H2020 5Growth Project
Grant No. 856709

D2.2: Initial implementation of 5G End-to-End Service Platform

Abstract

The main goal of 5Growth is to validate the ability of core 5G technologies to accommodate multiple advanced vertical services over common infrastructure and, importantly, to extend 5Growth´s baseline platform and architecture with innovations that fill the gaps towards meeting the requirements of the vertical use cases described in WP1.

This document describes a reference implementation of 5Growth baseline system and provides a skeleton for enhancements and innovations.

Document properties

Document number	D2.2
Document title	Initial implementation of 5G End-to-End service platform
Document responsible	Oleksii Kolodiazhnyi (Mirantis)
Document editor	Oleksii Kolodiazhnyi (Mirantis)
Authors	Xi Li (NEC), Andres Garcia Saavedra (NEC), Josep Xavier Salvat (NEC), Giada Landi (NXW), Juan Brenes (NXW), Pietro Piscione (NXW), Francesca Moscatelli (NXW), Chrysa Papagianni (NBL), Danny De Vleeschauwer (NBL), Koen De Schepper (NBL), Luca Valcarenghi (SSSA), Alessio Giorgetti (SSSA), Davide Scano (SSSA), Molka Gharbaoui (CNIT), Barbara Martini (CNIT), Koteswararao Kondepu (SSSA), Andrea Sgambelluri (SSSA), Simone Panicucci (COMAU), Chiara Napione (COMAU), Lucrezia Morabito (COMAU), Pablo Murillo (TELCA), Pedro Bermúdez (TELCA), Paola Iovanna (TEI), Fabio Ubaldi (TEI), S. Stracca (TEI), Panagiotis Kontopoulos (NKUA), Lina Magoula (NKUA), Jordi Baranda (CTTC), Luca Vettori (CTTC), Engin Zeydan (CTTC), Ricardo Martínez (CTTC), Josep Mangues (CTTC), Manuel Requena (CTTC), Josep M. Fàbrega (CTTC), Paolo Dini (CTTC), Ramon Casellas (CTTC), Oleksii Kolodiazhnyi (MIRANTIS), Konstantin Tomakh (MIRANTIS), Denys Kucherenko (MIRANTIS), Claudio Casetti (POLITO), Carla Fabiana Chiasserini (POLITO), Corrado Puligheddu (POLITO), Sonia Fernandez (TID), Diego Lopez (TID), Kiril Antevski (UC3M), Luigi Girletti (UC3M)
Target dissemination level	Public
Status of the document	Final
Version	1.0
Delivery date	May 29, 2020
Actual delivery date	May 31, 2020

Production properties

Reviewers	Andres Garcia Saavedra (NEC), Carlos Guimaraes (UC3M)

Disclaimer

This document has been produced in the context of the 5Growth Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement Nº H2020-856709.

Contents

List of Figures

List of Tables

List of Acronyms

5Gr-RL – 5Growth Resource Layer

5Gr-SO – 5Growth Service Orchestrator

5Gr-VS – 5Growth Vertical Slicer

5GT-MTP – 5G-TRANSFORMER Mobile and computing Transport Platform

5GT-SO – 5G-TRANSFORMER Service Orchestration

5GT-VS – 5G-TRANSFORMER Vertical Slicer

AI – Artificial Intelligence

CSP – Communications Service Provider

E2E – End-to-End

KPI – Key Performance Indicator

LC – Lifecycle

LCM – LC Management

ML – Machine Learning

MP – Monitoring Platform

MQ – Message Queue

NBI – NorthBound Interface

NFV – Network Function Virtualization

NFVI – NFV Infrastructure

NFV-NS – NFV Network Service

NFVO – NFV Orchestrator

NS – Network Slice

NSD – Network Service Descriptor

NSI – Network Slice Instance

NSMF – Network Slice Management Function

NSO – Network Service Orchestrator

NSSMF – Network Slice Subnet Management Function

NST – Network Slice Template

OSS – Operations Support System

PA – Placement Algorithm

PoP – Point of Presence

QoS – Quality of Service

RAN – Radio Access Network

RO – Resource Orchestrator

RT – Real Time

SBI – SouthBound Interface

SDN – Software Defined Networking

SLA – Service Level Agreement

SLPOC – Single Logical Point of Contact

SO – Service Orchestrator

UC – Use Case

UE –User Equipment

UP – User Plane

VA – Vertical Application

VIM – Virtual Infrastructure Manager

VM – Virtual Machine

VNF – Virtual Network Function

VNFD – VNF Descriptor

VNFM – VNF Manager

VoMS – Vertical-oriented Monitoring System

VSB – Vertical Service Blueprint

VSD – Vertical Service Descriptor

VSI – Vertical Slice Instance

WBI – Westbound Interface

WAN – Wide Area Network

WIM – WAN Infrastructure Manager

Executive Summary and Key Contributions

The main goal of 5Growth is to validate the ability of core 5G technologies to accommodate multiple advanced vertical services over common infrastructure and, importantly, to extend baseline 5G platforms and architecture proposals with innovations. The innovation gaps towards meeting the requirements of the vertical use cases described in D1.1 [1] were analyzed in D2.1 [3].

This deliverable presents an overview of the implementation and release details of the innovations provided for the 5Growth platform in Release 1 (May, 2020). Specifically, the innovations selected for 5Growth Release 1 are:

- **Architectural innovations:**
  - Enhanced support of verticals
    - **I1**: Radio Access Network (RAN) segments in network slices
    - **I2**: Vertical-service monitoring extensions
  - Control and Management innovations
    - **I4**: Control-loops stability
- **Algorithmic innovations**
  - **I8**: Smarter orchestration and resource control
- **Framework innovations**
  - **I12**: 5Growth Continuous Integration/Continuous Delivery (CI/CD)

The **key contributions** presented in this document are the actual software modules and modifications to existing modules in 5Growth´s baseline platform to accommodate a first version of the aforementioned innovations. The developed software is available in software repositories on GitHub following the next references [4], [5], [6], [7], [11], [12] and [13].

1. Implementation of 5G End-to-End Service Platform

1.1. High-level 5Growth Architecture

5Growth is set out to enhance the 5G-TRANSFORMER platform along the following dimensions: usability, flexibility, automation, performance and security, through the set of innovations proposed in Section 1.2. The 5G-TRANSFORMER architecture [2] is decomposed into three building blocks; the Vertical Slicer (5GT-VS), supporting the creation and management of slices for verticals; the Service Orchestrator (5GT-SO), for end-to-end service orchestration and federation of resources and services from multiple domains, and the Mobile Transport and computing Platform (5GT-MTP), acting as the underlying fronthaul and backhaul transport network infrastructure. Details on these functional components can be found in [3].

The 5Growth baseline platform rebrands the three architectural building blocks of the 5G-TRANSFORMER architecture, as depicted in Figure 1, to reflect the innovations introduced by the project, namely:

1. The 5Growth Vertical Slicer (5Gr-VS), which is inherited from 5G-TRANSFORMER Vertical Slicer (5GT-VS), acts as one-stop shop entry point for verticals to request a custom network slice. In particular the northbound interface of the 5GT-VS exposed to the vertical OSS/BSS and the functionality of the 5GT-VS will be expanded to support: (i) innovations that will enable customized per-slice capabilities for monitoring, security and performance assurance, powered by machine-leaning and analytics; (ii) verticals with more control over their slices (with emphasis on the Radio Access Network (RAN)), and (iii) additional support towards the multi-domain at the service level to request services offered by different Communication Service Providers (CSPs).
2. The 5Growth Service Orchestrator (5Gr-SO), which is inherited from 5G-TRANSFORMER Service Orchestrator (5GT-SO), provides both network service and resource orchestration capabilities in order to instantiate network slices within and across multiple domains. In particular, the 5GT-SO layer will be enhanced by novel frameworks, algorithms and architectural approaches pertaining to monitoring, security and the smart orchestration, lifecycle management and control of (federated and dynamic) slices, optimizing RAN, Transport, Core and cloud/edge computing (EC) resources.
3. The 5Growth Resource Layer (5Gr-RL), which is inherited from 5G-TRANSFORMER Mobile Transport and computing Platform (5GT-MTP), hosts all the compute, storage and networking physical and virtual resources where network slices and end-to-end services are executed. The resource layer is responsible for managing the infrastructure at the vertical sites and the required transport resources to interconnect them. 5GT-MTP will be augmented to support (re-programmable) mechanisms and algorithms to support and improve security, closed-loop management and control of individual resources in the RAN, EC, transport or core domains.

