute resource fraction based on its current limit utilization, i.e. the the gap between configured limit and physical limit. The meaning of equations 5 and 6 are that regardless of whether the violated KPI data is uptrend or not, the total scale up amounts for the impacted resource(s) for all N_i VNFs is a factor of the remaining available resource(s) for scaling (x_k), which is the default scale up step fraction ($\alpha_{\text{Vnf\_up}}$) of the mentioned resource limit gap(s); The only difference is: if there is no trend, the factor is 1, while when there is a trend the factor is $\beta_i.\frac{n_v}{nt}$.

After calculating the total resource scale up amount, this amount is distributed among N_i VNFs according to their current relative remaining available resource ratio(s), compared to the total remaining resources of all those VNFs. We can get the amount of resource k on each VNF$_j$ in slice s_i as:

$$Q_{i,j,k} = \min\{\frac{Q_i.rf_{jk}}{\sum_{l \in 1..N_i} rf_{lk}}, rf_{jk}\} \tag{7}$$

where rf_{jk} is the available amount of resource k that can be used to scale up VNF$_j$ on slice s_i: $rf_{jk} = r_{jk}^{phy} - r_{jk}^{cap}$.

VNF pro-active scale up and link scale up steps A general formula to calculate new resource configured limits for those types of scale up actions (both link and compute resources) on slice s_i:

$$r_{jk}^{\prime cap} = (1 + \alpha_{*\_up}).[(1 - \Gamma_{jk}) + \Gamma_{jk}.(r_{jk}^{cap} + \beta_{jk})], \tag{8}$$

where α_{*_up} is either α_{l_up} or $\alpha_{\text{Vnf\_up}}$ (default up scaling step for link and VNF), Γ_{jk} indicates whether the resource k is uptrend (1) or not (0), β_{jk} is the trend's slope of the resource. *Note*: For link capacity scale down, the link index j is the same as the slice index i (i.e. $i = j$) and there is only 1 type of resource k which is the link's throughput. Unlike the reactive VNF scale up, these proactive scale up formulas don't need to count for the change in any KPI value but only on the changes of the resource itself when it is uptrend (the term β_{jk} estimates the max resource increase in the next Tw in case the resource still keeps the same trend and average slope).

Furthermore, if the newly calculated limit exceeds the resource's physical limit, the new limit will be adjusted to the physical one.

Scaling down steps As mentioned before, the default scaling step used to calculate the new configured limits for the resource scale down actions depend on the current resource's threshold level and trend. Thus, the new resource limit is calculated as:

$$r'^{cap}_{jk} = \begin{cases} r^{cap}_{jk}.(1 - \alpha_{*\_down\_h}) & \text{if } T_{low} \\ r^{cap}_{jk}.(1 - \alpha_{*\_down\_l}) & \text{if } T_{high} \ \& \text{ up trend} \\ r^{cap}_{jk}.(1 - \alpha_{*\_down\_m}) & \text{if } T_{high} \ \& \text{ not up trend} \end{cases} \quad (9)$$

where $\alpha_{*\_down\_h}, \alpha_{*\_down\_m}, \alpha_{*\_down\_l}$ are the 3 default scaling down steps (high/medium/low), $*$ can be replaced by l (link) or Vnf, T_{low} is $LowScalingThresholds$ check passed and T_{high} is $HighScalingThresholds$ check passed (mentioned in algorithm 5).

Vertical Scaling steps during VNF horizontal scaling For horizontal scaling actions (scale out and scale in), besides the addition or removal of VNF instances, they also include scaling up/down those newly added or removed ones.

- VNF scale in: The VNF to be removed from the slice will reduce its configured resources to a minimal (inactive) state to save compute resource cost. Then the remaining VNFs serving the slice will have some additional scale up if needed to cover for the resources from the removed instances.

- VNF scale out: All current VNFs in the considered slice will be filled up to their full physical limits for the impacted resource(s) and the new VNF instance will be allocated a (pre-defined) sufficiently high resource amount (for example, 3 times of the initial minimum resources) so that it will not cause a network bottle neck.

Slice type specific treatments

Everything is in our hands Besides using different threshold levels check and multiple scale down steps, another important factor in our closed loop is that all types of thresholds and default scaling steps are fully configurable and adjusted differently for each individual resource, slice and action type.

Table 4 summarize all the parameters we defined for thresholds and default scaling steps that were mentioned in previous sections. The values listed on the four right columns are our first official set of values, with special treatments for each slice type highlighted in red. Those initial set of values came out from rational thinking and fined-tuning based on observation from the experimental process, which will be explained after we present some guidelines on how the values are selected (not just by random guessing) as following.

Rules of thumb to control / modify the special treatment parameters There is no single formula or algorithm to summarize the method to tweak the parameters in Table 4 to fit some certain need. However, based on the algorithm's logic and rational thinking (not just by random guessing) plus days (or even weeks) of tuning them by hand, at least we come up with a general recommendation or some guidelines so that even giving the table to a new user with not much experience in network, he or she can still have some level of control over the closed loop's behavior without setting non-sense values.

- In general, the max and average threshold check values for vertical scale up ($\gamma_{*\_up\_max}$ & $\gamma_{*\_up\_avg}$) must be higher than their high thresholds check for scale down ($\gamma_{*\_down\_max\_h}$ & $\gamma_{*\_down\_avg\_h}$) at least 10-20%, where that gap determine the "stable region" for resources and the closed loop doesn't need to check for and perform any scale up or scale down.

- The max resource scale up check ratios ($\gamma_{*\_up\_max}$) determine when the closed loop check for and trigger resource scale up action at the end of each Tw. If a specific resource of some slice type is known for frequently having sudden and huge increases, the closed loop should response to the resource level early enough, and we do that by setting a lower max ratio check threshold for that resource and slice type. Besides the thresholds, that resource type should also need more capacity buffer. Although both

of the resource or slice KPIs' increasing rate and the default scaling steps have impacts on the final scaling up amounts, the default scaling up steps (α_{*_up} or $\alpha_{*\_up\_pro}$) is still the dominant factor deciding the minimum scale up step (as shown in Eq.5 and Eq.8); so that resource type would need a larger default scaling step. Two consequences of bigger scaling steps are: First, the number of scaling up actions will be reduced (since it take less scale up steps to reach the same limit level with a bigger minimum step); Second, together with the lower threshold, bigger default steps makes the overall resource capacity utilization (ratio between configured limits vs physical limits) become higher, as the configured limits staying longer time at high levels (details explained on the performance metric section 3.3.5 later). Moreover, for compute resources, the quicker a resource limit reaches its physical limit, the more likely scale out action will happen (if that current total resource level on the slice is high enough and exceeds scale out threshold check) and will yield even more overall resource capacity consumption.

- On the contrary, if a slice type is known of having quite stable resource levels and the resources increase or decrease at a relatively slow rate, the scale up thresholds check can be set at a higher level so that the scale up check can be postponed a little bit later. And since the resource trend will not increase sharply, the default resource scale up steps can be set with smaller values than other slice types. Smaller scale up steps and slower response would give overall better (lower) resource cap utilization compared to when the scale up is performed early with a bigger step; however, there would be more scaling up actions performed.

- For scale down actions, similar principals are applied but on the opposite direction. When current resources are at very low percentage compared to their configured limits (how low is determined on the max and average resource scale down low thresholds), if the default scale down high steps ($\alpha_{*\_down\_h}$) is small, the first few scale

down steps will be small and it will take longer time and more steps (more Tw) to reduce resource limits until resources reach the high scale down threshold check level; thus, in general they should be set at big enough values so that the resource limits will reduce faster, for resource capacity saving purpose. To decide how "big" is enough, we can do some estimation to make sure that in the worst case the post-reduced configured limits will not reach too closed to the current resource levels, specifically after limit reduction the resource is at best staying in the stable region mentioned above and not fall into the scale up check region. Here is an example on how estimate the default high scaling step: At the end of period $Tw = t$, the max resource level is at below 20% compared to current limit (which is assumed our max scale down low threshold), then we reduce the limit by 60% (the scale down high step is 0.6) and the limit has 40% left. First scenario, in the next period $Tw = t + 1$, if the resource level does not increase , it will be at around 50% compared to the new limit (20% over 40%). Let's say the scale up check threshold is 85% and the max scale down high threshold is 60%. In this first scenario, if the resource is not uptrend (still stay at maximum 50% level) which is below 60%, then the next scale down would use the medium step and reduce the limit by 20% and resource level increased to 62.5% ($50\% \div 85\%$) and already in the stable range (60% to 85%), no scale down action will be performed in the next period $Tw = t + 2$. In another scenario when the resource is uptrend to 55%, the low scale down step would be applied, and if we select the low scale down step as 10%, the new resource level would be at maximum 61.1% ($55\% \div 90\%$). In worst scenario, in $Tw = t + 1$ resource might increase up to 15% then it will still be at around 87.5% ($[20\% + 15\%] \div 40\%$) and don't have a risk of lacking resource. Thus all of the selected scale down thresholds and steps in this example make sense. In conclusion, the combination of scale down high threshold and the low or medium scale down steps will determine how cautious (how early and

big) the scale down is performed when the current resource is not at a so safe level.

- Besides the max threshold ratios, the average threshold ratios ($\gamma_{*\_*\_avg\_*}$) is the second factor to determine the likeliness that the scaling action is happening. For scaling up, if the average threshold is set much lower than the max threshold, the combined condition is more likely to be met and the scale up action is easier to happen, because normally the average resource value would be much lower than the max value in the same time interval; So a low average threshold can be applied to increase the proactive level for resource types that have high variations. In contrast, if the average threshold is raised close to the max threshold, the average threshold check will be harder to satisfy, so it is suitable to apply for resource and slice types that has relatively stable resource trend (not much variation). Scaling down actions are on the opposite direction: the higher the average threshold the more likely the scale down condition is met, while setting a low average threshold would make the scale down action not easy to happen.

Applied those principles, the different values on Table 4 for each slice type are set based on some of its most noticeable characteristics according to ETSI (European Telecommunications Standards Institute) [18].

- For eMBB (Enhanced Mobile Broadband), this slice type requires very high peak data rate and its network's throughput level are always much higher compared to the other slice types; moreover, eMBB slices' traffics can suddenly increase sharply. That is why for link throughput scale up, we use lower thresholds to check for scale up action trigger conditions so that the closed loop can respond earlier, and the default scale up step is larger to create more buffer; For link scale down, the ratios used in high and low thresholds checks are also lower than other slice types to make it safer when deciding the link scale down action, and even when the throughput is at a

low threshold level, the default high scale down step is also smaller to avoid over-reduction of throughput cap, which might badly impact the next time period.

- For URLLC (Ultra-Reliable Low Latency Communications), most applications of this slice type are delay critical or are very sensitive to delay. On the one hand, in our simulator's VNF delay model the utilization of VNFs plays a big role in contributing to max E2E delay, that is why from the table, for most of the VNF scaling related parameters, this slice type always requires higher scale up steps, earlier scale down threshold check (smaller ratios), and safer scale down and scale in thresholds. On the other hand, URLLC traffic does not require as high throughput bandwidth as other slice types, so it does not need any special treatment on link data rate scaling.

- For mMTC (Massive Machine Type Communications), with the high density of IoT (Internet of Things) devices over the network, the overall energy consumption should be low to increase network energy efficiency. At the same time, it is also known that IoT devices and applications do not consume as much compute resources as other slice types. Thus, the VNF high scaling down step ($\alpha_{\text{Vnf\_down\_h}}$) for this slice type is selected to be bigger so that they can help reduce compute resources more aggressively.

Special rules for known historical issues (Algo.1 line 19)

In addition to the generic rules for reactive and proactive actions, our closed loop also allows for the addition of customized rules for specific KPI violations of certain slice types that arise from acknowledged issues with a pre-defined set of actions to address them.

In order to maintain a record of identified issues and corresponding customized response actions, we define a table known as the Response Flow Checklist(s) (RFC), as shown in Table 5.

Table 4: Special treatments for each slice type, action type and resource type (Configurable) - Original official version

Action type / Resource	**Parameter**	**eMBB - slice1**	**URLLC - slice3**	**mMTC - slice2**	**VoIP - slice4**
Link data rate scaling (Throughput - TP)	MAX_TP_SCALE_UP_TRIGGER_RATIO ($\gamma_{l\_up\_max}$)	0.7	0.8	0.8	0.8
	AVG_TP_SCALE_UP_TRIGGER_RATIO ($\gamma_{l\_up\_avg}$)	0.4	0.5	0.5	0.5
	DEFAULT_LINK_SCALE_UP_STEP (α_{l_up})	0.2	0.1	0.1	0.1
	MAX_TP_SCALE_DOWN_CHECK_RATIOS High/Low ($\gamma_{l\_down\_max\_h}$ / $\gamma_{l\_down\_max\_l}$)	0.5 / 0.2	0.6 / 0.3	0.6 / 0.3	0.6 / 0.3
	AVG_TP_SCALE_DOWN_CHECK_RATIOS High/Low ($\gamma_{l\_down\_avg\_h}$ / $\gamma_{l\_down\_avg\_l}$)	0.3 / 0.1	0.4 / 0.2	0.4 / 0.2	0.4 / 0.2
	DEFAULT_SCALE_DOWN_STEPS Low/Med/High ($\alpha_{l\_down\_l}$ / $\alpha_{l\_down\_m}$ / $\alpha_{l\_down\_h}$)	.05/.1/.15	.05/0.1/0.2	.05/.1/.2	.05/.1/.2
VNF resources scaling (CPU/RAM/ Storage)	DEFAULT_SCALE_UP_STEP (α_{Vnf_up})	0.1	0.15	0.1	0.1
	DEFAULT_SCALE_UP_STEP_PROACTIVE ($\alpha_{Vnf\_up\_pro}$)	0.1	0.2	0.1	0.1
	SCALE_UP_CHECK_MAX_RATIO (γ_{Vnf_up})	0.8	0.7	0.8	0.8
	SCALE_DOWN_CHECK_MAX_RATIOS High/Low ($\gamma_{Vnf\_down\_max\_h}$ / $\gamma_{Vnf\_down\_max\_l}$)	0.6 / 0.3	0.5 / 0.2	0.6 / 0.3	0.6 / 0.3
	SCALE_DOWN_AVG_RATIOS High/Low ($\gamma_{Vnf\_down\_avg\_h}$ / $\gamma_{Vnf\_down\_avg\_l}$)	0.4 / 0.1	0.3 / 0.1	0.4 / 0.1	0.4 / 0.1
	SLICE_KPI_SCALE_DOWN_CHECK_MAX_RATIOS Low/High ($\gamma_{Kpi\_down\_max\_l}$ / $\gamma_{Kpi\_down\_max\_h}$)	0.7/0.85	0.7 / 0.8	0.7 / 0.85	0.7 / 0.85
	SLICE_KPI_SCALE_IN_CHECK_MAX_RATIOS Low/High ($\gamma_{Kpi\_in\_max\_l}$ / $\gamma_{Kpi\_in\_max\_h}$)	0.7/0.85	0.7 / 0.8	0.7 / 0.85	0.7 / 0.85
	DEFAULT_SCALE_DOWN_STEPS Low/Med/High ($\alpha_{Vnf\_down\_l}$ / $\alpha_{Vnf\_down\_m}$ / $\alpha_{Vnf\_down\_h}$)	.05/.1/.15	.05/.1/.15	.05/.1/.2	.05/.1/.15
	SCALE_IN_TOTAL_MAX_THRESHOLD (relative) ($\gamma_{Vnf\_in\_total\_max}$)	0.85	0.8	0.85	0.85

Table 5: RFC table

Id	KPI	Status	SliceType	RfcName
*	*	*	*	*

where

- **Id**: Id of the RFC record
- **KPI**: The violated KPI that need to be matched for RFC execution
- **Status**: The KPI violation status to be matched for RFC execution

- **SliceType**: Slice type to be matched for RFC execution
- **RfcName**: The name of the RFC, which outlines to the specific steps to be checked and executed to address the issue

Algorithm 6 summarizes the RFC check process, which involves checking each matching RFC using the input violation's KPI, slice type, and status. For each matching RFC, additional case-specific information is taken into account to determine if the RFC's specific conditions are met. If these conditions are satisfied, it confirms that the current violation matches the criteria of the known issue, and the set of special actions defined by the RFC name will be executed.

Thanks to the historical violation records, the closed-loop system can identify conditions that require information from previously archived violations, further enhancing its ability to customize the response to network issues. This reduces the need for manual repetitive efforts by operators to handle known historical network issues that have already been resolved in the past, but may have only had an interim solution without being completely resolved. By automating the response to these known issues, the system reduces the need for manual intervention, freeing up operators to focus on identifying the root cause of the problem and preventing its recurrence in the future.