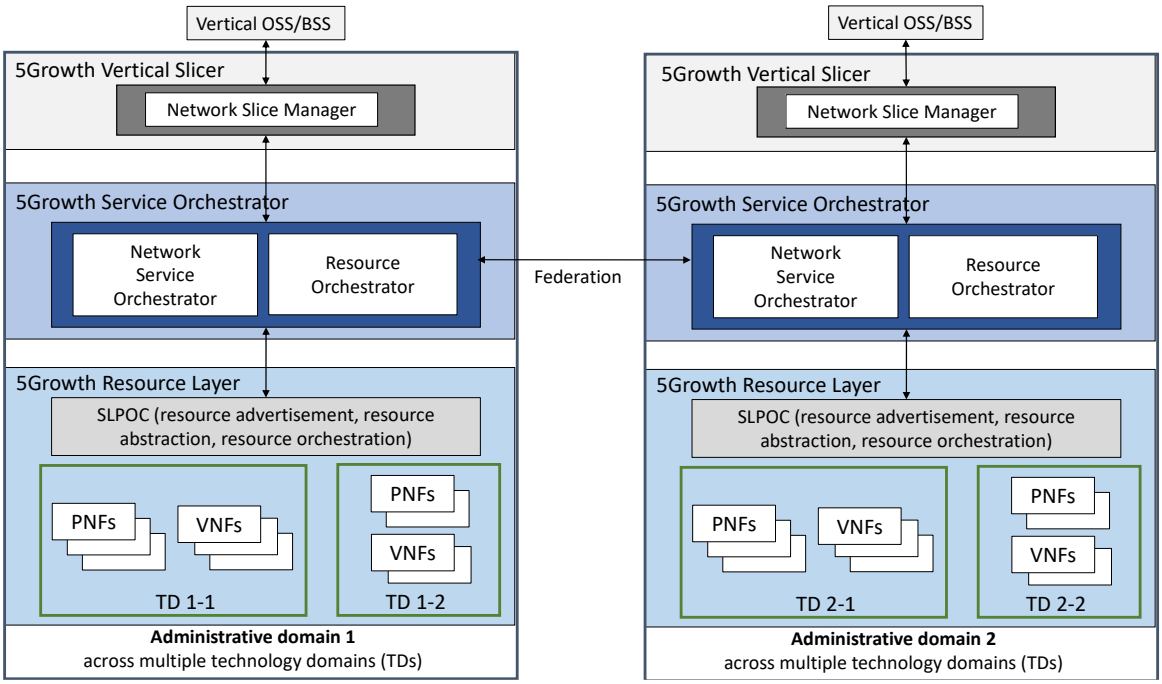

FIGURE 1: 5GROWTH BASELINE ARCHITECTURE

Apart from the enhancements per each architectural building block, 5Growth will investigate and advance cross-layer interactions and dependencies that will eventually lead to more stable, performant and secure/resilient slices and thus end-to-end services for the verticals.

1.2. Extensions over 5Growth Baseline Architecture

D2.1 [3] introduced the following set of innovations that shall extend 5Growth´s baseline platform:

- **Architectural innovations:**
  - Enhanced support of verticals
    - **I1**: Radio Access Network (RAN) segments in network slices
    - **I2**: Vertical-service monitoring extensions
  - **I3**: Monitoring orchestration
  - Control and Management innovations
    - **I4**: Control-loops stability
    - **I5**: AI/ML support
  - End-to-End orchestration innovations
    - **I6**: Federation and inter-domain
    - **I7**: Next-generation Radio Access Networks
- **Algorithmic innovations**
  - **I8**: Smarter orchestration and resource control
  - **I9**: Anomaly detection
  - **I10**: Forecasting and inference
- **Framework innovations**
  - **I11**: Security and auditability
  - **I12**: 5Growth Continuous Integration/Continuous Delivery (CI/CD)

The detailed description of each of these innovations can be found in D2.1 [3]. These innovations will be led by the different tasks T2.2 – devoted to develop an enhanced and automated vertical support – T2.3 – focused on telemetry monitoring, analytics and orchestration – and T2.4 – in charge of closed-loop automation, SLA modeling and control for vertical service lifecycle management; all conveniently wrapped up and coordinated under the umbrella of an additional task T2.1. As a result, the high-level architecture of 5Growth, accommodating these innovations is illustrated in

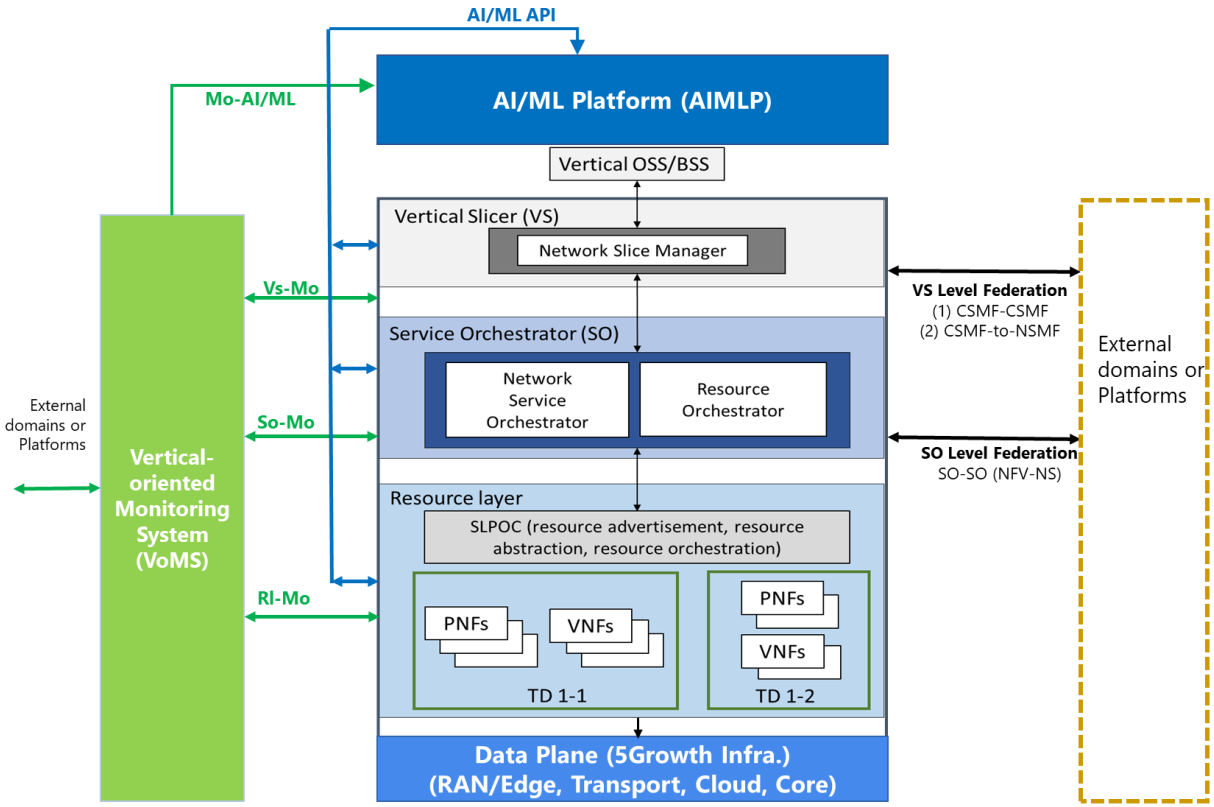

FIGURE 2: 5GROWTH HIGH-LEVEL ARCHITECTURE

1.2.1. Release plan

T2.1 has planned out design, implementation and test work to ensure a smooth evolution of the 5Growth platform (some innovations have dependencies between them) and a balanced workload across tasks T2.2, T2.3, T2.4. In addition to considering those innovations that are prioritized for the work in WP3, such plan has resulted in a 2-release plan that is summarized in Table 1.