Algorithm 6 AutoRFC - Check for known historical issues

Input: $vTrack$, RFC **Output:** Suggested checklist(s) to be performed
for all violation v in $vTrack$) **do**
    if $\exists r \in \{RFC : v$[KPI]=r[KPI] & v[Status]=r[Status] & v[SliceType] $= r$[SliceType]$\}$ **then**
        Check and execute checklist r[RfcName](v)

As a demonstration of the RFC check capability, we developed a simple RFC to address a known issue related to the initial configured limits of certain simulation scenarios. The details of this RFC and its associated results are discussed in Section 3.4.

Summary of Closed Loop Actions Flow

The sequence of closed loop actions can be summarized as shown in Figure 13. The diagram outlines the general order in which the actions are checked and executed every Tw, while keeping in mind that each action will only be executed if its specific conditions are met.

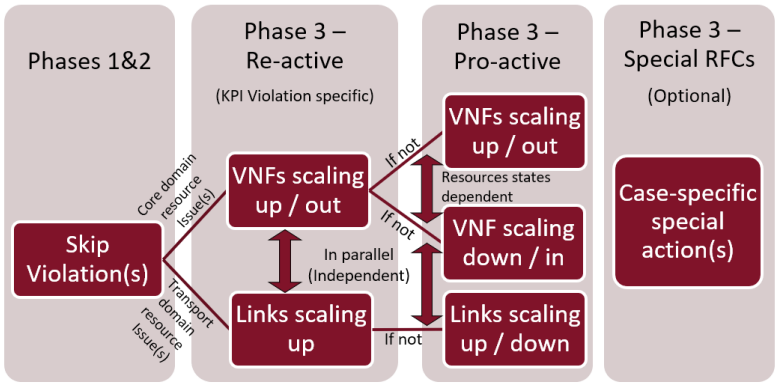


Figure 13: Closed loop actions flow every Tw

3.3.5 Performance metrics

There is no existing framework to specifically evaluate the impact and performance of a 5G quality assurance closed loop algorithm, so first we designed some indicators to measure our closed loop across three major areas (slice KPI violations, resources utilization and actions summary), with two major indicator types: simple aggregation indicators (or first-type indicators) and scoring indicators and then based on them to derive some higher level summarized indicators.

Slice KPI violation(s) summary

Gives an overall look on KPI violations on slices. Each of the below parameters are calculated per slice direction and KPI basis, then they are aggregated over each slice and then across all slices:

- **Total violations count**: Sum of all KPI violations on each slice.
- **Invalid violations count**: Sum of all INVALID violations on each slice.
- **Max intensity ratio**: the max of max KPI intensity ratios across all slices (max KPI violation value vs its KPI threshold)
- **Average & Max violation duration (%)**: violation duration is the ratio (in %) between the total violation points per KPI and slice direction over total simulation period; Then take the max and average numbers across slices.
- **KPI violation score**: Summarize intensity ratio and max violation duration of all slices into a single number for convenience when doing comparisons. Each KPI violation type for each slice will be calculated with a score normalized from 0 to 10 (using min-max scaling method), averaged from its max intensity score and max duration score, where the best score (10) is when there is no violation of that KPI type on the slice, and worst score (0) is the worst intensity ratio and the worst duration percentage across all 4 scenarios. Then the KPI scores of all KPIs will be aggregated by the weighted sum proportional to the KPI priorities of each slice into the slice's overall score. All the slices' scores then will be aggregated again by their slice priorities complemented to 100 (use "$100-$KPI priority" to take weighted sum of the slices' scores) into a single KPI violation score. Since the max aggregation in the previous first-type indicators cannot not reflect when there is no violation on a slice, while this scoring method can (and even better with KPI and slice priorities taken into account), the score would give a better estimate on how good or bad the KPI violation status of the whole simulation is.

Resource utilization summary

VNF utilization summary There are two indicators:

The first indicator is **Overall VNF resource utilization (%)**, which is the average of the 3 total compute resources utilization (CPU, RAM and Storage). Each resource type utilization is calculated by the ratio of total sum of all VNFs' weighted average configured limits over the total physical limits of all VNF instances, assuming the resource pricing scheme is applied to resource configured capacities used (regardless of actual consumption on those capacities).

$$U_{\text{Vnf\_overall}} = \frac{\sum_{k \in \{\text{CPU, RAM, Storage}\}} U_{k\_total}}{N_k} \tag{10}$$

where $N_k (= 3)$: Resource type count, and U_{k_total} is each resource type k's total utilization across all VNF instances:

$$U_{k\_total} = \frac{\sum_{j=1}^{N_{\text{Vnf\_total}}} \bar{r}_{jk}^{cap}}{\sum_{l=1}^{N_{\text{Vnf\_total}}} r_{lk}^{phy}} \tag{11}$$

where $\bar{r}_{jk}^{cap}$ is the weighted average configured limit during the whole measured simulation time of resource k belonging to VNF_j (detailed formula in next part), and $N_{\text{Vnf\_total}}$ is the total number of VNF instances in the network.

The second indicator is **Overall VNF resource score**: it scores each individual resource's overall utilization (U_{k_total}) against their minimum and maximum possible range (10 is the best score if all VNFs are at the minimum allowable limit all the times, and 0 is the worst case when all VNFs staying at physical limit of that resource type from the beginning to the end), then take the average of the 3 resources' score. Since each resource type has different min and max resource range, an average between the 3 normalized score is better than just simply taking the average between the 3 types of compute resources' utilization like the way "**Overall VNF resources utilization (%)**" is calculated.

Link utilization summary : There are also two indicators.

The fist indicator is **Overall link data rate utilization (%)**, which is the sum of the

total weighted average of each link's configured limit over the total physical capacity shared among all slices (Th^{total}), assuming the max link data rate capacity pricing scheme is similar to the compute resources scheme.

$$U_{\text{Link\_overall}} = \frac{\sum_{i=1}^{N_\text{S}} \bar{r}_i^{cap}}{Th^{total}} \tag{12}$$

where N_S is the number of slices ($N_\text{S} = 4$ in our experiments) and $\bar{r}_i^{cap}$ is the weighted average link data rate configured limit of slice s_i.

Both link data rate and compute resources' weighted average utilization per link or VNF instance per resource type k are calculated using the following formula:

$$U_{Rjk} = \frac{\bar{r}_{jk}^{cap}}{r_k^{phy}} = \frac{\sum_{i=1}^{n_{\Delta k}} \left(r_{jki}^{cap} . \Delta_{t_i}\right)}{T} . \frac{1}{r_k^{phy}} \tag{13}$$

, where $n_{\Delta k}$ is the number of limit changes of resource type k over the measured simulation time T (either the whole simulation or any specific time interval), Δ_{t_i} is the duration of each individual configured limit level r_{jki}^{cap} of resource type k on VNF/link instance j, where that duration is measured from the current limit change event to the next one on that resource. Equation 13 also shows the calculation for $\bar{r}_{jk}^{cap}$ used in the two previous equations 11 and 12 ($\bar{r}_i^{cap}$ in Eq.12 is the same with this $\bar{r}_{jk}^{cap}$), which is the resource's weighted average configured limit over the whole measured simulation duration T, and dividing that number to the physical limit r_k^{phy} yields the overall utilization ratio of the resource on the link or VNF instance (U_{Rjk}).

The second link resource indicator is "**Overall link resource score**", which takes the average of all the slices' link resource scores, where each slice's link score is also normalized from 0 to 10 of its overall utilization (the higher score the better utilization) based on the min and max possible link data rate limit range.

Overall resource score The final and single indicator to summarize the resources consumption of the simulation is the average of the **Overall VNF resource score** and the **Overall link resources score**.

$$\text{score}_{resource} = \frac{\text{score}_{link} + \text{score}_{VNF}}{2}$$

The two score types are calculated using normalized values and are on the same scale, which makes averaging them a valid operation. However, for a more detailed analysis, it may be useful to examine each individual link and resource score type separately. Nonetheless, a single score provides a convenient and easy-to-read summary of the overall performance at a higher level.

Closed loop actions summary

Closed loop actions are also a source of energy consumption or cost, but they need further research to estimate the cost aspect of actions. So the current indicator in this area is the **Total actions count**, which is the sum of all closed loop actions applied on each slice. Other variations can be developed, such as immediate average actions per hour or overall actions count per hour per action types, depending on the needs and the cost or pricing scheme applied. This actions count indicator does not have a scoring counterpart.

Indicators and metrics example

Table 6 shows an example of the performance metrics summary of a simulation without closed loop actions on a known special bottle neck network stage, which causes a lot of critical KPI violations across all slices (further details on the "Simulation Results" section 3.4). From the table it can be seen that: i) The highest violation peak among all slices is 77 times compared to its KPI limit ("Max intensity ratio"); ii) On average a KPI on a slice direction would violate for a cumulative duration of up to 25.8% of the total simulation

time, or about 6 over 24 hours ("Average violation duration") and the max KPI violation would take up to nearly 50% or nearly 12/24 hours (Max violation duration); iii) Thus, this is one of the most severed KPI violation of all scenarios, and the very low "KPI violation score" (1.6 out of 10) reflect that. iv) On the resources side, due to the very small initialized resource limits but not the smallest possible VNF limit, the overall VNF utilization is low (only 23.15%) and yielded a very good VNF score of 8.96, while link data rate utilization is as low as the slices' lowest levels so link score got a perfect 10, making the overall resource consumption score as good as 9.48; v) Finally the "Total actions count" is 0 because this is a simulation without closed loop actions enabled.

Table 6: Example of a simulation's performance metrics (without closed loop action)

Scenario description	Total violations count	Invalid violations count	Max intensity ratio	Average violation duration (%)	Max violation duration (%)	KPI violation score (0 - 10)	Overall VNF resources utilization (%)	Overall VNF resources score	Overall links data rate utilization (%)	Overall link resources score	Overall resources score	Total actions count
Bottle-neck - No action	55	48	77.00	25.88	49.58	**1.60**	23.15	8.96	6.84	10.00	**9.48**	0

Despite some better aspects of the scoring scheme versus the first type aggregated indicators, there is a limitation in the scoring approach versus the first type: the score only reflects the comparison against the relative best and worst cases of our known scenario settings (i.e. resource capacities and KPI limits selection, network dimension) & network traffics, so if there is a new scenario with worse performance, all the current scores must be revised (re-normalized according to the new min and max values). It is easy to set absolute minimum ratio values for each metric category, but there is no absolute max ratios since they depend on the selection/design of those known scenario settings. So if other researchers want to leverage our scoring method to compare their closed loop algorithm with this work, but do not use exactly the same simulation settings as us, the score comparison between different sets of scenarios with different physical resource capacities or violation performance will not be so meaningful. In contrast, the max and average aggregations of the first-type indicators can be used to directly compare "visually" between different

scenarios and closed loop versions (though it only partially reflect the whole simulation performance, and it also still required the same physical resource capacities and KPI limits for the proper comparison).

Furthermore, all those indicators mentioned above are designed in a way that they can be viewed both online (for example, performance for the last X hours) and offline (for any desired interval after simulation completes). Those current performance metrics can help service providers assess some high level trade offs between some tolerance of KPI violation and operational costs (resources or energy consumption). Moreover, when applying a proper pricing or cost bracketing technique (similar to the way popular cloud providers do nowadays, such as applying a price for every x thousand of actions per action type, or pay-per-use price of resource capacities registered, plus applying penalty fees for providers due to KPI violations), the enhanced metrics can be used as a benchmark to compare different quality assurance closed loop algorithms having the same goal and/or to translate everything into a final single operation cost number. Additionally, they can be even fine-tuned to form an appropriate cost function in case we want to apply a Reinforcement learning approach.

Closed loop stable (converged) state:

To evaluate the stability of the closed loop (i.e how its performance varies with different initialized resource limit settings), we will run the simulation through several different scenarios, each has different sets of initialized resource limits (scenarios settings described in the next sections). With the same traffics going through the network slices, and regardless of the initial resource limit settings of all scenarios, we expect that after some time from the simulation start, all the set limits modulated by the closed loop algorithm should reach to a similar stage that is optimized (fit) to the incoming traffic. So we have additional evaluations and statistics to check for:

- **Time to converge:** How much time the does the closed loop take (in the worst case) to bring the configured limits to the so-called “converged” state (i.e. In the most abundant resource state, how long does it take for the limits to reach the optimal stage.

- **Post convergence “violation score variations” and “resource score variations”**: These two indicators summarize the variations among all scenarios’ violation and resource scores after the simulations reach the “converged state”. These variations are calculated as the "mean $\pm$ standard deviation" of each score type (KPI score and overall resource score) across the scenarios. For instance, with the same set of closed-loop special treatment settings (from Table 4), the closed loop will run through four different scenarios with varying initialized resource limits. Each completed scenario will produce one set of KPI and resource scores, such as those presented in Table 6, with the difference that the measurements will start to be captured from the observed average converged time of the specific closed-loop settings set. Then, the performance variations of that particular set of closed-loop settings will be evaluated by computing the mean and standard deviation (std_dev) of the four KPI violation scores, representing the variation in KPI violations, and similarly, the mean and std_dev of the four overall resource scores, representing the performance variation for the resources across all four scenarios.

Figure 14 provides an illustration of the variations of the KPI score, overall resource score, and hourly action rate across all scenarios in the absence of closed-loop actions. The variations are represented in 3D space as a dot for means and lines for standard deviations across those three aspects of each single set of closed-loop settings. The variation in KPI violation scores is 5.56 ± 4.31 (the text parallel with the "KPI score" axis), and the variation in resource scores is 5.82 ± 2.76 (the text parallel with the "Resource score" axis). As there is no closed-loop action, the score is not satisfactory in terms of both mean and standard

deviation. However, these scores are heavily dependent on the initial limit settings and can be either good or bad. The hourly action rate variation is 0 (the text parallel with the "Hourly action rate" axis).

In the results section, the performance variations summary for simulations with closed-loop actions will be presented with vertical bars representing the mean hourly action rate, and vertical lines representing the standard deviation of the hourly action rate. The hourly action rate is calculated as the total count of actions in the simulation divided by the record duration. The dot in the plot will be at the height of the corresponding bar to show the relationship between all three aspects (KPI, resource, and action rate) in 3D space.

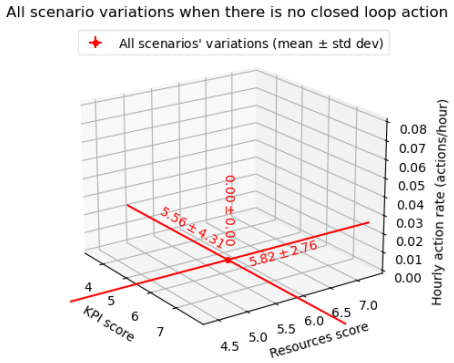


Figure 14: Example of variation summary across all scenarios in the absence of closed-loop actions. The dot represents the mean of KPI score (x-axis) and overall resource score (y-axis), and the lines represent their respective standard deviations. The colored text displays the mean and standard deviation of the relevant score type.

Fine-tuning the special treatment values given to the parameters on Table 4 will impact those variations and converging time as explained in the previous guidelines.

Closed loop's flexibility

In addition to evaluating performance based on statistical metrics, it is crucial for a closed loop algorithm to be customizable to meet the specific needs of network operators. This section demonstrates that modifying the parameters listed in Table 4 can have a significant impact on the overall slice KPI and resource scores, as well as those scores of each specific

slice type, within a manageable range. It can also affect the total action count or hourly average action rate. By creating and testing various parameter sets, we can evaluate the impact on the closed loop's converged time, post-convergence variations in KPI and resource scores, and average hourly action rate variation. All parameter sets will be displayed in a 3D plot, similar to the plot shown in Figure 14, to facilitate comparison between different settings.

Summary of performance metrics

In summary, Table 7 presents a comprehensive list of all the metrics described, including their scope and metric types.

Table 7: List of performance metrics

Scope	**Area**	**Metric**	**Metric type**
Single Simulation	KPI violation	Total violations count	Type-1
		Invalid violations count	
		Max intensity ratio	
		Average/Max violation duration (%)	
		KPI violation score	**Scoring**
	Resource capacity utilization	Overall VNF resources utilization (%)	Type-1
		Overall links data rate utilization (%)	
		Overall VNF resources score	
		Overall link resources score	
		Overall resources score	**Scoring**
	Closed loop action	Total action count	Type-1
		Hourly action rate	
Cross simulations	Closed loop flexibility	Converged time	Type-1
	Performance variation	KPI score variation	Scoring
		Resource score variation	
		Hourly action rate variation	

3.4 Simulation results

3.4.1 Settings

Having a packet level simulation network similar to Figure 10, we selected four end-to-end KPI types (five total KPIs as in Table 9) and six VNF related KPIs (max and average CPU, RAM and Storage) of eight total VNF instances, considering a simple setting of four slices (each slice serves one type of service per slice type as shown in Table 8) and traffic flowing in both directions (Uplink, Downlink) we could be monitoring up to 88 KPIs ($5 \times 4 \times 2 + 8 \times 6$). Not all of them are included in the final QA goal (required to be assured within the guaranteed limits).

Table 8: Slices and descriptions

Slice#	**Slice Type**	**Slice Description**
1	eMBB	Vehicles to Infrastructure (V2X), Cars and Pedestrians request HD video streaming (DL only)
2	mMTC	IoT devices(Static User and Trucks) transmitting small data frames at given time intervals (UL only)
3	URLLC	Static Users Low Resolution (LR) Gaming video with stringent URLLC target latency constrains (DL & UL)
4	Voice	VoIP model (In between the 3 main usage scenarios) (DL & UL)

Table 9 summarizes the E2E KPI thresholds (limits) for each KPI and slice type. There is no single source containing all the threshold values for those required KPI types, so they are collected from multiple references. For experimental purposes, we applied scaling factors to those limits to meet the capabilities of the network simulator used. The network dimensioning (i.e. slices' traffic) and those final KPI limits (after applying scaling factors) were selected in a way that even in the max network and compute resource configurations, there are still some max KPIs violated with short-period peaks. Thus, as mentioned in the goal in section 3.3.1, the closed loop will not try to reduce or resolve those unavoidable

violations, but focuses on preventing potential violations due to lack of resources and then reduces the resource consumption if possible.

Table 9: End-to-end KPI limits and scale factors

QA goal? [Y/N]	KPI Type / Unit	KPI	Slice type							
			EMBB		mMTC		URLLC		VoIP	
			Threshold	Scale Factor	Threshold	Scale Factor	Threshold	Scale Factor	Threshold	Scale Factor
Y	Delay ms	Average Delay / Max Delay	300	0.20	10	2.5	$30_{(i)}$	2	100	0.25
Y	Jitter ms	Jitter	100	0.012	$N/A_{(iv)}$	N/A	$5_{(iii)}$	1.05 (UL)/ 1 (DL)	$10_{(iv)}$	0.2 (UL)/ 06 (DL)
Y	Packet loss %	Packet loss	1E-04	1E+04	1E-02	285	$0.1_{(ii)}$	10	1.00	2.5
N	Throughput Kbps	Throughput	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Sources: (i): Table 3.1 from [5]; (ii): Table 3 from [47]; (iii): Tables 14 from [16]; (iv): Table 3 & 10 from [7]

3.4.2 Experiments

Experiment scenarios

To conduct a comprehensive evaluation of our quality assurance closed loop algorithm, we have devised four distinct scenarios. Each scenario is a combination of two domain resource types, namely link throughput and compute resources, where each resource type has two different sets of initialized resource configurations. We will compare the closed loop actions enabled versus disabled for each scenario, resulting in a total of eight cases. Each case will be executed at least two times, and we will select the best result to represent that particular case. The performance metrics described in previous sections will be utilized to compare and evaluate each scenario.

Table 11 provides a summary of the four scenarios, along with our expected outcomes for the closed loop algorithm. On the other hand, Table 10 lists the configuration details of the two domain resource settings (two sets of initial settings for each domain) used in Table

11. It also includes the minimum and maximum allowable scaling limits for each instance of link or VNF that the closed loop utilizes.

Table 10: Link data rate & VNF resource configs

Area	**Min allowable limits (scaling only)**	**Physical limits**	**Initial Limits / Configs**
Link data rate (Mbps)	Slice 1/2/3/4: 140 / 40 / 50 / 30	Slice 1/2/3/4: 512 / 256 / 256 / 256	***Link Settings 1***: 100 / 20 / 10 / 10 ***Link Settings 2***: 512 / 256 / 256 / 256
VNF resrouces (CPU: unit RAM: MiB Storage: MiB)	CPU / RAM / Storage: 0.3 / 128 / 128	CPU / RAM / Storage: 1.3 / 1024 / 1024	***VNF Settings 1***: - Slice to VNFs mapping: {1:[0], 2:[1], 3:[2], 4:[3]} - VNF[0..3]: 0.5 / 512 / 512 - VNF[4..13]: 0.3 / 64 / 128 ***VNF Settings 2***: - Slice to VNFs mapping: {1:[0, 4, 5], 2:[1, 6], 3:[2], 4:[3, 7]} - VNF[0..7]: 1.3 / 1024 / 1024 - VNF[8..13]: 0.3 / 64 / 128

Configs "***Link Settings 1***" set the network in a bottle neck stage that puts all slices into high KPI violations due to the lack of link data rate resources, as shown in Figure 15 where the top plots (no bottle neck) show the link capacity of each slice in a sufficient configured limit and normal levels of packet loss, while the bottom plots show minimal link configured limits that cause extremely high and long periods of packet loss across all slices and link directions. It can also be noticed that our minimum allowable link data rate limits are higher than the initial configured levels as the workaround so that the closed loop will not bring the network to the bottle neck state again in the future, at the expense of some link bandwidth resources. And since the current implementation of dynamic link data rate only controls the immediate data rate of each individual packet, it does not guarantee that the upper configured limits cannot be exceeded. Our future work might apply a proper method

Table 11: Simulation scenarios to evaluate closed loop's performance (Compare between no closed loop v.s. with closed loop actions)

Scenario	Initial Link Configs	Initial VNF Configs	Expectation / Goal on KPI violations / Compute & Network Resources
1	*Link Settings 1* (bottle neck)	*VNF Settings 1* (min)	Closed loop solve bottle neck problem Not use much more compute & network resources
2	*Link Settings 1* (bottle neck)	*VNF Settings 2* (max)	Closed loop solve bottle neck problem Save more compute resources Not use much more network resources
3	*Link Settings 2* (no bottle neck)	*VNF Settings 1* (min)	Closed loop yields better KPI violation score Save more on network resources Not use much more compute resources
4	*Link Settings 2* (no bottle neck)	*VNF Settings 2* (max)	Closed loop don't cause any extra KPI violations Save more on both compute & network resources

to control link's bandwidth, such as using the queue [13].

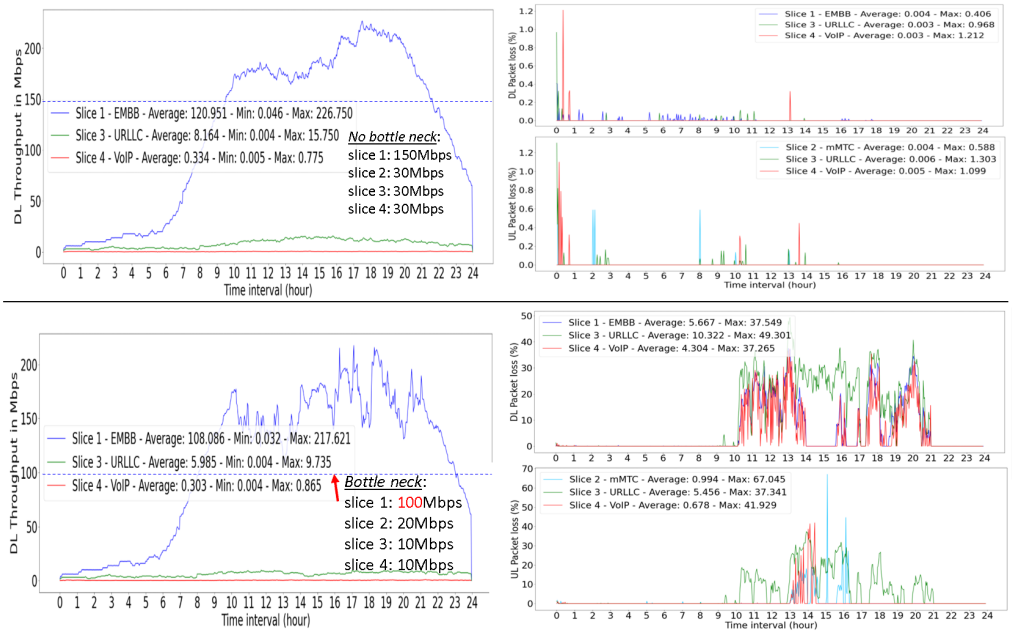


Figure 15: Slices' received throughput (DL - left) and packet loss (UL/DL - right) with link capacities in bottle neck stage (bottom) v.s. not in bottle neck (top).

On the other hand, configs "***Link Settings 2***" initializes links with max (physical) limits, which are much higher than the max known throughput demand of each slice type.

For the compute resources, "***VNF Settings 1***" initializes the VNF instances with relatively low limits (but not as low as the minimum allowable scaling limits) and each slice is allocated with only one VNF instance while "***VNF Settings 2***" sets the max allowable

limits (equal to physical ones), and *purposely* allocates VNFs to the slices not according to there application's demand.

Hence, **scenario 1** represents the worst-case scenario for both initialized network and compute resources (using the same settings as the bottleneck example table used to illustrate the performance metrics). On the other hand, **scenario 4** reflects the maximum initial resource configurations, but the allocation is not yet optimized. Meanwhile, scenarios 2 and 3 feature opposite initial resource limit extremes across the two domains.

One more thing to be noted is that links' minimum and physical data rate limits are different according to the slice type's traffic demand, while we decided to go with only one type of virtual machine (VM) which has the same levels of min and max (physical) resource limits across all the VNF instances; then we would find the proper range of compute resources capacities of that single VM type so that both vertical scaling and horizontal scaling can happen, which was a challenging part achieved through trial and error. It is challenging because with sufficiently high enough resource capacities in a single VNF instance, each of the slices only need one VNF instance and all the closed loop needs to do is to scale up/scale down (vertical scaling) without the need of scale out or scale in (horizontal); thus, the VNF's max compute resources must not be too low nor too high.

Furthermore, we did not attempt to evaluate and compare vertical scaling versus horizontal scaling in-depth (despite existing academic works such as [62] and industry recommendations such as [21]). This decision was due to the need to enable multiple VM types with differing physical resource capacities, which would require designing a significant number of additional scenarios. The scope of the experiment would then become too extensive to execute and manage, particularly when considering that a full simulation can take up to 20 hours with our OMNet++ packet level simulator.

Impacts from closed loop actions

Impacts on Slices' KPI violation As mentioned in the goal for each scenario, we anticipate that the closed loop algorithm will assist in reducing the total violations count, intensity ratio, and violation duration indicators (as listed in the "Slices KPI violation(s) summary" from section 3.3.5) or maintain them at the same levels, depending on the scenario's settings. This will help maintain or increase the final violation scores when compared to simulations without the closed loop actions. In particular, we assume that a closed loop is deemed effective when the overall KPI violation improvement ratio (the ratio between the two simulation's KPI scores) is considerably higher than the reduction ratio between the two simulation's resource scores (more details are provided in the results summary section). Therefore, the greater the disparity between these two ratios, the more efficient the closed loop algorithm is deemed to be.

Table 12 provides a summary of the slice-level KPI violations in scenario 1 with closed loop actions enabled, utilizing the original set of slice type special treatments listed in Table 4. This is the same scenario depicted in the example presented in Table 6. The corresponding simulation-level indicators are shown in Table 13. Aggregating the columns from the slice level to the simulation level, excluding the score column, is a straightforward process. To calculate the overall simulation KPI score, we consider the four slices with slice priorities of $50/19/30/20$ and slice KPI scores of $9.9/7.96/9.86/8.86$. The complements of the slice priorities to 100 are $50/81/70/80$. Therefore, the overall simulation KPI score can be computed as: $(9.9\times50+7.96\times81+9.86\times70+8.86\times80)/(50+81+70+80) = 9.03$.

The corresponding simulation-level indicators are shown in Table 13. Aggregating the columns from the slice level to the simulation level, excluding the score column, is a straightforward process. To calculate the overall simulation KPI score, we consider the four slices with slice priorities of $50/19/30/20$ and slice KPI scores of $9.9/7.96/9.86/8.86$.

The complements of the slice priorities to 100 are $50/81/70/80$. Therefore, the overall simulation KPI score can be computed as: $(9.9 \times 50 + 7.96 \times 81 + 9.86 \times 70 + 8.86 \times 80)/(50 + 81 + 70 + 80) = 9.03$.

Table 12: KPI violation(s) summary on each slice
(Scenario 1 - with actions) - Original closed loop settings

Slice name	Slice Priority	Total violations count	Invalid count	Max intensity ratio	Average violation duration (%)	Max violation duration (%)	Weighted violation score
Slice 1: eMBB	50	3	3	1.02	0.42	0.42	**9.90**
Slice 2: mMTC	19	11	10	1.04	2.08	4.03	**7.96**
Slice 3: URLLC	30	3	2	1.80	0.14	0.14	**9.86**
Slice 4: VoIP	20	6	6	2.08	0.21	0.28	**8.86**

Table 13: KPI violation result summary on simulation level
(Scenario 1 - with actions) - Original closed loop settings

Scenario (Scen.)	Total violations count	Invalid violations count	Max intensity ratio	Average violation duration (%)	Max violarion duration (%)	KPI violation score
Scen. 1 - With actions	23	21	2.08	0.71	4.03	**9.03**

Impact on resources consumption The closed loop algorithm proactively triggers scale down or scale in actions to reduce the limit of abundant resources. When a resource utilization level reaches the threshold check and has a tendency to keep increasing, the closed loop will increase the relevant limit to prevent violations due to potential lack of resources.

Figure 16 illustrates how compute resources are scaled for a single VNF instance during the simulation. The green and blue lines represent the average and maximum resource levels over time, respectively. The orange dashed lines indicate the configured limits, which change due to scaling actions, and the red lines show the physical limits of the VNF instance. First, it can be seen that the scaling limit is performed independently and differently among the three types of compute resources, depending on their utilization status, and the configured limit lines were able to adapt to the resource level with some level of

buffer (the minimum guaranteed buffer is determined by the combination scale up threshold ratio $\gamma_{\text{Vnf\_up}}$ and default scale up step $\alpha_{\text{Vnf\_up}}$ of the corresponding slice type), with some unavoidable delays due to the characteristics of a time driven closed loop. Secondly, the scaling up steps correlate to the uptrend rate of the resources (some other non correlated big scale up steps are due to re-active scaling and slice's KPI uptrend). Third, the scaling down steps are also different. The size of the steps when the resource usage at low levels are bigger compared to when the resource usage is at higher levels versus the configured limits (thanks to the two level thresholds and the three types of default scaling steps applied). Finally, all different levels of the orange lines and their duration are used to calculate the weighted average configured limits ($r_{jk}^{\bar{cap}}$) mentioned in equation 13 of the "Performance metrics" section 3.3.5. In addition, the areas above the orange lines represent the non utilized resources that were saved by the closed loop.

Figure 17 shows an example of the impact of link data rate scaling up and down on a slice (eMBB slice type) running under scenario 2 and the rest of its E2E KPIs. The initialized link data rate limit for this slice type is 100Mbps, which is below the minimum scaling limit, so we don't see the data rate limit (the blue dotted line in throughput plot at top left corner) was scaled down further. From around the 6 to the 7 hour, the traffic on this slice started to rapidly increase, and thanks to the proactive link scale up actions, the link data rate limit was increased early enough to prevent the bottle neck situation described previously. All of the three slice's KPI violations (both happened on max delay at around the 9, 10 and the 13 hour) were closed as invalid by our algorithm, so all link scaling actions observed are proactive ones. Similar to the VNF resources, the link data rate limit has different scale up and scale down steps; Moreover, as the slice type is eMBB, the default scale up step is bigger than the other slice types and gives higher steps especially when the throughput is sharply uptrend to create earlier and higher buffer on the link bandwidth, while the scale down steps is smaller and happen later (due to the lower scale down check

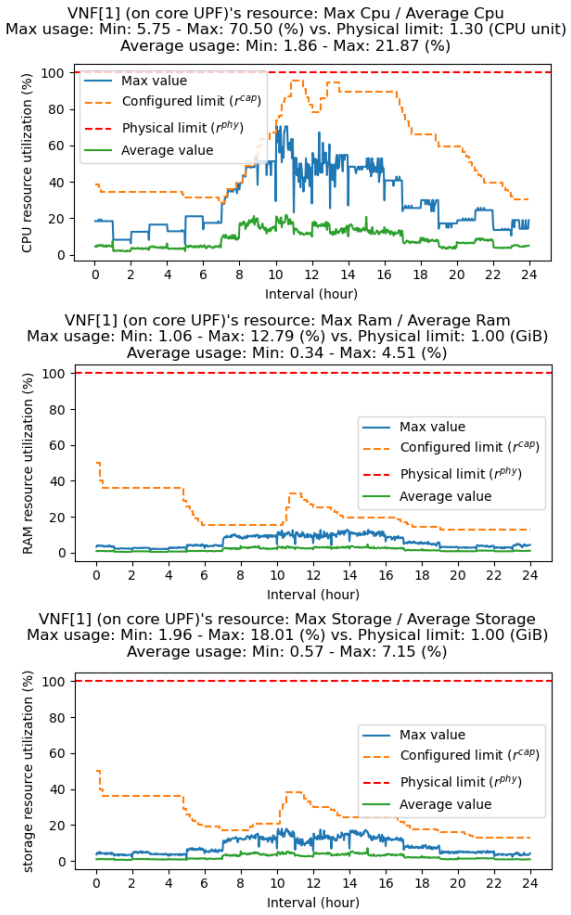


Figure 16: Compute resources utilization v.s. configured and physical limits of a VNF instance (dynamic configured limits are results of VNF scaling up/down actions)

thresholds) to make sure the scale down action is safe. Furthermore, the ratio between the link data rate weighted average configured limit (calculated from the varying line using equation 13) and the slice's max physical link data rate limit (512 Mbps in this case, as stated in Table 10) gives the slice's overall link utilization.