Accordingly, Release 1 (R1), the release introduced in this document, includes a first version of I1, I2, I4 and I8. The remaining innovations, in addition to extensions to I1, I2, I4 and I8, will appear in its final version in M24 in Release 2 (R2).

TABLE 1: RELEASE PLAN

Category	Innovation	Release 1 (M12)	Release 2 (M24)
Vertical support	I1: RAN segment in network slices	X	X
	I2: Vertical-service monitoring	X	X
Monitoring orchestration	I3: Monitoring orchestration		X
Control and management	I4: Control-loops stability	X	X
	I5: AI/ML support		X
E2E orchestration	I6: Federation and inter-domain		X
	I7: Next Generation RAN		X
Smart orchestration and resource control algorithms	I8: Smart orchestration and resource control algorithms	X	X
Anomaly detection	I9: Anomaly detection		X
Forecasting and inference	I10: Forecasting and inference		X
Security and auditability	I11: Security and auditability		X
5Growth CI/CD and containerization	I12: 5Growth CI/CD and containerization	X	X

2. Release 1 innovations implementation

5Growth Release 1 adds a set of innovations on top of our baseline platform, which are summarized in Table 2.

TABLE 2: SUMMARY OF SOFTWARE RELEASE 1

Innovation	Repository	License
RAN segments in network slices	https://github.com/5growth/5gr-vs	Apache v2.0
	https://github.com/5growth/5gr-rl	GPL
Vertical-service monitoring	https://github.com/5growth/5gr-mon	Apache v2.0
Control-loops stability	https://github.com/5growth/5gr-so	Apache v2.0
Smart orchestration and resource control algorithms	https://github.com/5growth/5gr-rl/tree/master/5gr-rl-ra-server/	Apache v2.0
	https://github.com/5growth/5gr-rl/tree/master/rl/plugins/WIM/ONOS-OpenFlow-Slicing	Apache v2.0
	https://github.com/5growth/5gr-rl/tree/master/rl/plugins/WIM/ONOS-P4-Slicing	Apache v2.0

In the following, we describe in detail the software implementation of each of the innovations available at R1.

2.1. RAN segment in network slices

The support of the management of the RAN segment for network slices allows the 5Growth platform to guarantee a truly end-to-end QoS across network slices. This is done through on-demand configuration and provisioning of suitable resources in the RAN domain, based on the requirements of the vertical service and the type of the requested slice (e.g. eMBB, uRLLC, mIoT). As described in D2.1 [3], the service requirements related to the mobile traffic can be expressed through a set of attributes (e.g. coverage area, data rate in uplink and downlink, UE density and speed, latency, jitter) that are related to the specific type of network slice.

Enabling the configuration of the RAN segment as part of the workflow for the provisioning of vertical services and related network slices has several implications at the different layers of the 5Growth architecture. In particular, D2.1 proposed two possible functional architectures to support the management of the RAN configuration. At the implementation level, we have selected the solution where the RAN configuration is coordinated at the 5Gr-SO level, as depicted in Figure 3. This decision is motivated by the choice to be compliant with the fundamental principle of the 5Growth architecture where the 5Gr-VS is agnostic of the underlying resources and the resource orchestration decisions and actions are taken by the 5Gr-SO and 5Gr-RL.

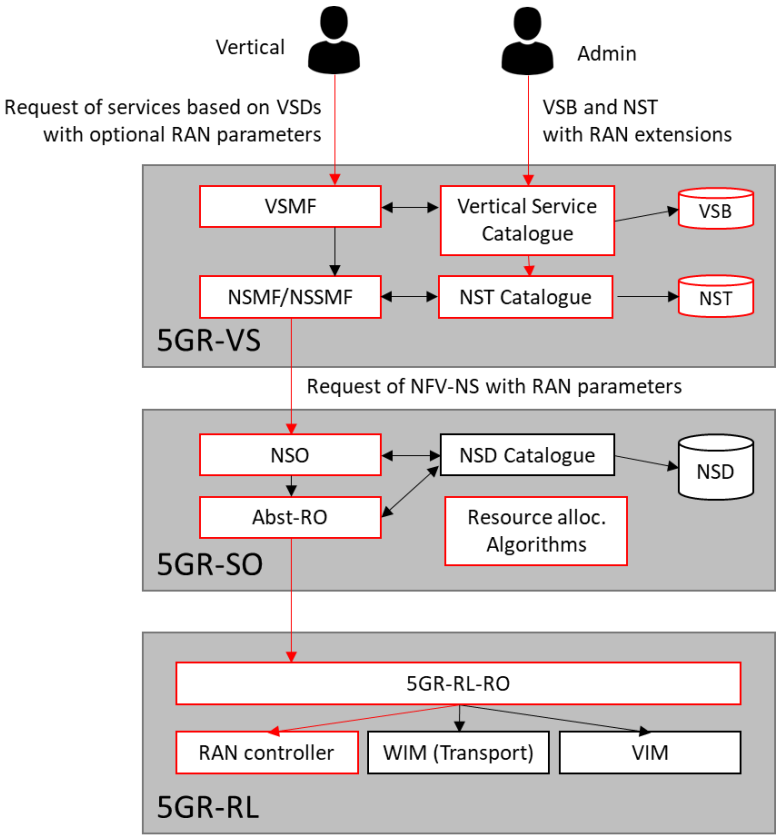

FIGURE 3: FUNCTIONAL ARCHITECTURE SELECTED FOR THE IMPLEMENTATION OF RAN CONFIGURATION SUPPORT (BOXES IN RED REPRESENT MODIFICATIONS TO BASELINE ARCHITECTURE)

As shown in the figure, which represents an update of the one proposed in D2.1 based on the latest architectural decisions, the implementation impacts all the layers of the 5Growth architecture, all its interfaces and most of the information models. In summary, the major updates in the implementation of each layer are the following:

- 5Gr-VS: support of RAN modeling in Network Slice Templates (NST) catalogue; extension of Vertical Service Blueprint (VSB) to define macro-categories of services characterized by different RAN requirements; extensions of lifecycle management, request processing and translation procedures between Vertical Service Descriptors (VSD) and NST at the Vertical Service Management Function (VSMF); extensions of Network Slice Management Function (NSMF) to request NFV Network Services (NFV-NS) with RAN parameters extensions to the 5Gr-SO.
- 5Gr-SO: support of extended 5Gr-VS requests at NSO level; support of RAN abstraction as exposed by the 5Gr-RL; additional RAN-aware resource placement and allocation algorithms; enhanced interaction with 5Gr-RL to request RAN configuration.
- 5Gr-RL: abstraction of RAN domain, including the management of related PNFs; interaction with RAN controllers through radio plugin; enforcement of RAN configuration based on the capabilities of the infrastructure.

Table 3 shows the release plan for the implementation of the functionalities mentioned above across Release 1 (released as part of this deliverable) and Release 2. The global idea is to focus Release 1

on the extensions of information models and interfaces, the overall 5Gr-VS functionalities and the 5Gr-RL functionalities related to preliminary RAN abstraction and integration with an emulated, dummy RAN plugin for testing purposes. Release 2 will introduce the 5Gr-SO internal functionalities and will evolve the 5Gr-RL procedures based on the availability of the hardware, implementing also the related interactions.

TABLE 3: RELEASE PLAN FOR RAN MANAGEMENT IN NETWORK SLICING

Release 1	**Release 2**
Support of RAN modeling in network slices at **5Gr-VS**: • NST catalogue • Update of VSB information model • Update of Translator, VS LCM and NS LCM • Update of VS-SO interface (client side)	Support of PNFD on-boarding at **5Gr-VS**
Support of enablers for RAN configuration procedures at **5Gr-SO:** • Update of VS-SO interface (server side)	Support of enablers for RAN configuration procedures at **5Gr-SO:** • Extended VS-RL interface (client side) • New algorithms for RAN-aware resource placement • Support of PNF management
Support of RAN configuration at **5Gr-RL:** • PNF support • Initial modeling of RAN abstraction • Preliminary enforcement of RAN configuration • Dummy radio plugin for integration test	Support of RAN configuration and abstraction at **5Gr-RL:** • Evolution of RAN abstraction • Extended VS-RL interface (server side) • Evolution of radio plugin (based on hardware capabilities)

The high-level software architecture of the 5Gr-VS prototype implemented in Release 1 is shown in Figure 4, highlighting the components which have been extended to support RAN management. The source code of the 5Gr-VS is available as open source, under the Apache v2.0 license, on GitHub [4].

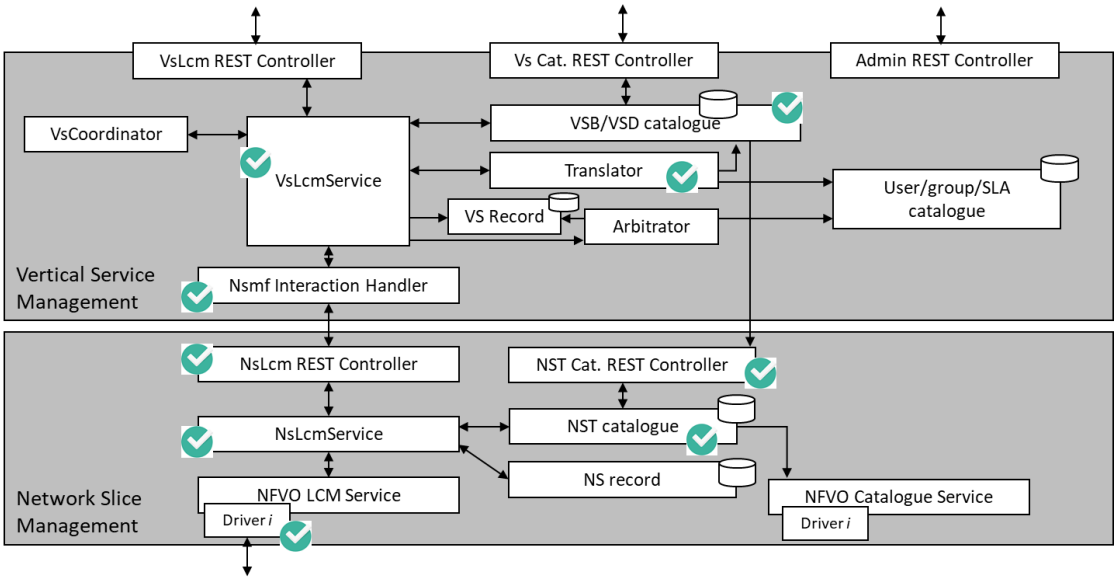

FIGURE 4: HIGH-LEVEL SOFTWARE ARCHITECTURE OF 5GR-VS, HIGHLIGHTING RAN-RELATED EXTENSIONS

The details of the extended software components are provided below:

- VsLcmService: this module is responsible for the lifecycle management of vertical services, coordinating the procedures to process the incoming requests from the 5Gr-VS NBI, to request the translation between the vertical services and network slices (interacting with the translator module) and the arbitration of concurrent vertical services (interacting with the arbitrator module), and to coordinate the requests for instantiating, terminating and scaling network slices as required by the vertical service lifecycle. This module is extended to process requests and translation procedures handling the additional RAN-related parameters.
- VSB/VSD catalogue: this module manages the PostgreSQL-based database of blueprints and descriptors of vertical services. It is extended with the new information models related to service categories at the VSB level to abstract the RAN-related service requirements. Moreover, the catalogue handles also the on-boarding of NSTs received from the 5Gr-VS NBI towards the NST catalogue at the Network Slice Management function.
- Translator: this module performs the translation between the VSD and the NST that is selected to be used for the instantiation of the network slice. The translator is updated to determine the RAN parameters of the slice profile starting from the high-level service category defined in the service blueprint.
- NSMF Interaction Handler: this module implements the driver to interact with the NSMF, which in this case is based on a REST client. The extensions are mostly related to the network slice information model.
- NsLcm REST controller: this module is a REST server that receives the requests issued by the VSMF to instantiate, terminate and scale network slice instances. As for the previous component, the extensions are related to the network slice information model.

- NST catalogue and NST catalogue REST controller: these modules implement the catalogue of the network slice templates and its REST-based interface, respectively. Their internal information model includes the RAN parameters of the slice profile in the NST.
- NsLcmService: this module is responsible for the lifecycle management of network slice instances, coordinating the procedures to process the incoming requests from the VSMF, the decisions on network slice composition and sharing (e.g. related to re-usage of existing network slice subnet instances) and the requests to the 5Gr-SO for instantiating, terminating and scaling the NFV-NS corresponding to the network slice instances. This module is extended to handle the additional RAN parameters within the network slice instances and the extended instantiation requests to the 5Gr-SO.
- NFVO LCM driver for 5Gr-SO: this driver implements the REST client to interact with the 5Gr-SO for all the requests related to the lifecycle management of NFV-NS. It is extended to encode the RAN parameters in the data related to the Service Access Point (SAP) corresponding to the connection points attached to the RAN.

The high-level software architecture of the 5Gr-RL prototype implemented in Release 1 is shown in Figure 5, highlighting the components which have been extended to support RAN management. The source code of the 5Gr-RL is available as open source, under the GPL license, on GitHub [5].

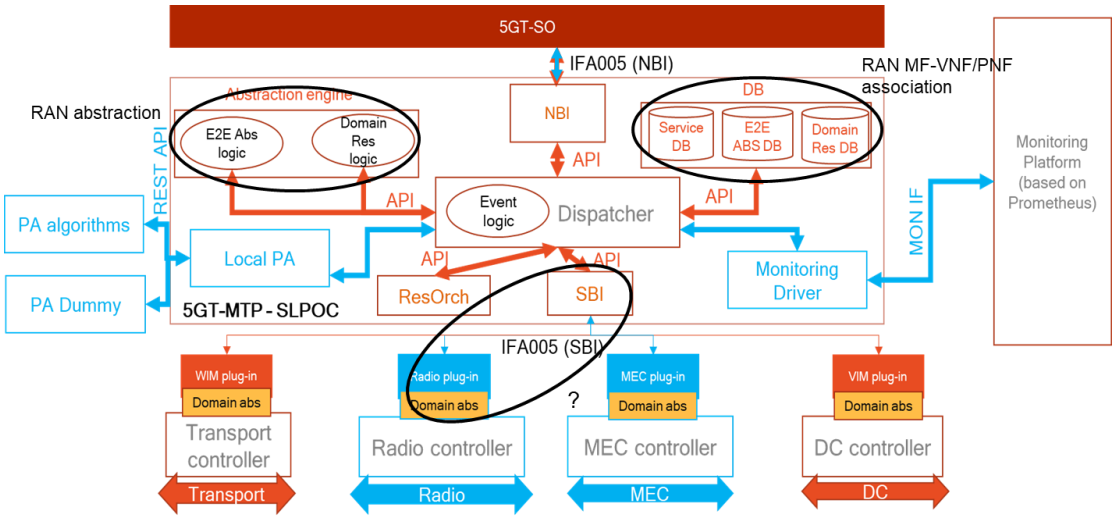

FIGURE 5: HIGH-LEVEL SOFTWARE ARCHITECTURE OF 5GR-RL, HIGHLIGHTING RAN-RELATED EXTENSIONS

The details of the extended software components are provided below.