Remaining performance factors from our original closed loop algorithm settings in the single scenario 2 is provided in Figure 18. Plot $a)$ shows the impact of VNF horizontal scaling (scaling in / out), which increases or reduces the VNF instances in each slice, on the left. On the right, the plot shows the total actions performed on each slice across the whole simulation on the right (which sums up to the **"Total actions count"** simulation

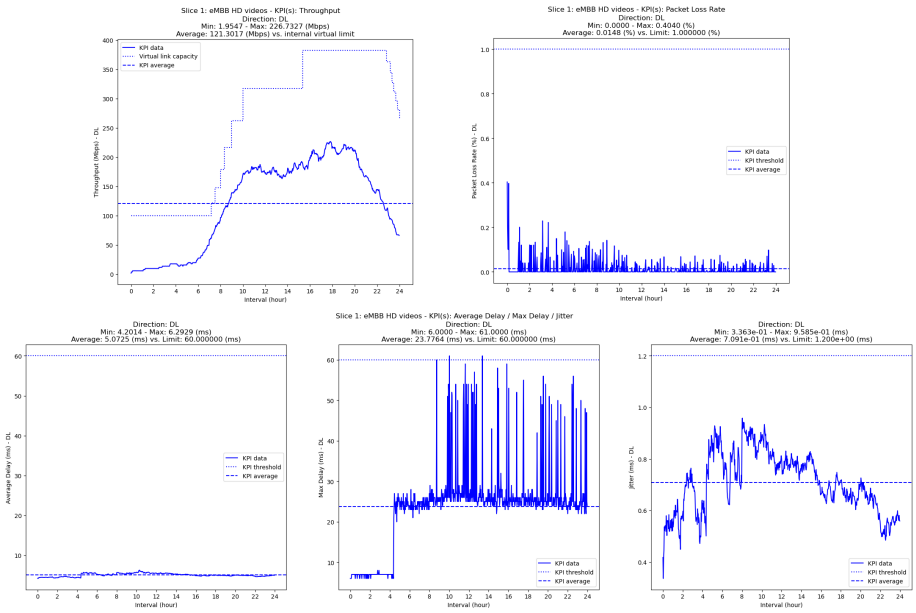


Figure 17: Impact of transport domain actions (link scaling up/down) on eMBB Slice KPIs (DL direction). Top left: Throughput with dynamic data rate limits; Top right: Packet loss; Bottom: Average delay - Max delay - Jitter

indicator), where the numbers next to action names in x-axis labels are the total action counts of that action across all slices; it is obvious to see that the number of VNF scale up and down actions is dominant compared to other action types, while horizontal scaling actions (in/out) contribute the least to the total number. Plot b) displays the overall compute resources utilization at aggregated level and and broken down into per-resource utilization, together with their scores; On that plot, the overall average percentage number (40.715%) is the first type indicator "**Overall VNF resources utilization (%)**" and the score next to it (6.862) is the score type indicator "**Overall VNF resources score**". Plot c) breaks down into further overall resource utilization for each resource type of all individual VNF instances; it can be noticed that there are totally 14 VNF instances in the network, but the VNF instances from 08 to 13 belong to the edge UPF and are always in-activated (their resource limits are at low standby levels). Finally, plot d) summarizes the links capacity utilization at both simulation level (Overall) and at each slice (link) level; similarly, the overall percentage (14.134%) is the first type indicator "**Overall links data rate utilization**

(%)" and the score (8.94) is the score-type indicator "**Overall link resources score**".

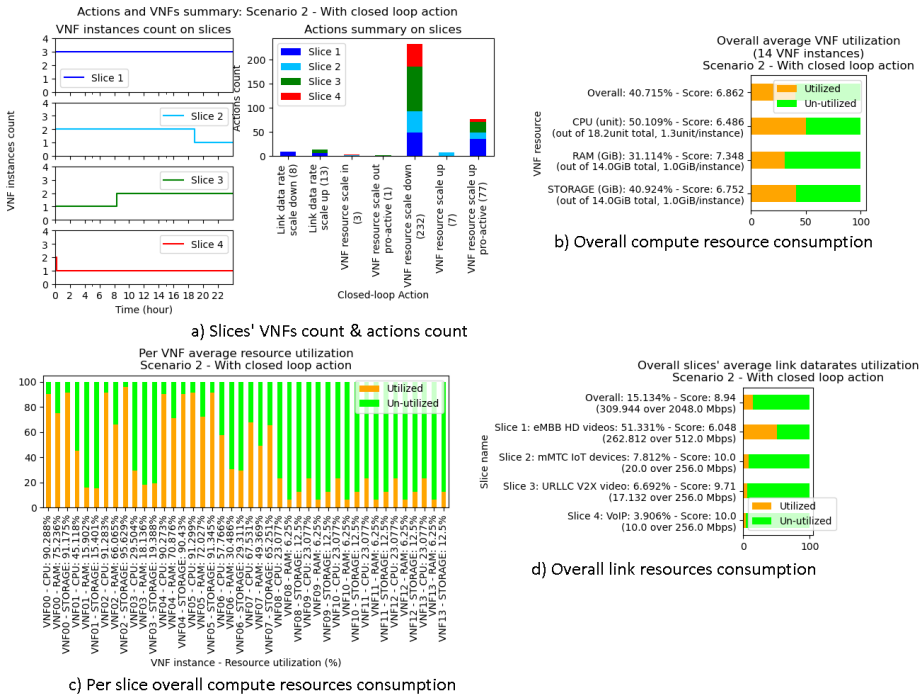


Figure 18: Different aspects of closed loop performance

Benefits of implementing the RFC check and execution functionality

The response flow checklist (RFC) we implemented to illustrate the algorithm 6 (AutoRFC) is for a known packet loss issue on the mMTC slice for scenarios 1 and 2. As shown in Tables 10 and 11, with scenarios 3 and 4, the links' configured limits started at high levels, as described in the "Link Settings 2" configs, and they are never scaled down below the minimum allowable thresholds across all slices. Those boundaries kept the total available bandwidth in the whole network big enough, so that the packet loss violations on the mMTC slice 2 never occurred, as shown in Figure 19a. However, in scenarios 1 and 2, the initial link capacity limit of "Link Settings 1" for slice 2 is below its minimum allowable scale down limit (20 vs. 40 Mbps). Since the actual throughput of this slice is always staying much lower than this initial limit, the slice's configured link capacity would not be

increased or decreased during the whole simulation with the existing generic link scaling action rules. As the bandwidth of this slice is still in the region belong to the network's bottle neck stage, though the impact is not as severed as when there is no closed loop action, it still caused frequent short packet loss violations on this slice. The violations and KPI metrics on this scenario is shown in Figure 19b.

To address the gap between those two set of scenarios, the special RFC was made as following: First, an RFC record is added into the RFC table as shown in Table 14.

Table 14: RFC record for mMTC packet loss issue

Id	KPI	Status	SliceType	RfcName
0	packetLoss	SKIPPED	mMTC	mMTC_pkLoss_linkScaleUp

So when the RFC table is checked at line 19 of the high level algorithm 1, if there is an SKIPPED packet loss violation on the mMTC slice found on $vTrack$ table, the RFC named "*mMTC_pkLoss_linkScaleUp*" will be checked and executed: If in the last 2 hours there is at least one INVALID packet loss violation on the mMTC slice, and if the slice's current configured link capacity is below its minimum allowable scale down limit, then special action is that the link capacity will be increased to the minimum allowable limit. It is necessary to be noted that the 2 hours period is beyond the normal Tw of the closed loop so it need to access the archived historical records for the check and execution of the "*mMTC_pkLoss_linkScaleUp*" RFC. As a result, packet loss violation did not completely gone for slice 2, but the occurrence frequency greatly reduced as shown in Figure 19c. In addition to the visual plots, the violation metrics and scored are attached at the bottom of each plot to show the improvement of the KPI performance in slice 2 in the same scenario when the RFC is implemented.

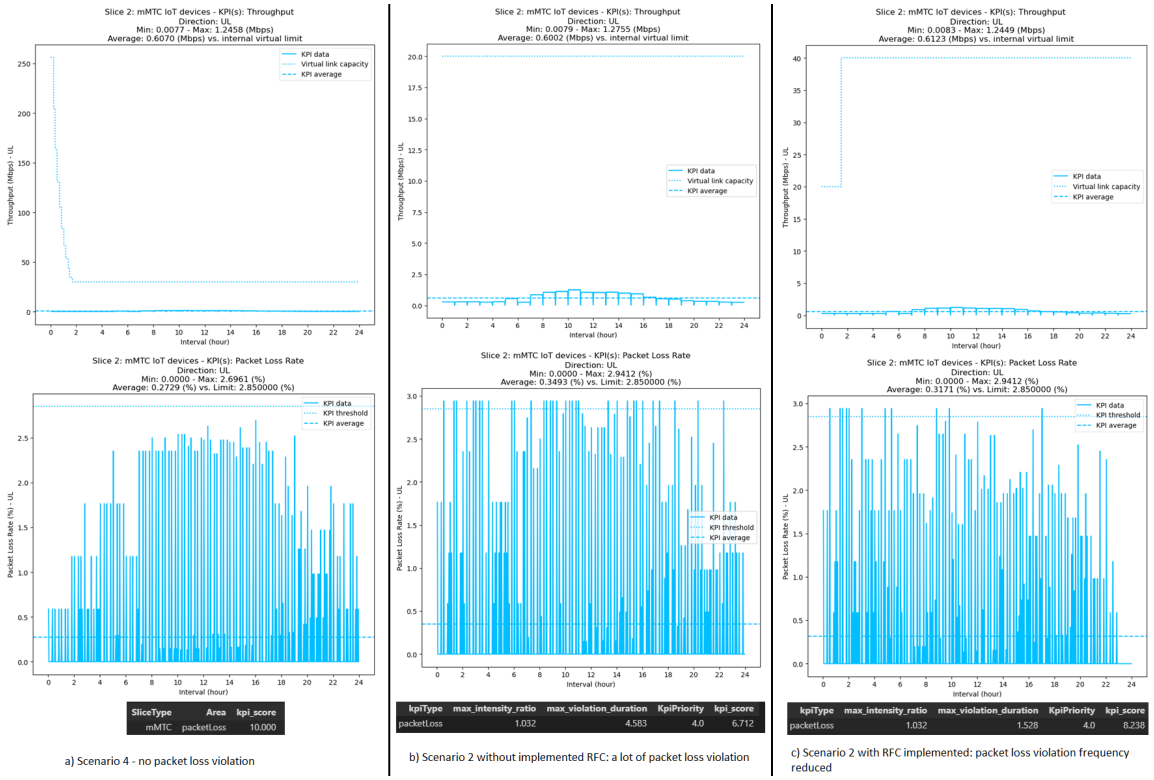


Figure 19: Effects of implemented RFC on packet loss violation of mMTC slice, scenario 2 (2nd and 3rd plots). Top: mMTC slice's throughput and configured limits; Bottom: slice packet loss KPI and corresponding packet loss KPI scores

Simulation results summary

Performance metrics on all scenarios Table 15 summarizes all performance metrics of all 4 scenarios with and without closed loop actions. KPI violation comparison of scenario 1 has been discussed before in the closed loop's impacts section 3.4.2 with total improvement score (between no actions and with actions simulation) increased about 5.64 times (from 1.6 to 9.03), while the VNF resource score just slightly decreased from 8.96 to 7.9 and link resource score decreased from 10 to 8.96, making the overall resource score reduced from 9.48 to 8.43 (1.12 times). Thus, on scenario 1 the closed loop only used 1.12 times more resources but helped to reduce overall KPI violations by 5.64 times, which is a good result, and meets scenario 1's goal from Table 11. For scenario 2, the closed loop not only solved the bottle neck problem but also reduced the KPI violation by nearly 10 times

(0.9 to 8.98) while only consumed 1.1 times more resources (7.14 v.s. 7.9) by improving on VNF resources utilization and use slightly more link resources (met goal of scenario 2). On scenario 3, the closed loop improved the resource score by almost twice (4.48 to 8.24, by using just 4% more VNF resources and improved the link resource score from 0 to 7.89) while still improving the KPI violation by 5% (9.85 to 9.9), so scenario 3's goal is also met. In the remaining scenario 4, the goal was not fully met yet when the KPI violations slightly degraded 1.3% (from 9.85 to 9.72) despite having better type one indicators[3], while the closed loop helped save up to 3.67 times the resource consumption (from 2.14 to 7.86) with savings from both compute and link resources.

To summarize, the initial goals for scenarios 1-3 out of 4 are satisfied with the first official parameter settings (shown in Table 4). Hence, further improvements can be made, which will be discussed in the upcoming section on "Closed loop flexibility".

Table 15: Simulations result summary (full simulation time) - Original special treatment settings

Scenario (Scen.)	Total violations count	Invalid violations count	Max intensity ratio	Average violation duration (%)	Max violation duration (%)	KPI violation score	Overall VNF resources utilization (%)	Overall VNF resources score	Overall links data rate utilization (%)	Overall link resources score	Overall resources score	Total actions count
Scen. 1 - No action	55	48	77.00	25.88	49.58	**1.60**	23.15	8.96	6.84	10.00	**9.48**	0
Scen. 1 - With actions	23	21	2.08	0.71	4.03	**9.03**	31.86	7.90	14.97	8.96	**8.43**	328
Scen. 2 - No action	54	49	88.12	23.62	46.94	**0.90**	63.12	4.29	6.84	10.00	**7.14**	0
Scen. 2 - With actions	24	20	2.08	0.79	4.58	**8.98**	40.72	6.86	15.13	8.94	**7.90**	338
Scen. 3 - No action	11	11	3.81	0.37	0.56	**9.85**	23.15	8.96	62.50	0.00	**4.48**	0
Scen. 3 - With actions	8	7	2.67	0.15	0.28	**9.90**	26.04	8.59	20.79	7.89	**8.24**	251
Scen. 4 - No action	11	11	3.81	0.37	0.56	**9.85**	63.12	4.29	62.50	0.00	**2.14**	0
Scen. 4 - With actions	9	7	2.35	0.19	0.28	**9.72**	32.26	7.86	21.00	7.87	**7.86**	346

Closed loop scores variation As observed from all scenarios, the "**Time to converge**" for link data rate resources is approximately 2 hours and for the VNF resources is around

[3]The KPI score and other KPI indicators look counterposed in this scenario 4: the action-enabled simulation had lower score but better violation count, max intensity ratio, average and max violation duration compared to the no-action simulation. The reason is that slice 4 in the action-enabled run had worse score due to some accumulated jitter violation that was not the worst in terms of duration and peak, but pulled the overall KPI score down due to the high priority of VoIP slice type. This is an example when the first type indicators does not reflect the situation as well as the scoring indicators.

2.5 hours after the simulation starts. To assess the impact of closed-loop actions and to compare the performance variation across all scenarios, we have analyzed the data by measuring the performance for the entire simulation period from 0 to 24 hours, as well as from the time the network reaches the resource convergence status, which is after 2.5 hours until the end of the simulation period.