For Release 1 there is no new module added, but some existing building blocks are extended to include the support of PNF and enforce the RAN management. Specifically, the following building blocks are extended:

- **Abstraction Engine.** This block was extended by including inside the logical abstract PoP the list of PNF that are supported by the logical PoP. The PNFs are represented by an identifier ("PNFid") and is planned for R2 release that such info is used by Placement algorithm in the SO to match the "PNFid" provided by the PNF descriptor (PNFD).

- **Database.** This block was extended to include the PNF information and more specifically the PNF supported by each domain, the mapping with the abstract PoP and the PNF status (i.e. if the PNF is activated or not). In addition, it contains any info to enforce the RAN resource management.
- **South-Bound Interface:** It was extended with new API to manage with the domain the PNF and the info to enforce RAN management. Specifically, PNF management includes 3 new API; one to retrieve the info from domain, one for activate PNF and one for deactivate the PNF. In a similar way the enforce RN management include 3 new API; one for retrieve the info, one to allocate the enforcement info and one for terminate the enforcement info.

As last part a new Radio Dummy Plugin; this plugin implements all Radio API and provides always affirmative response. It is used as stub for integration test and as a model for future radio plugin implementation.

2.2. Vertical-service monitoring

5Growth Vertical-oriented monitoring system (5Gr-VoMS) is based on the 5G-TRANSFORMER Monitoring Platform [2] with some extensions to its functionality. Initially, 5G-TRANSFORMER Monitoring Platform included the following elements:

- **Prometheus Server:** a platform that collects metrics from monitored targets,
- **Grafana:** Provides graphs and visualizations of metrics
- **AlertManager:** Processes Prometheus alerts
- **Config Manager:** a lightweight REST adapter exposing a single, simplified REST API for the configuration of the whole monitoring system (e.g. to create monitoring jobs for target monitoring parameters, or thresholds for monitoring alerts), acting as the "management" NBI of the Monitoring Platform.

The Config Manager is responsible for translating the platform-independent requests for monitoring-related management actions into requests directed to both Prometheus and Grafana, taking care of all the necessary configuration steps.

The monitoring platform was extended with the next components:

- Prometheus Pushgateway and Prometheus MQ Agent to provide Prometheus data from MQ Kafka.
- HTTP server holds RVM agent archives and initial script for virtual machines
- Logstash is the data collection pipeline tool. It collects data inputs and feeds them into Elasticsearch. It aggregates all types of data from different sources and makes it available for further use.
- Elasticsearch allows storing, searching and analyzing big volumes of data. It is mostly used in these applications as the underlying engine for implementing search task functionality. It has been adopted in search engine platforms for modern web and mobile applications. Apart from a quick search, the tool also offers complex analytics and many advanced features.

- Kibana is used for visualizing Elasticsearch documents and helps developers to have a quick insight into it. Kibana dashboard offers various interactive diagrams, geospatial data, and graphs to visualize complex queries. It can be used for search, view, and interact with data stored in Elasticsearch directories. Kibana helps to perform advanced data analysis and visualize the data in a variety of tables, charts, and maps.
- Kafka MQ is a bus for metrics and control information for RVM agents.

Elastic Search offers the possibility to collect logs from any application and configure how these logs are parsed, allowing the storage and analysis of the logs to infer specific metrics and KPIs, such as availability of applications and servers, setup times, fault detection, etc. This functionality is not allowed by the Prometheus Platform. Developed components in this innovation, a description of each developed component is provided along with their location under the code is available as open source, under the Apache license in the **5gr-mon** repository [7].

Figure 6 shows the Vertical-service monitoring architecture.

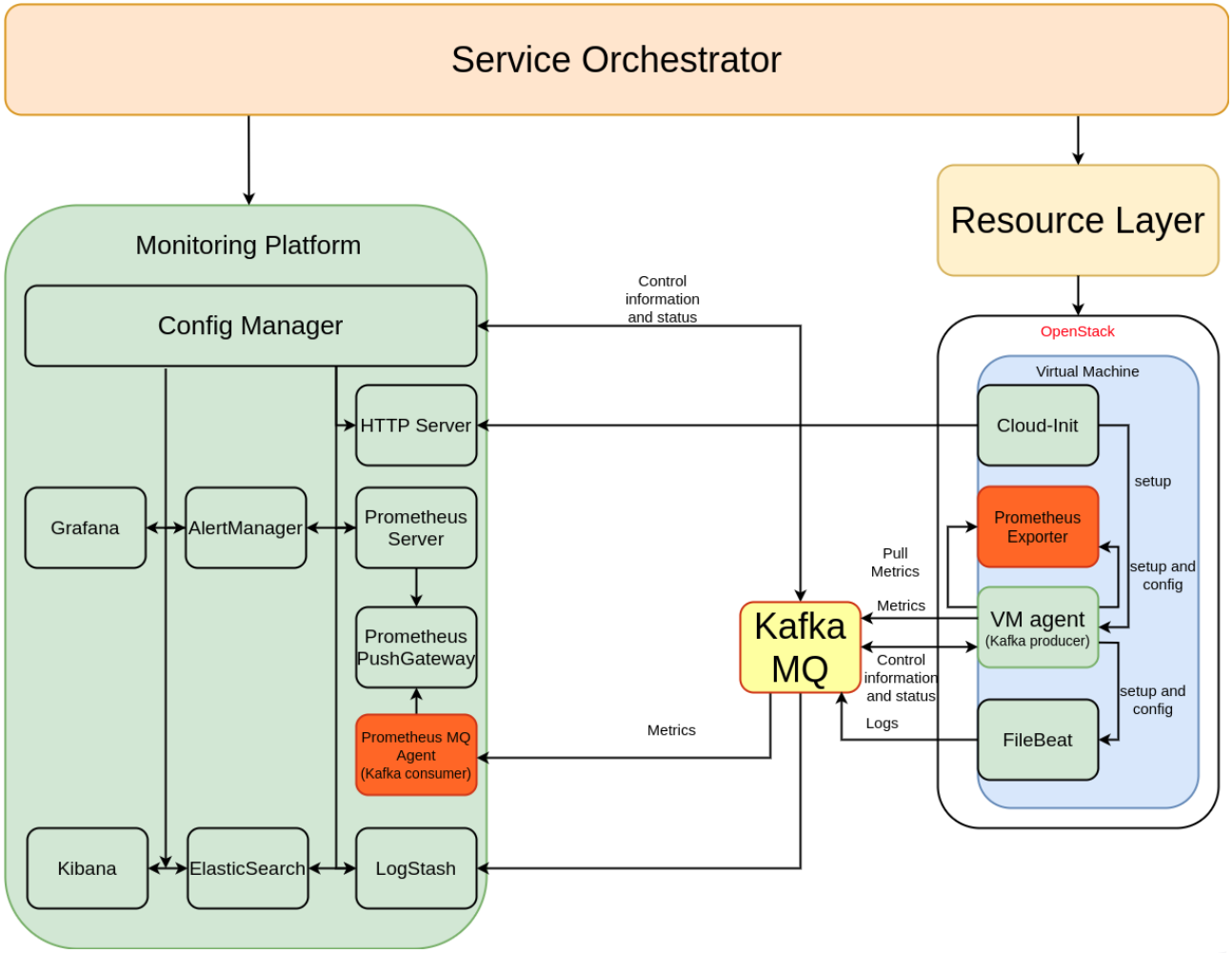

FIGURE 6: VERTICAL-SERVICE MONITORING ARCHITECTURE

The Functionality and API of the Config manager were extended. Config manager controls Kibana, ElasticSearch and Logstash. Config manager generates init script for RVM agent installation.