Table 16 summarizes those score variations. As mentioned previously (in Performance metrics section 3.3.5), the variation is calculated as the mean $\pm$ standard deviation of the score across the 4 scenarios. It can be observed that regardless of the measured duration, without the closed loop the network performance is very bad with a very low mean and a high standard deviation for all scoring areas. With the closed loop algorithm enabled, the KPI violation performance is good (always above 9) while the resource consumption was never higher than 30% of the minimal resource levels (worst resource score is $7.76 - 0.6 = 7.16$ out of 10). In the post 2.5 hours measurement, the variation result shows that after the initial phase of modulating the resource limits to the proper levels, on average most of the resource types achieved the same level of utilization across all initial scenarios with the maximum variation of only 6% (VNF score's std-dev 0.6 out of 10) and the overall resource score only varied up to 2.1%. After that 2.5 hours of initial phase, compared to the whole simulation duration measurement, all the resource levels became more stable (resource scores having smaller standard deviations) and KPI performance was also slightly improved 0.6% (9.47 v.s. 9.41) with less variation. The result also shows that regardless of the initialization state, after some converging time the closed loop would deliver almost the same result in terms of KPI violations and resource consumption, and by tweaking the slice specific parameters from Table 4 we can actively make some trade offs between violation performance, the level of resources consumed and also the average amount of action counts.

Table 16: Summary on simulation scores' variation across scenarios (with & without closed loop actions) - Original parameters set

Summary - Measured duration	KPI score variation	Resources score variation	VNF score variation	Link score variation
No actions - 0h - 24h	5.55 ± 4.31	5.81 ± 2.76	6.62 ± 2.34	5.00 ± 5.00
With actions - 0h - 24h	9.41 ± 0.41	8.11 ± 0.24	7.80 ± 0.61	8.41 ± 0.54
With actions - 2.5h - 24h	9.47 ± 0.37	8.10 ± 0.21	7.76 ± 0.60	8.44 ± 0.39

Closed loop flexibility: To test the closed loop's flexibility, at first, we modified some parameter values in Table 4, as outlined in the guidelines in Section 3.3.4). Specifically, we adjusted the vertical scale-up process by increasing both the maximum and average thresholds, which slows down the action trigger. We also set the high scale-down steps to be much larger and increased the scale down thresholds check, making it more likely for the scale-down action to occur and take bigger steps when resource capacities are high. These changes were made with the goal of achieving better resource savings, as indicated by higher resource scores. We then ran the same four scenarios again, using the closed loop with these new settings.

Figure 20 illustrates the differences in scaling behavior for both link and compute resources of two set of settings in scenario 4[4]. Setting 1 is the original parameter values shown in Table 4, while Setting 2 is the set of modified values in favor of more resource saving. The results indicate that in Setting 2, as shown in the bottom plots, both link and resources vertical scaling actions were quicker to reduce the resource limits to the converged stage. Particularly, the link capacity reduced from 512 Mbps to below 150 Mbps in just about one hour, and fewer scaling actions were needed compared to Setting 1, which took two hours in the top plots). Moreover, the VNF scale-down limits reached sub 40% in just about two hours, compared to more than 4 hours in Setting 1. In addition, the scale up

[4]Reminder about scenario 4: both configured link capacity and compute resource limits start at maximum levels

steps' magnitudes in Settings 2 were also reduced, as observed from the shape and maximum levels of the limit lines. This helped to achieve better overall weighted average limits, but it required more scaling actions to be taken.

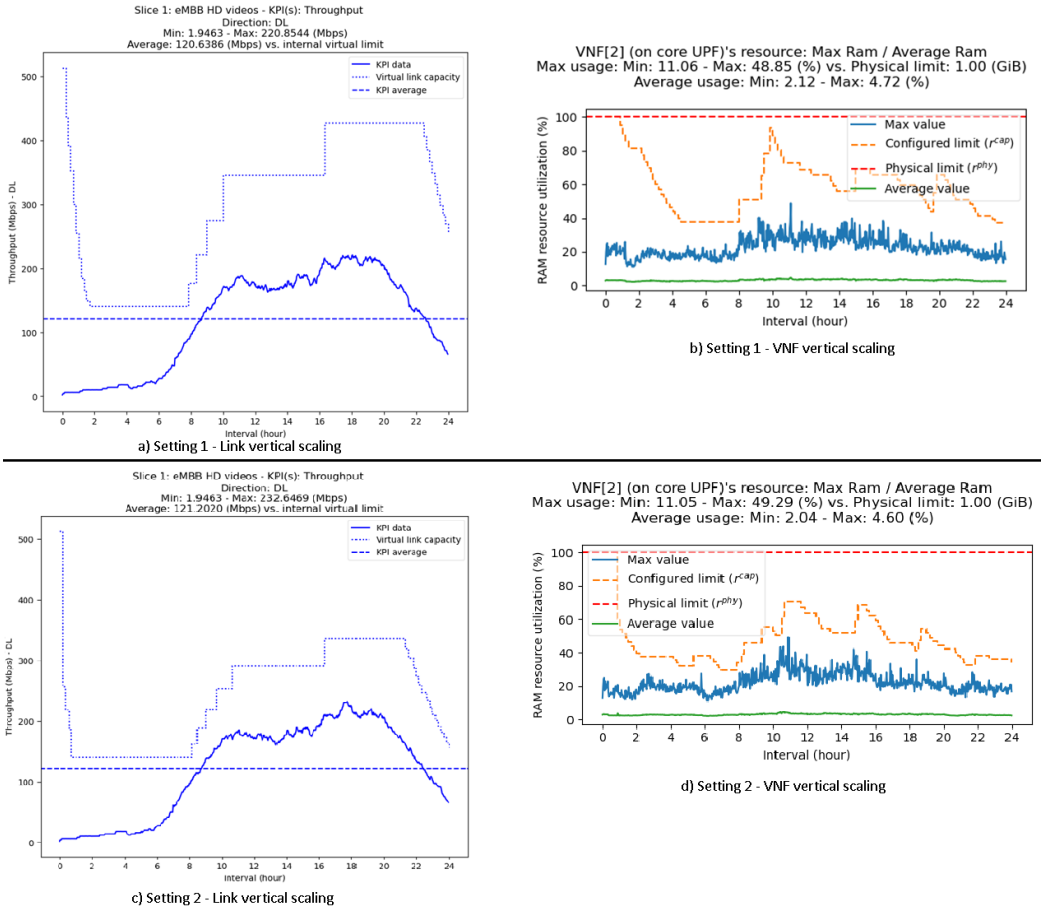


Figure 20: Comparison between 2 special treatment sets of parameters (Setting 1 on top and Setting 2 at bottom) in term of resources scaling and converge time.

Of course, the resource gains did not come without a cost, as they had an impact on KPI violation scores. We found that the closed loop with Setting 2 performed much better in scenario 3 and 4, while KPI violation performance became worse in scenario 1 and 2.

To gain further insight into the performance range of the closed loop system, we developed and executed two additional parameter sets, referred to as Settings 3 and 4. Setting 3 was designed in the opposite direction with Setting 2 to provide extra precaution levels when scaling up and down, with bigger and earlier scaling up steps and smaller medium

and low scale down steps with longer delay thresholds; This was intended to improve KPI violation performance while still benefiting from the big high scale down steps of Setting 2, thereby achieving shorter convergence time and better resource saving. Setting 4 represents the final fine-tuning attempts, based on the lessons learned from the first three sets of settings, and includes the enabling of an RFC to improve packet loss on the mMTC slice for scenarios 1 and 2 (as mentioned in section 3.4.2).

To provide a more visual representation, Figure 21 compares the performance variations (post convergence) among the four settings. The plot utilizes a similar method to the no-action summary shown in Figure 14, but includes bars that represent the average actions per hour, taking into account that each set of settings has different convergence times.

The KPI and resource scores variations of Setting 1 (represented by the red dot and lines in Figure 21) are consistent with the first three columns of the last row in Table 16.

When compared to Setting 1, the closed loop's performance variations with Setting 2 (represented by blue bar in Figure 21) showed an improvement in resource score by approximately 4.26%, increasing from 8.1 to 8.46. The variation in resource score also improved (with a standard deviation of 0.09 compared to 0.2), indicating that the system utilized the resource capacity better than Setting 1 across all scenarios. However, the average KPI violation performance of Setting 2 was worse than Setting 1, degraded by 2.43% from 9.47 to 9.24. Additionally, the KPI variations were much larger (0.78 vs. 0.37) due to the different KPI performances in each scenario or the impact of initial limit settings. It should be noted that the overall average hourly action rate in Setting 2 was higher than Setting 1, at 13.42 $\pm$ 2.18 compared to 12.03 $\pm$ 1.69. This may contribute to the overall execution cost of the system.

Analyzing the performance variations of Setting 3 (represented by the green bar in Figure 21), we observe that this setting achieved a more balanced state between all three aspects. Specifically, it achieved a good KPI score mean and standard deviation (with only

a 1.4% degradation of mean from 9.47 to 9.33 compared to Setting 1, while keeping the same and even slightly smaller standard deviation of 0.35 vs. 0.37), gained better resource utilization compared to Setting 1, and had the lowest hourly action rate among the four settings.

Finally, Setting 4 (the back bar), supported by the implemented RFC "mMTC_pkLoss_linkScaleUp", combined the most effective elements from the previous three settings, resulting in the highest overall KPI score with the lowest variation across scenarios. This setting also maintained a respectable resources score, though slightly lower than Setting 2. However, despite having an average hourly action rate nearly as high as Setting 2, Setting 4 exhibited the largest variation in action rate among all four settings.

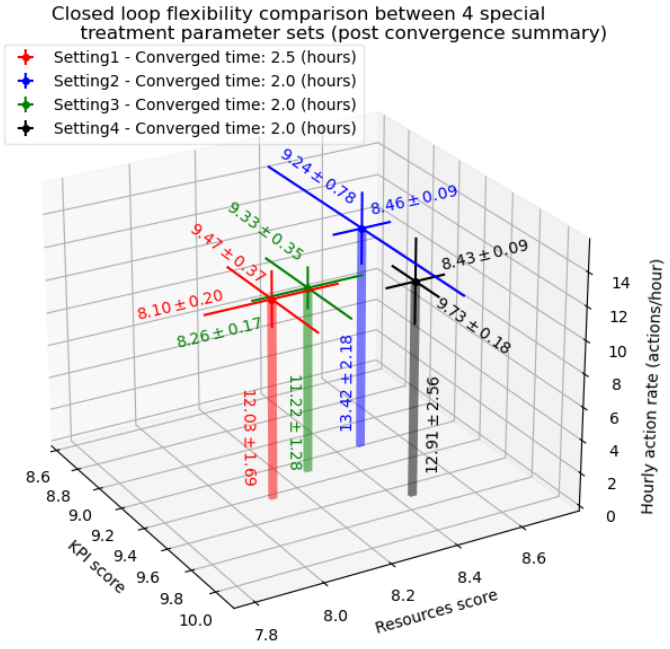


Figure 21: Comparison between 4 special treatment sets of parameters in term of closed loop performance variations across 4 scenarios

To provide further insights, we conducted a deeper analysis of the slice-level metrics. We calculated similar variation metrics to evaluate the impact of each version of the closed-loop settings on each slice type.

Retrieving the score variation summary for KPI violation and link resources at the slice level is straightforward, as they are calculated at that level prior to aggregation to the simulation level. However, for the VNF resource score, we must map each VNF resource limit change event to the relevant slice and VNF mapping record at that same moment to determine the accumulated VNF resource limits multiplied by the time consumption of each slice. This allows us to calculate the slice-level VNF resource ratios using the following formula:

$$\text{ratio}_{ik,k\in\{C,R,S\}} = \frac{\sum_{j=1}^{N_{\text{Vnf\_total}}} \sum_{n=1}^{n_{\Delta k_i}} \left(r_{jkn}^{cap}.\Delta_{t_{ni}}\right)}{T.\sum_{l=1}^{N_{\text{Vnf\_total}}} r_{lk}^{phy}}$$

where ratio_{ijk} is the overall (accumulated) resource ratio of VNF resource k of slice i, $n_{\Delta k_i}$ is the number of limit changes of VNF j in the duration $\Delta_{t_{ni}}$ in which that VNF belongs to slice i, and T is the whole measured simulation time.

We can then derive the slice-level VNF resource score based on the minimum and maximum ratios of each slice's compute resource across all simulations using min-max scaling.

The results of the slice-level analysis are presented in Figure 22. For the eMBB slice (top left corner), all four settings performed well in terms of KPI scores, with Setting 2 achieving a perfect score of 10 across all scenarios. However, the eMBB slice type requires the largest amount of bandwidth, resulting in poor resource scores, which were improved in the three latter settings compared to the original Setting 1.

For the URLLC slice (bottom left corner), which has stringent KPI requirements, Settings 3 and 4 achieved the best (minimal) KPI violation variation across the four settings, while Setting 2 had the lowest KPI score mean and the highest standard deviation. Setting 4 achieved good enough KPI score and the best resource score results, making it the most optimized setting. This URLLC slice also required the largest number of hourly action rate to maintain good buffer on VNF resource limits compared to other slice types due to the

small scale up and medium scale down steps.

The mMTC slice (top right corner) has high energy efficiency requirements and relatively good overall resource consumption scores, but not the best KPI scores.

Finally, the VoIP slice, with low bandwidth consumption, achieved both very good KPI and resource scores without consumed so many actions. This slices also had consistently small resource score standard deviations across all four settings. The KPI score variations were also less prone to differences in initial scenario limit settings compared to the mMTC slice.

To summarize, the closed loop parameters can be tuned by network operators to provide customized slice treatments based on the Service Level Agreement (SLA) contract with different styles of customers. This allows network operators to prioritize KPI violations, resource utilization, or action rates based on the customer's needs or to find a balance between all three aspects. In other words, the network operators can choose the closed loop settings that align with the SLA contract and customer requirements.

It is also worth noting that the KPI violation scores in scenario 4 for the three sets of settings 2, 3, and 4 were better compared to the no closed loop run, which helped exceed the scenario 4's goals and fulfill all four scenarios' goals. Tables 17 and 18 list all values of the four sets of settings in one place, with each table covering a domain area, and each line of numbers representing a parameter for one set of settings. Special treatment values for different slice types are highlighted using the same colors as in the previous variation summary plots in Fig.21. Additionally, Table 19 summarizes all scenario performance results of the four sets of settings for the full simulation time, where the scores are also highlighted with the same settings' colors.

Ideally, having more data points would provide a more comprehensive understanding of the closed loop's performance. However, collecting additional sets of settings would be too time- and labor-intensive. Each data point requires modifying parameters' values

in Table 4 and running at least eight simulations to cover all four scenarios for each set of settings. Nonetheless, the four sets of settings evaluated in this study provide valuable insights into the closed loop's effectiveness and highlight the trade-offs between KPI violations, resource utilization, and action rates.

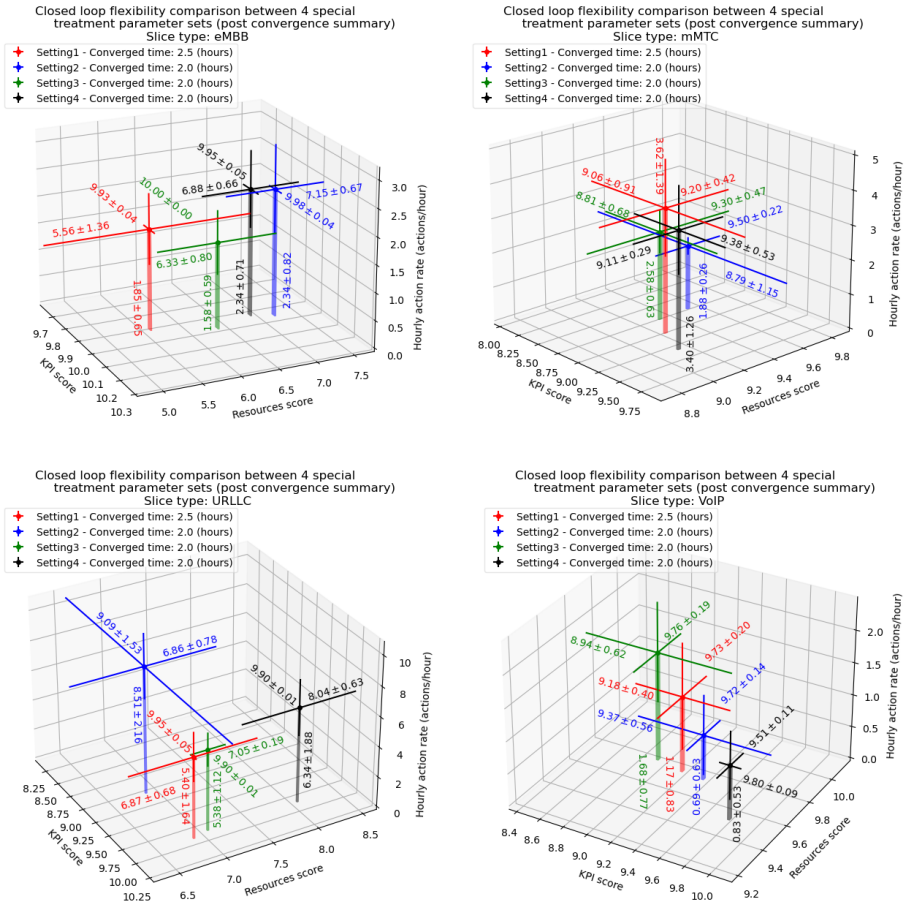


Figure 22: Performance variation summary broken down by each slice type (Noted: each plot have different scales, so for plot-to-plot comparison we should look at the numbers, not the lines' lengths).

All simulations observation: As we were unable to generate more sets of closed loop parameter settings, we made the most with what we had by analyzing all the simulations conducted across all scenarios for the four existing sets of settings instead of just the best

simulation of each scenario as in previous analyses. We also used an interpolation technique to estimate the range of KPI performance and the behavior of each scenario's simulations based on their resource score and hourly action rate, rather than relying on summarized statistics such as mean and standard deviation. Although this approach is not as comprehensive as generating more data points, it provides a more thorough understanding of the closed loop's performance within the existing data limitations.

Figure 23 presents four interpolated surface plots for the four scenarios and our four settings, with a total of 65 simulations combined (see the small table at the bottom right corner of the figure for detailed simulation and scenario counts). The x and y axes denote the hourly action rate and resource score, respectively, while the z axis represents the KPI violation score. The surface's color indicates the KPI score levels, and settings' colors were kept the same with previous figures.

To estimate Z (KPI score) from the inputs X (action rate) and Y (resource score), we experimented with various interpolation methods and function families. After assessing the fitting quality in terms of root mean square error (RMSE) and R-squared coefficient of correlation, as well as visual impression, we decided to use the quadratic function as our final method:

$$Z_{interp} = a_0.X^2 + a_1.Y^2 + a_2.X.Y + a_3.X + a_4.Y + a_5$$

where a_0 to a_5 are coefficients determined by the fitting process.

After analyzing the results in four scenarios, it can be concluded that there is no strong correlation between action rate and resource score vs KPI performance. Each scenario, with its initial configured resource limits, has different impacts on closed-loop performance, and the impact level may depend on the optimization of each setting.

Scenario 1's surface plot falls into the quarter of high resource score and middle-to-high action rate, and when resources scores are high and action rates fall in the average zone, the

KPI performance of simulations appears to be better. However, only a few simulations are in that high KPI score range. Most simulations on scenario 2 fall into middle-to-high action rate and a more spread resource score region, which also did not have a good KPI score compared to scenarios 3 and 4. However, its interpolated surface was the flattest among all scenarios, indicating consistency in KPI performance across all simulations from all settings.

The interpolated surface of scenario 3 is also quite flat and falls in the region of low action rate and high resource score, with KPI performance consistently good among all settings. The surface in scenario 4 falls into the middle region of both action rate and resource score, while having the second-best KPI scores among scenarios.

Regarding settings' performance, simulations from Setting 1 had worse resource scores than the rest in all scenarios and consumed the least action rate (except for scenario 4), while only achieving comparable KPI performance with the remaining settings in scenario 3 and worse KPI performance in all other scenarios. This was understandable since it was the first official setting. For setting 2, with the first attempts to improve resource scores compared to setting 1, it achieved the resource saving goal for most scenarios except scenario 2, and the KPI scores were at the same levels with KPI scores from setting 1's simulations. Settings 3's simulations performed well in KPI score for all scenarios except scenario 3, while having the lowest action rates and average resource score. The final settings 4 still proved its effectiveness with good KPI score and resource score across scenarios, with the only drawback of relatively high action rates.

In conclusion, this is still a multi-objective problem and should be investigated further regarding the relationship between KPI violation, resources utilization, and action rate. As such, network operators must find a compromise between these three aspects to achieve optimal performance.

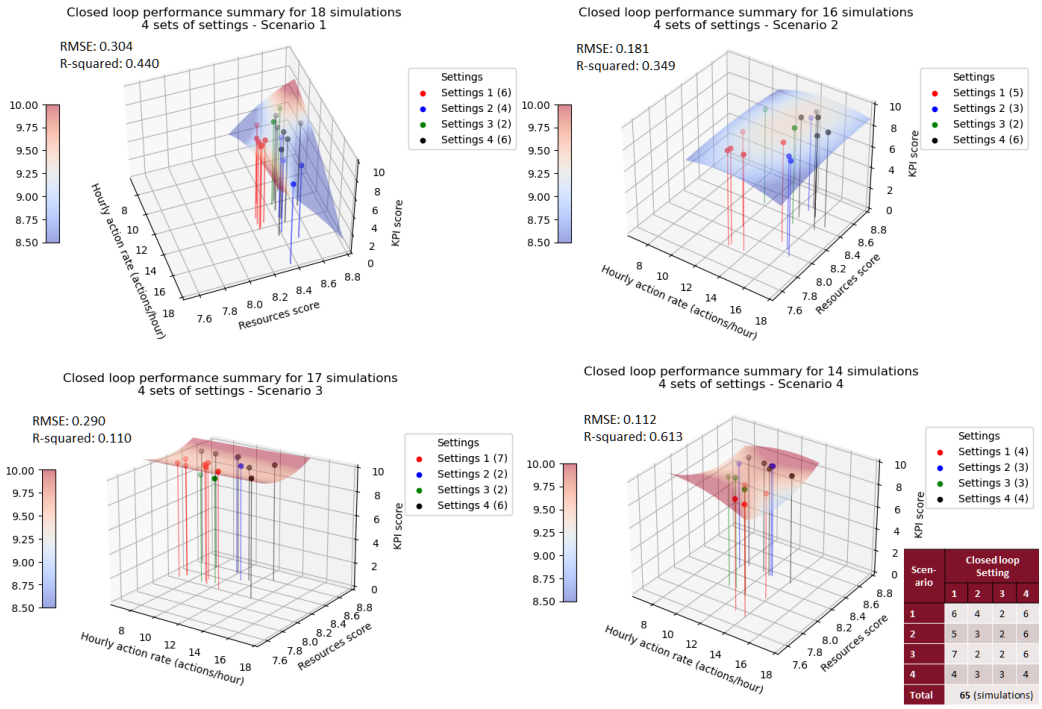


Figure 23: Interpolation of KPI score from all 65 executed simulations across all 4 closed loop settings, grouped by scenario. The numbers in parentheses next to each Setting's legend are the number of simulations run using that setting in the relevant scenario.

3.5 Conclusion

Based on the analysis of simulation results from section 3.4, we can conclude that our quality assurance closed loop algorithm met all of the performance metrics and the stated goals, despite the fact that it uses a pure heuristic approach without any application of machine learning or deep learning (which proved that despite its inevitability in the long run, when ML/DL is not yet available, heuristic methods can still be a good alternative for a quick and easy production kick off, as similar to the example in [10]). Even more than that, with the flexibility in adjusting parameters settings based on different slice and resource types, the closed loop give operators ability to control and negotiate the quality vs. cost trade off in the SLAs with various customers. To empower operators with full control over this flexibility, we created a comprehensive guide that allows them to modify the slice and resource type specific parameters. Additionally, customized RFCs can be developed

for historical issues, further adding values to the closed loop system for network operators. Furthermore, another important way to utilize our closed loop algorithm is to help auto generate "labeled" training data (which is too rare to find for 5G network quality assurance domain these days) for future closed loop algorithms leveraging machine learning, deep learning techniques. Finally, the performance metrics we have designed are the first of their kind in our knowledge. These metrics serve as a valuable reference for benchmarking future work and can be used in real-time monitoring during production. They offer the capability of summarizing the overall closed-loop performance at a high level, as well as breaking it down into specific slices, resource types, and action levels.

For future plan, there are many paths that can be followed to improve the closed loop, such as: One, integrate some short-term traffic data prediction that predicts both input KPIs and resource values, which gives much better estimation of future network states than the current simple trend and slope tests; such real pro-active approach would enhance both KPI violation performance and resources utilization. Two, enable more sub-domain KPIs so that the closed loop has better controls over the network with more faults segmentation and troubleshooting capabilities. Three, we can add more E2E KPI types in the coverage to extend the scope of the closed loop, which the most viable option is adding throughput as an official KPI to be guaranteed [13].

Table 17: Special treatments for each slice type, action type and resource type (Configurable) - All four settings (one per line): Settings1 - Settings2 - Settings3 - **Settings4**
Area: Transport domain

Action type / Resource	**Parameter**	**eMBB - slice1**	**URLLC - slice3**	**mMTC - slice2**	**VoIP - slice4**
Link data rate scaling (Throughput - TP)	MAX_TP_SCALE_UP_TRIGGER_RATIO ($\gamma_{l\_up\_max}$)	0.7 0.75 0.65 **0.7**	0.8 0.8 0.65 **0.7**	0.8 0.8 0.75 0.8	0.8 0.8 0.75 0.8
	AVG_TP_SCALE_UP_TRIGGER_RATIO ($\gamma_{l\_up\_avg}$)	0.4 0.55 0.4 **0.4**	0.5 0.6 0.5 0.5	0.5 0.6 0.5 0.5	0.5 0.6 0.5 0.5
	DEFAULT_LINK_SCALE_UP_STEP (α_{l_up})	0.2 0.15 0.2 **0.2**	0.1 0.1 0.2 **0.2**	0.1 0.1 0.1 0.1	0.1 0.1 0.1 0.1
	MAX_TP_SCALE_DOWN_CHECK_RATIOS High/Low ($\gamma_{l\_down\_max\_h}$ / $\gamma_{l\_down\_max\_l}$)	0.5 / 0.2 0.55 / 0.3 0.5 / 0.2 **0.55** / 0.3	0.6 / 0.3 0.6 / 0.3 0.5 / 0.2 0.6 / 0.3	0.6 / 0.3 0.6 / 0.3 0.6 / 0.3 0.6 / 0.3	0.6 / 0.3 0.6 / 0.3 0.6 / 0.3 0.6 / 0.3
	AVG_TP_SCALE_DOWN_CHECK_RATIOS High/Low ($\gamma_{l\_down\_avg\_h}$ / $\gamma_{l\_down\_avg\_l}$)	0.3 / 0.1 0.5 / 0.2 0.3 / 0.1 0.5 / 0.2	0.4 / 0.2 0.5 / 0.2 0.4 / 0.2 0.5 / 0.2	0.4 / 0.2 0.5 / 0.2 0.4 / 0.2 0.5 / 0.2	0.4 / 0.2 0.5 / 0.2 0.4 / 0.2 0.5 / 0.2
	DEFAULT_SCALE_DOWN_STEPS Low/Med/High ($\alpha_{l\_down\_l}$ / $\alpha_{l\_down\_m}$ / $\alpha_{l\_down\_h}$)	.05/.1/.15 .05/.15 /.5 .05/.1/.5 **.05/0.15/.5**	.05/.1/.2 .1/.2 /.6 .05/.1/.5 .1/.2 /.6	.05/.1/.2 .1/.2 /.6 .05/.1/.6 .1/.2 /.6	.05/.1/.2 .1/.2 /.5 .05/.1/.6 .1/.2 /.6

Table 18: Special treatments for each slice type, action type and resource type (Configurable) - All four settings (one per line): Settings1 - Settings2 - Settings3 - **Settings4**
Area: Core domain

Action type / Resource	**Parameter**	**eMBB - slice1**	**URLLC - slice3**	**mMTC - slice2**	**VoIP - slice4**
VNF resources scaling (CPU/RAM/ Storage)	DEFAULT_SCALE_UP_STEP (α_{Vnf_up})	0.1 0.1 0.15 0.1	0.15 0.15 0.15 **0.15**	0.1 0.1 0.15 0.1	0.1 0.1 0.15 0.1
	DEFAULT_SCALE_UP_STEP_PROACTIVE ($\alpha_{Vnf\_up\_pro}$)	0.1 0.1 0.15 0.1	0.2 0.15 0.2 **0.15**	0.1 0.1 0.15 **0.15**	0.1 0.1 0.15 0.1
	SCALE_UP_CHECK_MAX_RATIO (γ_{Vnf_up})	0.8 0.85 0.75 0.8	0.7 0.75 0.65 **0.7**	0.8 0.85 0.75 **0.75**	0.8 0.85 0.75 0.8
	SCALE_DOWN_CHECK_MAX_RATIOS High/Low ($\gamma_{Vnf\_down\_max\_h}$ / $\gamma_{Vnf\_down\_max\_l}$)	0.6 / 0.3 0.7 / 0.3 0.55 / 0.3 0.65 / 0.3	0.5 / 0.2 0.6 / 0.3 0.5 / 0.2 **0.6** / 0.3	0.6 / 0.3 0.7 / 0.3 0.55 / 0.3 0.65 / 0.3	0.6 / 0.3 0.7 / 0.3 0.55 / 0.3 0.65 / 0.3
	SCALE_DOWN_AVG_RATIOS High/Low ($\gamma_{Vnf\_down\_avg\_h}$ / $\gamma_{Vnf\_down\_avg\_l}$)	0.4 / 0.1 0.5 / 0.2 0.4 / 0.1 0.5 / 0.2	0.3 / 0.1 0.5 / 0.2 0.3 / 0.1 0.5 / 0.2	0.4 / 0.1 0.5 / 0.2 0.4 / 0.1 0.5 / 0.2	0.4 / 0.1 0.5 / 0.2 0.4 / 0.1 0.5 / 0.2
	SLICE_KPI_SCALE_DOWN_CHECK_MAX_RATIOS Low/High ($\gamma_{Kpi\_down\_max\_l}$ / $\gamma_{Kpi\_down\_max\_h}$)	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.85	0.7 / 0.8 0.7 / 0.8 0.65 / 0.75 0.7 / **0.75**	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.85	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.85
	SLICE_KPI_SCALE_IN_CHECK_MAX_RATIOS Low/High ($\gamma_{Kpi\_in\_max\_l}$ / $\gamma_{Kpi\_in\_max\_h}$)	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.8	0.7 / 0.8 0.7 / 0.8 0.65 / 0.75 0.7 / 0.8	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.8	0.7 / 0.85 0.7 / 0.85 0.7 / 0.8 0.7 / 0.8
	DEFAULT_SCALE_DOWN_STEPS Low/Med/High ($\alpha_{Vnf\_down\_l}$ / $\alpha_{Vnf\_down\_m}$ / $\alpha_{Vnf\_down\_h}$)	.05/.1/.15 .05/.15/.4 .05/.1/.3 .05/.15/.4	.05/.1/.15 .05/.1/.4 .05/.1/.3 .05/**.1**/.4	.05/.1/0.2 .05/.15/.5 .05/.1/.4 .05/.15/**.5**	.05/.1/.15 .05/.15/.4 .05/.1/.3 .05/.15/.4
	SCALE_IN_TOTAL_MAX_THRESHOLD (relative) ($\gamma_{Vnf\_in\_total\_max}$)	0.85	0.8 0.8 0.8 0.85	0.85	0.85

Table 19: Simulation results summary (full simulation time)
All 4 sets of special treatment settings

Setting	Scenario (Scen.)	Total violations count	Invalid violations count	Max intensity ratio	Average violation duration (%)	Max violation duration (%)	KPI violation score	Overall VNF resources utilization (%)	Overall VNF resource score	Overall links datarate utilization (%)	Overall link resources score	Overall resources score	Total actions count
N/A	**Scenario 1 - No action**	55	48	77.00	25.88	49.58	**1.61**	23.15	8.96	6.84	10.00	**9.48**	0
1	**Scenario 1 - With actions**	23	21	2.08	0.71	4.03	**9.03**	31.86	7.90	14.97	8.96	**8.43**	328
2	**Scenario 1 - With actions**	30	25	2.08	0.88	4.72	**8.12**	31.17	7.98	13.03	9.21	**8.60**	378
3	**Scenario 1 - With actions**	21	19	2.08	0.48	2.78	**9.17**	29.85	8.14	15.25	8.91	**8.53**	298
4	**Scenario 1 - With action**	21	15	2.50	0.49	2.08	**9.47**	28.83	8.26	15.93	8.68	**8.47**	365
N/A	**Scenario 2 - No action**	54	49	88.12	23.62	46.94	**0.91**	63.12	4.29	6.84	10.00	**7.15**	0
1	**Scenario 2 - With actions**	24	20	2.08	0.79	4.58	**8.98**	40.72	6.87	15.13	8.94	**7.90**	338
2	**Scenario 2 - With actions**	27	24	2.08	0.92	4.31	**8.96**	32.18	7.87	13.05	9.20	**8.54**	361
3	**Scenario 2 - With actions**	29	26	2.08	0.63	3.47	**9.08**	31.97	7.89	15.04	8.94	**8.42**	293
4	**Scenario 2 - With action**	20	18	12.44	0.35	0.97	**9.53**	28.35	8.32	15.96	8.67	**8.49**	379
N/A	**Scenario 3 - No action**	11	11	3.81	0.37	0.56	**9.85**	23.15	8.96	62.50	0.01	**4.49**	0
1	**Scenario 3 - With actions**	8	7	2.67	0.15	0.28	**9.90**	26.04	8.59	20.79	7.89	**8.24**	251
2	**Scenario 3 - With actions**	9	8	1.68	0.21	0.42	**9.92**	25.11	8.70	17.62	8.33	**8.51**	262
3	**Scenario 3 - With actions**	14	12	12.40	0.69	4.31	**9.07**	27.27	8.44	20.18	8.01	**8.23**	234
4	**Scenario 3 - With action**	9	7	1.70	0.19	0.42	**9.91**	23.34	8.90	18.52	8.18	**8.54**	257
N/A	**Scenario 4 - No action**	11	11	3.81	0.37	0.56	**9.85**	63.12	4.29	62.50	0.01	**2.15**	0
1	**Scenario 4 - With actions**	9	7	2.35	0.19	0.28	**9.72**	32.26	7.86	21.00	7.87	**7.87**	346
2	**Scenario 4 - With actions**	4	3	1.31	0.24	0.69	**9.95**	29.56	8.18	17.52	8.35	**8.26**	335
3	**Scenario 4 - With actions**	8	7	1.77	0.21	0.42	**9.92**	31.70	7.93	20.33	8.00	**7.96**	310
4	**Scenario 4 - With action**	15	9	1.85	0.43	0.97	**9.89**	27.84	8.38	18.54	8.18	**8.28**	286

Chapter 4

Further discussions

4.1 Simulation network

Figure 24 depicts the network topology that was implemented in the OMNeT++ simulator to evaluate our closed loop algorithm. The architecture comprises a core domain consisting of a UPF connected to 4 application servers via a single router, an RAN network featuring 8 gNodeB stations, each connected to a gNodeB router, and user equipment (UE) devices that link to the RAN network. The core and RAN networks are interconnected by a set of 4 LSR routers, which represent the transport network. Additionally, an edge domain is included, which comprises an edge UPF (upfMec) and 4 edge servers that are similar to the core network. The only difference is that the edge network connects directly to the RAN without a transport network in between.