5Gr-SO takes task “Create VM” and makes a request "Create RVM agent" to the monitoring platform. The monitoring platform returns the response with a cloud-init script. 5Gr-SO makes a request "Create VM" with parameter cloud-init to Resource Layer. Resource Layer sends the request to VIM (Virtual Infrastructure Manager)VIM creates a VM and returns a response "VM created" to Resource Layer. Resource Layer transmits the response to 5Gr-SO "VM created". When Virtual Machine starts it loads script. The script detects the type of the operating system, downloads, and starts the necessary RVM agent. 5Gr-SO repeats request "Get VM agent status" to Monitoring Platform until it receives response "VM agent status OK". This time interval includes the following processes: get VM created, get the Operation system started and get VM agent started until it sends keepalive messages to Monitoring Platform. Figure 7 shows an RVM agent installation workflow.

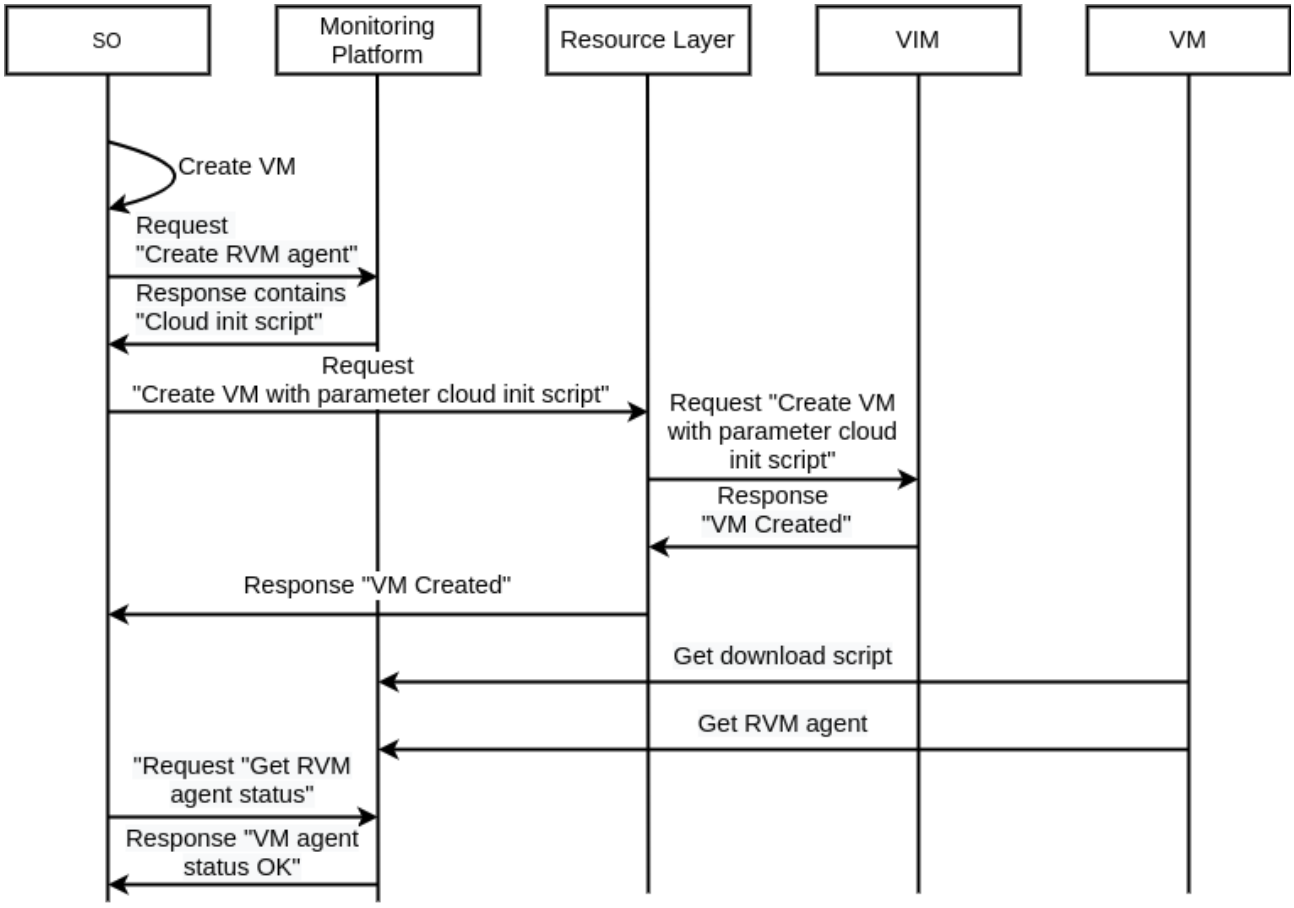

FIGURE 7: AN RVM AGENT INSTALLATION WORKFLOW

Config Manager has the possibility to execute the script and install different agents on a virtual machine by using the script, configure Prometheus collector on VM to gather metrics from Prometheus exporter and put it to Kafka's topic. Prometheus MQ Agent takes these metrics and puts them to Prometheus PushGateway. Prometheus takes collected metrics from Prometheus PushGateway. The detailed list of features is defined in Table 4.

TABLE 4: RELEASE PLAN FOR MONITORING PLATFORM

Release 1	Release 2
RVM Agent • Initially developed and added into the Monitoring Platform • Bash scripts execution • Keepalive support • Prometheus collector's support • Configuration reliability (avoid loss after restart)	**RVM Agent:** • Python script run support
Configuration Manager • REST API developed to support RVM agent installation • Functionality for generating RVM agent installation script • HTTP server integrated for storing RVM agent archives • Support bash script execution via RVM agent • Keepalive support • Prometheus collectors support for RVM agent • REST API ELK stack support • Kibana auto dashboard instantiation extension for the configuration manager. • Automatic configuration of Kafka topics in the logstash pipelines to insert data in the Elastic	**Configuration Manager:** • Python scripts run supported via RVM agent
Service Orchestrator • RVM agent installation • Remote command execution through RVM agent • Prometheus collectors creation/deletion	**Additional components:** • Native and 3rd-party client-side monitoring probes • Prometheus exporter for collecting customer metrics (PECCM) • KPI measurement support
Network service descriptor • Definition of Config manager additional functions	
Monitoring Architecture • ElasticSearch stack integration • Filebeat probe with kafka • Logstash log parser from kafka • Kibana visualization tool integration • Elastalert Alert manager integration • Two new network monitoring probes, IPFIX Collector and Packetbeat.	

2.3. Control-loops stability

The support of closed control-loops enables the 5Growth platform to satisfy the established Service Level agreements with Vertical industries' users in an automated way across the system and throughout the whole life-cycle of the service. As described in D2.1 [3], at the architecture level, the control loops can be placed within a layer, or across layers, or even across domains. The logic of these close control-loops is comprised of three steps: (1) collecting monitored data, (2) perform real-time data analytics to infer anomalies, forecasts or contexts and then decide proper actions for optimization and reconfiguration of the service and/or the system (such as auto-scaling, self-healing, etc.) using AI/ML techniques, and (3) perform the actions in the system.

Enabling the configuration of control-loops as part of the workflows for the management of vertical services requires the interaction with a new entity in the 5Growth architecture, which is the 5Gr-AI/ML platform (still under development and scheduled for Release 2) to provide a set of AI/ML models to support the decision-making process. In Release 1, the scope has been focused on the implementation of a closed-loop in the 5Gr-Service Orchestrator (5Gr-SO) to handle the AI/ML-based scaling of NFV-NS case. In this work, multiple concepts have been explored, for instance applying third-party software tools for distributed large-scale data processing (Apache Kafka, Apache Spark) and defining the necessary interactions with the 5Gr-AI/ML platform, the monitoring platform as well as the rest of the 5Growth platform. The same methodology can be applied to close control-loops in other layers, like the 5Gr-RL to predict and react in case of problems in the transport infrastructure.