4.2 Deep dive into Closed loop architecture

Figure 25 summarizes the framework of our closed loop architecture in a more detailed level with the main functional blocks and sub-blocks as well as interaction flows between blocks. All of them were designed from scratch, and the design ideas were mostly inspired

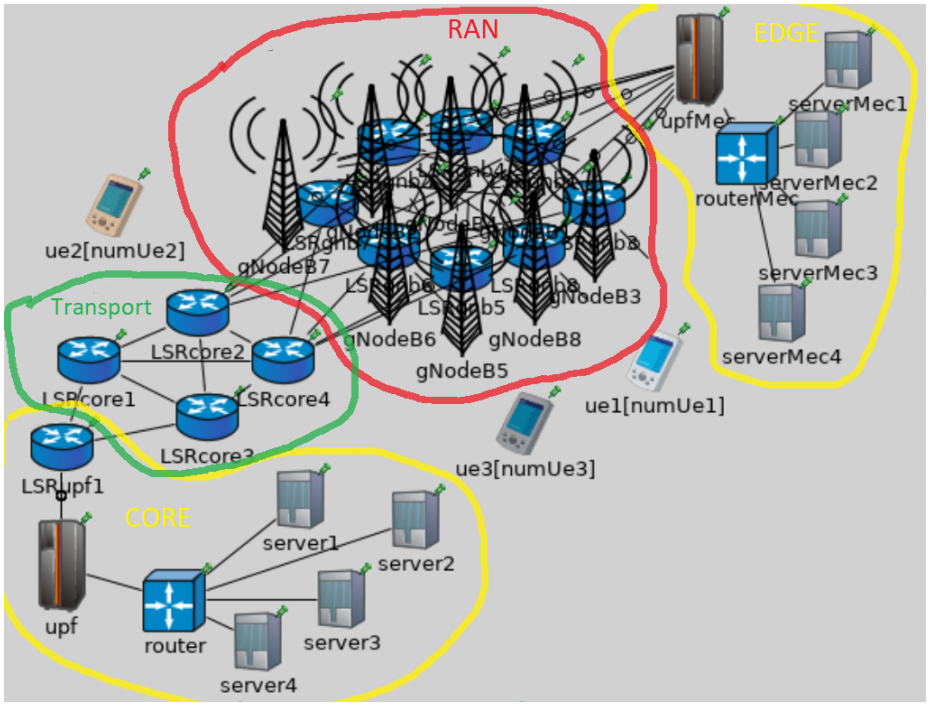


Figure 24: The network built for simulation

by the popular "MAPE-K" (Monitor, Analyze, Plan, Execute and Knowledge) closed loop concept in orchestration [30] that was mentioned in Chapter 2.

The **"Input"** block retrieve web socket data (slices KPI and compute resources data) from OMNeT++'s web socket (ws) client. The WS-to-Sqlite adapter accumulated and split data into chunks of individual Tw and fetch each chunk into a predefined Sqlite template (following original sqlite template format from OMNeT++) and save into individual Sqlite data files, each data chunk is processed in a separate thread to prevent consequent processing causing extra delay to the core closed loop algorithm. When each chunk data is written completely, it notifies the core closed loop that the latest Tw data is available. The core closed loop when notified can start retrieve slices' KPIs and VNF resources data, each slice or a group of VNFs are queried in an independent thread. Moreover, the input block also has the capability to retrieve initialized settings from OMNeT++ to feed to the Configuration block so that some of the initialized settings from the closed loop be in sync with OMNeT++, preventing the double efforts to setting initialized configs for both the closed

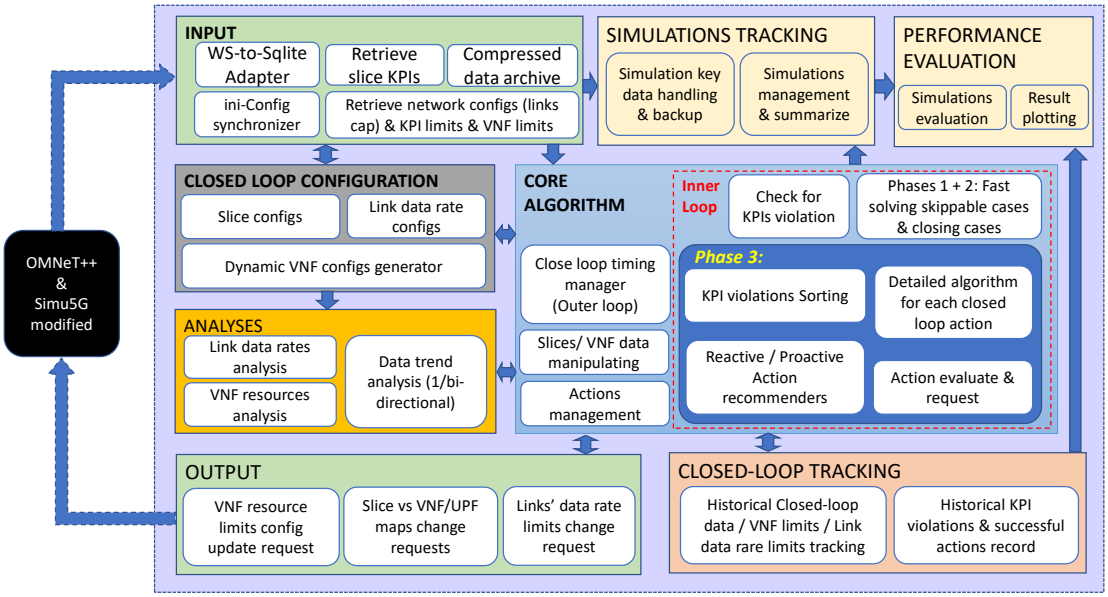


Figure 25: Detailed view of our Quality Assurance Closed Loop Architecture

loop and the simulator. Queried data are processed and store in dictionary format which is ready to used by the closed loop, and each process data is written into a pickle file (with much smaller size than the original sqlite db file) for archiving and reporting purposes.

The **"Closed loop configuration"** block helps configure the slices, VNF instances and initial resource limits based on retrieved .ini settings from OMNeT++, as well as keep track of the UPF location to Slice to VNF mappings during the simulation. The configs format is designed in a way to easily add new slice instances into the network in the future and the VNF instances are dynamically created based on the number of VNFs per UPF location retrieved from OMNeT++'s .ini configs. Moreover, all of the slice and resoure type special treatments parameters are stored in those slice, link and VNF configuration.

The **"Analyze"** block receives input data from the "Slices / VNF data manipulating" sub-block from closed loop and provides back the core data analysis capabilities for the closed loop ("Core Algorithm" block), with all of the threshold, ratios and trends calculation on slices, link and compute resources data.

The **"Core Algorithm"** block has an outer loop to communicate with the "Input" block

to synchronize the latest simulation time and data retrieval status with OMNeT++. The three sub-blocks "Check for KPIs violation", "Phase1 + 2" and "Phase 3" in the red dashed "Inner Loop" box have been described in the core algorithm design section 3.3 in Chapter 3. There is a data manipulating sub-block to retrieve latest or historical slice and VNF data files and perform any merge or filter needed to send the manipulated data to the "Analyize" block. More importantly, the "Action management" controls all of the action requests sent to the "Output" block, check if the action was performed successfully to update status for the "Closed-loop tracking block"

To completely "close" the loop, the **"Output"** block helps translate action request from the core closed loop into the proper files and formats that can be read by OMNeT++. Those files include: mapping of UPF location of each slice (core or edge UPF), mapping between slices and different VNF instances at each UPF location, compute configured limits (CPU, RAM, storage) for all VNF instances and configured link data rate limits for all links. After modifying those relevant files, the block will validate if the modification is correct, and if yes it will send back the necessary changes in VNFs and link configs so that the closed loop can update the relevant configurations to be in sync with those configured files used by OMNeT++. Finally, it will let the core closed loop know that the action request is executed completely.

The **"Closed loop tracking"** block keep track of all historical slice and VNF data files for each Tw in two historical tracking files, so that the core closed loop can easily get the required data file from any interval from the compressed data archive in the "Input" block. In addition, all the compute and link resource limit changes as the result from successful closed loop actions, and the actions themselves are also recorded into files. Final KPI violations' statuses and parameters are also recorded. All of those historical tracking files are be used by the closed loop's core algorithm or for closed loop performance evaluation purpose.

The two blocks **"Simulation tracking"** and **"Performance evaluation"** help automate the reporting of closed loop performance in each and all necessary simulations. Simulations are setup and running from different remote computers in our lab, at the end of each completed simulation all of the minimal simulation data required for evaluation metrics calculation and plotting will be sent to a centralized folder in the cloud. There is a master file to keep record of all of the completed simulation Ids, scenario and closed loop versions, as well as which simulation is selected in the final report. With that, the evaluation of all of the performance summaries and plots in the Results section 3.4 in Chapter 3 can be automatically performed with just a few clicks, or else the amount of manual works to do each individual analyses for all 16 simulations on table 19 of Chapter 3 would be tremendous.

4.3 Limitation on dynamic link data rate from simulation environment

In our simulation environment, we have modified the E2E virtual links' implementation beyond the traditional MPLS level 2 protocol (Multi-Protocol Label Switching) for routers. These links' data rate is dynamically updated at runtime with each tunnel following the configured data rate limit change requests from the closed loop. As no bandwidth control feature existed during our feature modification phase, this design choice was necessary. However, this has resulted in a limitation on the dynamic link data rate implementation in our simulation environment. Since all slices share the same physical links across the transport network, the behavior of dynamic data rate links is only capable of controlling the packet sending rate for each individual incoming packet along the line. This means that when one link is configured with a relatively low data rate limit that reaches a low threshold, the entire network becomes bottlenecked, impacting all slices and directions. This can lead to unusually high packet loss and a significant increase in other network KPI

values.

Figure 15 shown previously in Chapter 3 illustrates the network in both stages without any closed loop: In both scenario, the slices a provided sufficient VNF resource and only have difference in initial links data rates configured. For the settings above, the 4 slices' max configured link data rates are sufficiently high for all the slices' traffic: 150Mbps - 30Mbps - 30Mbps - 30Mbps, the first thing to notice is slice 1's limit of 150Mbps is just an "artificial" link data rate control as mentioned in the implementation limitation above, so its max Downlink (DL) throughput plot (top left corner, data retrieved and plotted from the receiver's side) can still exceed that limit and reached 226.75Mbps. Moreover, in this no-bottle-necked setting, the packet loss KPIs on the right in both directions (top right corner) are at normal state (max packet loss is just around 1-1.3%). On the plots below, slices' max configured link data rate limits are 100Mbps - 20Mbps - 10Mbps - 10Mbps and what we observed is although slice 1's throughput can still pass the 100Mbps limit but could not reach the max level (226.75Mbps) like the above settings and only increased up to 217.6Mbps with intermittent drops starting from 10th hour (bottom left corner); At the same time of around 10th hour, the packet loss values (bottom right corner) also started to rocket in all slices and directions, which showed that the network is in bottle neck stage (since the overall network's links data rate are constrained at 100Mbps, all the un-sent packets will be dropped and caused packet loss). This is the bottle neck situation that formed the worst case scenarios (scenarios 1 and 2) in our scenarios design part from Chapter 3.

And the workaround solution in our closed loop algorithm was to keep a minimum allowable link capacity much higher than necessary to prevent the network falls again into the bottle neck stage, thus reduced the potential link resources saving since the links cannot be scaled down further.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Automated solutions for Quality assurance of 5G Network Slices are never in need as much as nowadays with the evolve of network and telecommunications technology, yet there is an existing work that is found touching the level of details and complexity as our closed loop algorithm. Despite all kinds of challenges mentioned in Chapter 2, especially with the limitation in dynamic link data rate implementations from the network environment, our designed heuristic closed loop presented in Chapter 3 still prove it effectiveness in solving slice KPI violations due to network or compute resources' problems, thus guarantee the QA requirements for all diverse slice types in a of manner efficient resource utilization. Our heuristic closed loop is classified as "passive" where the major mechanisms are threshold based comparison but it still have some pro-active components with the action giving decisions partially depend on the latest short or medium trends in KPIs and resources data. Moreover, with the additional capability of integrating customized sets of rules and actions for specific historical issues through addition of RFCs, the network operator has more tools to optimize the network performance. However, the most selling point of the algorithm is

the ability and flexibility to tweak slice, and resource type specific parameters to adapt differently with different levels of resources state of the network. This ability provides huge benefits for both network providers and customers in negotiating about trade offs between KPIs quality and resource capacity utilization.

We also presented a comprehensive guide on how to adjust those configurable parameters to meet the requirements of each slice and resource type. The guide explains what modifications are necessary to alter the behavior of the closed loop in favor of a customer's specific tolerance level for KPI violations, while balancing the trade-off for resource savings. We based on the guide ourselves to create different parameters sets that change the closed loop tendency as shown in the final part in result section (3.4.2) of Chapter 3.

Our novel performance metrics, while not perfect, enable us to summarize the performance of the closed loop and network from three different perspectives: KPI violation, resource consumption, and closed loop actions, as well as the closed loop's flexibility, which are presented through visual plots. These metrics can serve as a valuable reference for benchmarking future work on quality assurance for closed loop algorithms. Furthermore, they can be implemented as a real-time closed loop performance monitor for network providers, allowing for detailed analyses at the slice, KPI, and resource domain levels.

Last but not least, our closed-loop algorithm provides an easy-to-implement solution that requires fewer considerations than machine learning/deep learning models when deploying them into production. Unlike ML/DL models that require constant monitoring to maintain model reliability and prevent potential issues due to distribution shift in production data, the heuristic nature of our closed loop offers some initial benefits. Moreover, the actions generated by the closed loop can be used as labeled data to train supervised learning models or as initial data for training reinforcement learning agents.

5.2 Future work

There are a lot of promising opportunities to improve our closed loop algorithm in many aspects.

As discussed in the conclusion section (3.5) of Chapter 3, integrating short-term traffic prediction techniques into our current closed loop algorithm could yield significant benefits. By combining our current threshold-based methods and trending analysis with predicted values of both KPIs and resource usage in the next Tw, we can make more precise scaling decisions, complete with necessary capacity buffers for each slice and resource type. This proactive approach enables the closed loop to anticipate the possibility of KPI violations in the near future, thereby improving the guarantee of KPI quality while still maintaining good control over resource utilization. Implementing such a real-time predictive approach will require further research into the selection of appropriate prediction models, as well as the development of methods for incorporating prediction data into the closed loop algorithm. Nonetheless, this represents an exciting avenue for future work in this area.

At present, our closed loop algorithm can only identify potential KPI violations due to transport or compute domain resource issues based on E2E KPI data. However, with access to more sub-domain KPIs, we could improve the algorithm's ability to identify and segment faults for root cause analysis. For example, E2E delay KPIs are composed of three sub-domain KPIs: delays at the core, transport, and RAN. If the closed loop detects a violation of E2E delay KPIs for a particular slice, it could analyze the sub-domain KPIs to determine which specific domain is the primary contributor to the violation. Based on this analysis, the closed loop could initiate further actions on the specific domain or, if necessary, notify the network operator to manually investigate and troubleshoot the suspect area. However, accessing these sub-domain KPIs will require modifications to the simulation environment. Specifically, we will need to enable data collection and monitoring of sub-domain KPIs, as well as develop methods for integrating this data into the closed loop algorithm.

Extending the quality assurance goal for another KPI type - E2E throughput - would be a valuable improvement to our closed loop algorithm. However, guaranteeing minimum and maximum bandwidth for each slice is challenging due to the current limitation of our dynamic data rate links. To address this issue, we can leverage the HTBQueue (Hierarchical Token Bucket queue) recently developed for OMNeT++ [13] into our simulation environment. This new queue type offers several advantages over our current implementation, including better traffic separation between traffic flows (which can be mapped into our virtual links), which can help to solve the known bottleneck issues in the network, and gain greater link capacity savings from the closed loop actions. In addition, the HTBQueue offers features for managing guaranteed flow bit rate (GRBR) and maximum flow bit rate (MFBR), which are essential for controlling and sustaining the QoS goals for E2E throughput KPI, as specified in the 3GPP specification [4]. To integrate the HTBQueue into our simulation environment, we will need to replace all of the current dynamic data rate links in the network with these queues and develop the capability for them to receive configured limit changes in real time, similar to what we have done for the current environment.

While our closed loop framework already allows for managing and updating slice assignments between different UPF locations (in the core or at the edge), we were unable to develop detailed algorithms for slice movement actions due to time and workload constraints. Adding such algorithms to our closed loop would provide network providers with even more options for balancing KPI quality and resource consumption trade-offs. For instance, slices with critical delay requirements such as URLLC could be temporarily allocated to the edge UPF to reduce E2E delay, at the expense of increased costs associated with operating UPF or other VNF functions at user premises. However, developing algorithms for these types of slice assignment actions requires further research on energy and cost strategies (such as those outlined in [54]), as well as additional scenarios design and compute resources re-dimensioning for performance validation.

In order to improve the effectiveness of our closed loop algorithm, it will be important to conduct further research on resource and cost analyses, including both operational costs and action execution costs. By better understanding the correlation between quality assurance goals and costs, we can develop more meaningful metrics to evaluate the trade-off between KPI quality and cost. This information will enable network operators to negotiate more effectively with customers and use our closed loop's flexibility to achieve optimal outcomes. In addition, this knowledge will help us develop appropriate reward or cost functions, which are the key part for evolving our heuristic closed loop into a full reinforcement learning (RL) agent. To achieve this, we will need to study how to map the flexibility of our scaling actions (in terms of continuous scaling steps and handling multiple action types at a time for a slice) into the limited action space of RL algorithms.

To make our closed loop algorithm more practical for real-life systems, we can take several actions. First, we can automatically archive the growing list of slice and VNF data files created by the closed loop after a defined period to reduce storage space. Second, we can work on improving our algorithm to handle more VNF types with different slice mapping styles, such as shared type VNFs that can be used by multiple slices simultaneously, or even managing service function chains. However, this will increase the complexity of our closed loop and algorithms. Lastly, we can integrate our closed loop algorithm into existing orchestration platforms like ONAP and Kubernetes, which will allow it to work seamlessly with other network management functions and enhance the overall efficiency of the system.

Bibliography

[1] 3GPP. Study on Key Quality Indicators (KQIs) for service experience, 03 2016. TR 32.862, Version 14.0.0.

[2] 3GPP. Management and orchestration; 5G end to end Key Performance Indicators (KPI), 09 2018. TS 28.554, Version 17.8.0.

[3] 3GPP. Key Performance Indicators (KPI) for UMTS and GSM, 07 2020. TS 32.410, Version 16.0.0.

[4] 3GPP. System architecture for the 5G System (5GS), 12 2021. TS 23.501, Version 17.3.0.

[5] 5G Americas. New Services & Applications with 5G Ultra-Reliable Low Latency Communications. Technical report, 5G Americas, 11 2018. White paper: `https://www.5gamericas.org/white-paper-archive/`.

[6] 5GPPP. Dynamically Reconfigurable Optical-Wireless Backhaul/Fronthaul with Cognitive Control Plane for Small Cells and Cloud-RANs, 2015. D2.1 Requirements Specification and KPIs Document. `https://www.5g-xhaul-project.eu/download/5G-XHaul_D2_1.pdf`.

[7] 5GPPP. D2.1 5G and Vertical Services, use cases and requirements, 2018. 5G-PICTURE D2.1: `https://www.5g-picture-project.eu/download/5g-picture_d21.pdf`.

[8] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song. Network Slice Lifecycle Management for 5G Mobile Networks: An Intent-Based Networking Approach. *IEEE Access*, 9:80128–80146, 2021.

[9] A. M. Alwakeel, A. K. Alnaim, and E. B. Fernandez. A pattern for a virtual network function (VNF). In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, New York, NY, USA, Aug. 2019. ACM.

[10] A. Amini, K. Banitsas, and J. Cosmas. A comparison between heuristic and machine learning techniques in fall detection using Kinect v2. In *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, pages 1–6, 2016.

[11] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984, 2020.

[12] B. Barani. EC perspective on 5G core standardization. Available at `https://docbox.etsi.org/Workshop/2017/20170406_ETSI_SUMMIT_5G_NWK_INFRASTRUCTURE/01_SESSION_A_CHALLENGES_5G_INFRAST_DEVELOP/PERSPECTIVE_5G_CORE_INFRAST_EC_BARANI.pdf`, 2017.

[13] M. Bosk, M. Gajic, S. Schwarzmann, S. Lange, and T. Zinner. HTBQueue: A Hierarchical Token Bucket Implementation for the OMNeT++/INET Framework. *CoRR*, abs/2109.12879, 2021.

[14] R. Boutaba. Netsoft lecture1 (Introduction to SDN). Available at `https://youtu.be/DbUyTXWuaU0`, 2018.

[15] R. Boutaba, N. Shahriar, M. A. Salahuddin, S. R. Chowdhury, N. Saha, and A. James. AI-Driven Closed-Loop Automation in 5G and beyond Mobile Networks. FlexNets '21, page 1–6, New York, NY, USA, 2021. Association for Computing Machinery.

[16] S. Canale, M. Tognaccini, L. M. de Pedro, J. J. R. Alonso, K. Trichia, D. Meridou, A. Georgakopoulos, I. Maistros, G. Loukas, V. Audebert, T. Doukoglou, M. Kitra, A. Tzoulis, E. Tzifa, R. Legouable, and R. Gavazzi. D1.1 requirements definition & analysis from participant vertical industries, 2018. 5G EVE. https://doi.org/10.5281/zenodo.3530391.

[17] C. Casetti, C. F. Chiasserini, T. Deiß, P. A. Frangoudis, A. Ksentini, G. Landi, X. Li, N. Molner, and J. Mangues. Network slices for vertical industries. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 254–259, 2018.

[18] S. Dahmen-Lhuissier. 5G. ETSI. Accessed: 2023-01-26. https://www.etsi.org/technologies/5g.

[19] N. de Sousa and C. Rothenberg. CLARA: Closed loop-based zero-touch network management framework. In *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 110–115, 2021.

[20] O. Delgado, B. Jaumard, Z. Ding, F. Bishay, and V. Bissonnette. Demo: A Network Simulator for 5G Virtualized Networks. In *IEEE International Conference on Network Softwarization (Netsoft)*, 2022.

[21] M. Ekuan, M. Alberts, U. Dahan, et al. Autoscaling, 2022. Microsoft: https://learn.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling.

[22] F. Granelli. Network slicing. In *Computing in Communication Networks*, pages 63–76. Elsevier, 2020.

[23] GSMA. An Introduction to Network Slicing, 2017. White Paper GSM Alliance.

[24] GSMA. Generic Network Slice Template. Version 4.0, 2020. GSMA Official Document NG.116.

[25] M. Hussain and I. Mahmud. pyMannKendall: a python package for non parametric Mann Kendall family of trend tests. *J. Open Source Softw.*, 4(39):1556, July 2019.

[26] H. Jmila, M. I. Khedher, and M. A. E. Yacoubi. Estimating VNF resource requirements using machine learning techniques. In *Neural Information Processing*, pages 883–892. Springer International Publishing, 2017.

[27] E. Kapassa, M. Touloupou, A. Mavrogiorgou, and D. Kyriazis. 5G & SLAs: Automated proposition and management of agreements towards QoS enforcement. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–5, 2018.

[28] T. Karamplias, S. T. Spantideas, A. E. Giannopoulos, P. Gkonis, N. Kapsalis, and P. Trakadas. Towards Closed-loop Automation in 5G Open RAN: Coupling an Open-Source Simulator with xApps. *Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pages 232–237, 2022.

[29] A. Kaushik. Network slicing - demystified, 2020. Linkedin. https://www.linkedin.com/pulse/network-slicing-demystified-ashutosh-kaushik/.

[30] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer (Long Beach Calif.)*, 36(1):41–50, Jan. 2003.

[31] J. Kim and M. Xie. A study of slice-aware service assurance for network function virtualization. In *IEEE Conference on Network Softwarization (NetSoft)*, pages 489–497, 2019.

[32] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[33] A. Leiter, A. Hegyi, N. Galambosi, E. Lami, and P. Fazekas. Automatic failover of 5G container-based User Plane Function by ONAP closed-loop orchestration. *NOMS IEEE/IFIP Network Operations and Management Symposium*, pages 1–2, 2022.

[34] Q. Liu, N. Choi, and T. Han. OnSlicing: Online End-to-End Network Slicing with Reinforcement Learning. CoNEXT '21, page 141–153, New York, NY, USA, 2021. Association for Computing Machinery.

[35] V. S. Mai, R. J. La, T. Zhang, and A. Battou. End-to-End Quality-of-Service Assurance with Autonomous Systems: 5G/6G Case Study. *IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 644–651, 2022.

[36] D. W. Meals, S. A. D. Jean Spooner, and J. B. Harcum. Statistical analysis for monotonic trends, Tech Notes 6, November 2011. Technical Report 6, Environmental Protection Agency by Tetra Tech, Inc, Fairfax, VA, 11 2011. National Nonpoint Source Monitoring Program (NNPSMP). `https://www.epa.gov/nps/nonpoint-source-monitoring-technotes`.

[37] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network Function Virtualization: State-of-the-Art and Research Challenges. *IEEE Communications Surveys & Tutorials*, 18, 09 2015.

[38] P. Naik, C. Govindarajan, S. Goel, K. Govindarajan, D. Behl, A. Singh, M. Thomas, U. Mangla, and P. Jayachandran. Closed-loop automation for 5G slice assurance. In *14th International Conference on COMmunication Systems NETworkS (COMSNETS)*, pages 424–426, 2022.

[39] NGMN Alliance. NGMN 5G White Paper. Available at `https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf`, 2015.

[40] C.-N. Nhu and M. Park. Dynamic Network Slice Scaling Assisted by Attention-Based Prediction in 5G Core Network. *IEEE Access*, 10:72955–72972, 2022.

[41] M. N. O. Manipon and https://orcid.org/0000-0003-0104-5465, neorico27.mnm@gmail.com, Saint Mary's University, Philippines. Effectiveness of ChemiCooking as a gamified intervention in nomenclature of compounds: Learning experiences of grade 11 students in a public school. *International Multidisciplinary Research Journal*, 5(1), Feb. 2023.

[42] ONAP. Offline Installer - Installation Guide. ONAP. Accessed: 2023-01-03. `https://docs.onap.org/projects/onap-oom-offline-installer/en/latest/InstallGuide.html`.

[43] M. Ramachandran, T. Archana, V. Deepika, A. A. Kumar, and K. M. Sivalingam. 5G Network Management System With Machine Learning Based Analytics. *IEEE Access*, 10:73610–73622, 2022.

[44] S. Redana, O. Bulakci, C. Mannweiler, L. Gallo, A. Kousaridas, D. Navrátil, A. Tzanakaki, J. Gutiérrez, H. Karl, P. Hasselmeyer, A. Gavras, S. Parker, and E. Mutafungwa. 5G PPP Architecture Working Group - View on 5G Architecture, Version 3.0, June 2019.

[45] R. Rokui, H. Yu, L. Deng, D. Allabaugh, M. Hemmati, and C. Janz. A standards-based, model-driven solution for 5g transport slice automation and assurance. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pages 106–113, 2020.

[46] M. Series. IMT vision–Framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU*, 2083, 2015.

[47] Siddiqi, Yu, and Joung. 5G ultra-reliable low-latency communication implementation challenges and operational issues with IoT devices. *Electronics (Basel)*, 8(9):981, 09 2019.

[48] F. Tonini, C. Natalino, M. Furdek, C. Raffaelli, and P. Monti. Network slicing automation: Challenges and benefits. In *Conference on Optical Network Design and Modeling - ONDM*, pages 1–6, 2020.

[49] E. Triantaphyllou, B. Shu, S. Sanchez, and T. Ray. *Multi-criteria decision making: An operations research approach*, volume 15, pages 175–186. John Wiley & Sons, New York, NY, USA, 02 1998.

[50] T. Underwood. All of our ML ideas are bad (and we should feel bad). Dublin, Oct. 2019. USENIX Association.

[51] I. Vaishnavi and L. Ciavaglia. Challenges towards automation of live telco network management: Closed control loops. In *2020 16th International Conference on Network and Service Management (CNSM)*, pages 1–5, 2020.

[52] A. Varga. OMNeT++. In *Modeling and Tools for Network Simulation*, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[53] H. Visser, S. Dangendorf, and A. C. Petersen. A review of trend models applied to sea level data with reference to the "acceleration-deceleration debate". *Journal of Geophysical Research: Oceans*, 120(6):3873–3895, June 2015.

[54] F. Wu, X. Li, H. Li, Q. Fan, L. Zhu, X. Wang, and V. C. M. Leung. Energy-time efficient task offloading for mobile edge computing in hot-spot scenarios. In *IEEE International Conference on Communications - ICC*, pages 1–6, 2021.

[55] M. Xie, P. H. Gomes, J. Harmatos, and J. Ordonez-Lucena. Collaborated closed loops for autonomous end-to-end service management in 5g. In *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 64–70, 2020.

[56] M. Xie, W. Y. Poe, Y. Wang, A. Gonzalez, A. Elmokashfi, J. P. Rodrigues, and P. Michelinakis. Towards closed loop 5g service assurance architecture for network slices as a service. In *European Conference on Networks and Communications (EuCNC)*, pages 139 – 143, 2019.

[57] M. Xie, W. Y. Poe, Y. Wang, A. J. Gonzalez, A. M. Elmokashfi, J. A. Pereira Rodrigues, and F. Michelinakis. Towards closed loop 5G service assurance architecture for network slices as a service. In *European Conference on Networks and Communications (EuCNC)*, pages 139–143, 2019.

[58] M. Xie, J. Pujol-Roig, F. Michelinakis, T. Dreibholz, C. Guerrero, A. G. Sanchez, W. Y. Poe, Y. Wang, and A. Elmokashfi. AI-driven closed-loop service assurance with service exposures. In *European Conference on Networks and Communications (EuCNC)*, pages 265–270, 2020.

[59] M. Xie, Q. Zhang, A. Gonzalez, P. Grønsund, P. Palacharla, and T. Ikeuchi. Service assurance in 5G networks: A study of joint monitoring and analytics. In *IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, 2019.

[60] M. Xie, Q. Zhang, A. Gonzalez, P. Grønsund, P. Palacharla, and I. T. Service assurance in 5G networks: A study of joint monitoring and analytics. In *Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1 – 7, 2019.

[61] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider. Nfv and sdn—key technology enablers for 5g networks. *IEEE Journal of Selected Areas in Communications*, 35(11):2468–2478, 2017.

[62] H. Yu, J. Yang, C. Fung, R. Boutaba, and Y. Zhuang. Ensc: Multi-resource hybrid scaling for elastic network service chain in clouds. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 34–41, 2018.

[63] G. Zhu, J. Zan, Y. Yang, and X. Qi. A supervised learning based QoS assurance architecture for 5G networks. *IEEE Access*, 7:43598 – 43606, 2019.

[64] M. Zinkevich. Rules of Machine Learning: Best Practices for ML Engineering. Google. Accessed: 2023-02-26. `https://developers.google.com/machine-learning/guides/rules-of-ml`.