Before explaining the modification introduced in the software of the 5Gr-SO and the 5Gr-VoMS platform, we present a new information element (IE), which has been introduced in the Network Service Descriptor (NSD) to support the AI/ML-driven scaling operation. In general, it will be also used to define other AI/ML-driven operations for required use cases. This new IE is as follows:

```
"aimlRules" : [
  {
        "ruleId": "aiml_rule1", # Identifier of the aiml rule
        "problem": "scaling",  # Specification of the problem to handle
        "nsMonitoringParamRef": ["mp1"] # Set of needed monitoring parameters to handle problem
      }
]
```

Table 5 summarizes the functionalities implemented in Release 1 (released as part of this deliverable) and planned features for Release 2. The overall idea is to focus Release 1 on the extensions of information models and interfaces, the overall 5Gr-SO functionalities related to AI/ML-based scaling operations of NFV-NS. Release 2 will be focused on developing the interfaces to the 5Gr-AIML Platform (which will be introduced in Release 2) and the design of more advanced AI/ML-driven scaling operations at the 5Gr-SO, and/or additional AI/ML-driven closed-loop operations at the 5Gr-VS or the 5Gr-RL layer.

TABLE 5: RELEASE PLAN FOR CLOSED-LOOP INNOVATION

Release 1	**Release 2**
Implement a closed-loop for AI/ML based scaling of NFV-NS at 5Gr-SO: • Update NSD with new IE "aimlRules" • Extend SLA Manager for an AI/ML driven orchestration of the scaling process • Extend Monitoring Manager at 5Gr-SO for assisting the SLA manager in the configuration of monitoring actions' related elements • Extend SOEc to interact with the extended SLA Manager • Additional element "AI/ML repository" • Initial AIML model using Spark MLlib as a closed-loop enabler • Exploit Apache Spark for a continuous NFV-NS performance inspection and trigger the corrective actions	• Define and implement 5Gr-AIML Platform interfaces with the rest of 5Gr building blocks (5Gr-VS, 5Gr-SO, 5Gr-RL). This work will be done in collaboration with I5. • Design additional AI/ML-driven closed-loop operations (in collaboration with I8) for more advanced AI/ML-based scaling at the 5Gr-SO, and/or the closed loops at other layers (5Gr-VS or 5Gr-RL).
Support detection of SLA deviations at 5Gr-Monitoring platform: • Create "scrapper" elements to map monitoring data to Kafka topics in Config-Manager using Apache Kafka streaming platform • Extending the API of the monitoring platform to enable the 5Gr-SO to make requests to the Monitoring platform to create/delete such "scraper" objects.	• Extend the capability of the 5Gr-Monitoring platform to collect more service-related performance metrics, e.g., to collect application-level metrics, to support more scaling operation options. • Define and implement the required interactions and interfaces of 5Gr-Monitoring platform with the 5Gr-AIML Platform (in collaboration with I5 and I2) to support the developed closed-loop operations.

The high-level software architecture of the 5Gr-SO prototype implemented in Release 1 is shown in Figure 8, highlighting the components that have been extended to support AI/ML-based scaling operations of NFV-NSs. The source code of the 5Gr-SO is available as open source, under the Apache v2.0 license, on GitHub [6].

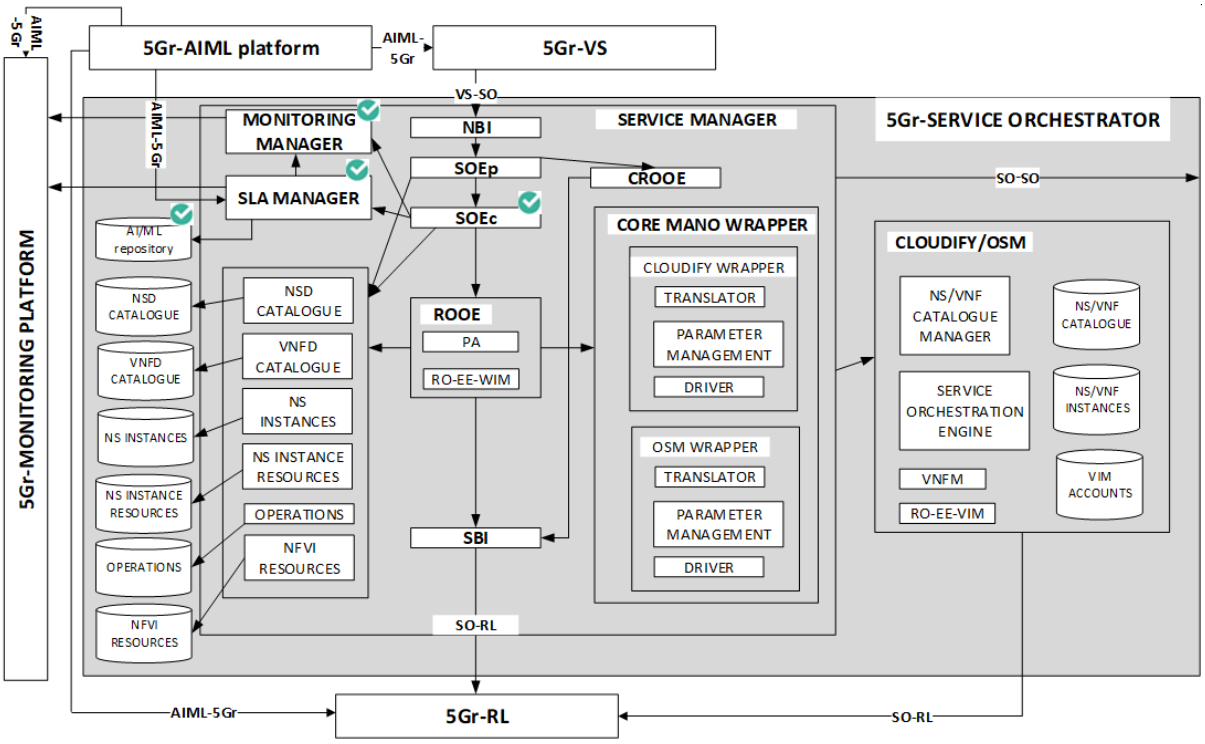

FIGURE 8: HIGH-LEVEL SOFTWARE ARCHITECTURE OF 5GR-SO, HIGHLIGHTING CLOSE-LOOP EXTENSIONS

The details of the highlighted components are provided below:

- **SLA Manager**: This module is responsible for handling SLA compliance for a given NFV-NS and triggering the scaling process as a reaction in case of an SLA violation. Initially, the SLA management was carried out with an alerting system configured in the 5Gr-VoMS in charge of detecting deviations on the monitored parameters of interest. This module has been extended to orchestrate the scaling process based on AI/ML models and processing routines provided by the AI/ML platform. This orchestration is executed as a continuous process over the entire NFV-NS runtime. This implies a closed-loop including, mainly, (i) the detection of AI/ML scaling based operation expressed in the NSD; (ii) the interaction with the 5Gr-Monitoring platform through the monitoring manager to filter the required monitoring data to the AI/ML model in Apache Spark [8]; (iii) performing AI/ML action; and (iv) detection if the SLAs being compliant and react in case of a violation.
- **Monitoring Manager**: It is part of the 5Gr-SO in charge of interacting with the monitoring platform to manage monitoring jobs. That is, the Monitoring Manager is in charge of the collection of performance metrics expressed in the NSD and configuration of visualization dashboards. This module has been extended to assist the SLA manager in the configuration of additional elements related to monitoring actions to assist in the close-loop AI/ML based-scaling process. Namely, the monitoring manager contacts the 5Gr-Monitoring Platform to (i) create/delete "data spaces" (named Topics) in Apache Kafka [9] to insert the required monitoring data to assist the close-loop AI/ML-driven scaling operation; and (ii) to create/delete "data scrapers" that filters the required monitoring data out of all monitoring data available belonging to the same NFV-NS or other NFV-NSs.

- **Service Orchestrator Engine child (SOEc)**: This sub-module of the Service Orchestrator Engine handles the service orchestration of regular NFV-NSs (i.e., a non-composite NFV-NS or a single nested NFV-NS included in the composite NFV-NS). This module has been extended to interact with the new capabilities of the SLA Manager module.
- **AI/ML repository:** this element is new in the architecture and it is an internal folder where the SLA manager stores the requested AI/ML models and processing routines required to launch the Apache Spark jobs in charge of checking the SLA compliance. These AI/ML models and processing routines are requested for the AI/ML platform[1].

Regarding the monitoring platform, in addition to the modifications explained in Section 2.2, it has been extended to support the close-loop AI/ML operation. In addition to using the available Apache Kafka streaming platform, the Config-Manager in the 5Gr-VoMS platform offers the possibility to create "scraper" elements. These elements filter out the monitoring data available in the 5Gr-platform and insert them in the requested Kafka topic to be ingested by the Apache spark process checking continuously the performance of the NFV-NS against the AI/ML module to detect SLA deviations and trigger the corrective actions.

The monitoring platform API has been extended so that the 5Gr-SO is able to make requests to the Monitoring platform to create/delete such "scraper" objects. Namely, the requests are implemented with POST method and DELETE method. The request/response body includes a set of parameters related to NFV-NS instance and the associated VNF, required performance metrics, time interval, and in addition an extra parameter "scraperId". ScraperId is a Prometheus scraper identifier. The Prometheus Scraper object sends the performance metric to Kafka's topic for specific Network Service and VNF. The source code of the 5Gr-VoMs is available as open source, under the Apache v2.0 license, on GitHub [7].

[1]Due to the on-going tasks related to the development of the AI/ML platform (I5), the interaction between 5Gr-AI/ML platform is not yet defined in R1. Thus in R1, the trained AI/ML models and processing routines are on-boarded and available in the AI/ML repository.

These requests are as shown below:

Request method: **POST**

URL: **/prom-manager/prometheus_scraper**

Request body:

```
{
  "nsid": "fgt-9f62028-cbe4-4d07-b89d-e39d07943175",
  "vnfid": "webserver",
  "performanceMetric": "VcpuUsageMean",
  "kafkaTopic": "fgt-9f62028-cbe4-4d07-b89d-e39d07943175_scaling",
  "interval": "15"
  "expression": "avg by(vnfdId, nsId) (1 -
(irate(node_cpu_seconds_total{mode="idle"}[1m]))) * 100"
}
```

Where:

"nsid": is Network Service (NS) identifier,

"vnfid": is Virtual Network Function (VNF) identifier,

"performanceMetric": performance metric which will be tracked and published to "kafkaTopic". This performanceMetric is specified at the NSD to be used for the AI/ML "scaling" problem,

"interval": is a parameter that defines how often Prometheus scraper takes performance metric from Prometheus and sends it to Kafka's topic.

"expression": The prometheus query language expression for calculating performance metric. This expression has been generated when the different dashboard for the NFV-NS have been created and can be reused for the AI/ML based scaling problem

The response body includes all these parameters and an extra parameter "scraperId". ScraperId is a Prometheus scraper identifier. The Prometheus Scraper object sends the performance metric to Kafka's topic for specific Network Service and VNF.

5Gr-SO makes a request that is shown below to delete the Prometheus scraper object.

Request method: **DELETE**

URL: **/prom-manager/prometheus_scraper/{scraperid}**

Request body: No request body, all parameters are provided through URL.

Where :

"scraperid": is a Prometheus scraper identifier

The response body in a case of success:

```
{
  "deleted": [
  "scraperid"
  ]
}
```

And in case of false:

```
{
  "code": "404",
  "error": "Not Found",
  "description": "No such PrometheusScraper"
}
```

2.4. Smart orchestration and resource control algorithms

5Growth aims at fully automated network slice lifecycle management and SLA enforcement. In addition to the architectural innovations to achieve so, novel algorithms and techniques are needed. Introduced innovations enable the optimized deployment, management and control of elastic network slices (and their resources) that support dynamic traffic/workload demands and their respective SLAs. In correspondence to the targets set in D2.1 [3], in Release 1 we focus on the 5Gr-RL (see Figure 9).

The 5Gr-RL supports the allocation/de-allocation of logical resources (i.e., NFVI PoP's VNFs and Logical Links, LLs) over the underlying infrastructure, as selected by the 5Gr-SO. It coordinates a pool of dedicated cloud and network resource and domain controllers (i.e., VIMs and WIMs) to: i) retrieve controller-specific information (e.g., topology, resource availability and capabilities, etc.) used for deriving/composing a global infrastructure view; ii) allocate/de-allocate NFVI PoP and WAN resources (i.e., VNFs and LLs) when managing the lifecycle of 5Gr-SO network services (i.e., VNFs and LLs). In R1, 5Growth RL meets the following targets:

- Introduces on-line resource allocation algorithms, providing optimized end-to-end resource allocation across wireless, packet and optical switching domains.

- Enables data plane customization per network slice, supporting performance isolation via real-time control and traffic management.

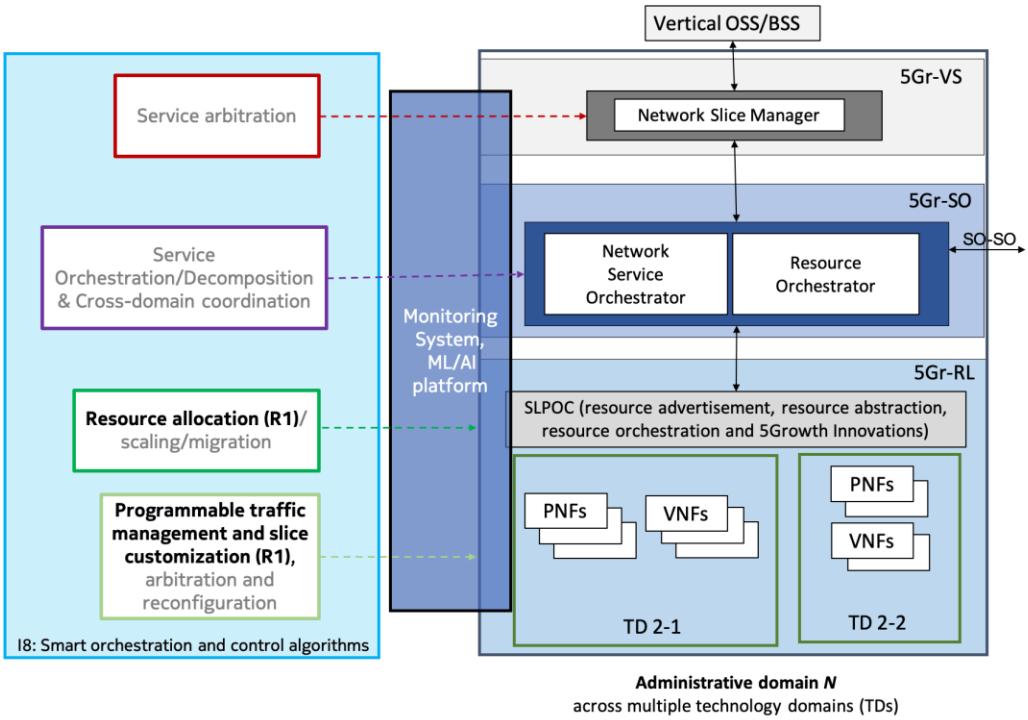

FIGURE 9: 5GROWTH I8 INNOVATIONS TOWARDS SMART ORCHESTRATION AND RESOURCE CONTROL

2.4.1. Advanced resource allocation mechanisms at 5Gr-RL

The 5Gr-RL operates with a higher granular resource view than the upper-layer 5Gr-SO layer. This leads the 5Gr-RL to handle:

- Abstraction computation to derive the LLs between pairs of NFVI PoPs as well as providing aggregated resource information at each NFVI PoP.
- Expansion resource selection and computation (e.g., WAN paths) fulfilling the requirements (such as required bandwidth, maximum latency, etc.) needed over the set of LLs supporting a network service.

Those functions were supported in the 5GT-MTP entity using a modular architecture where the MTP core process interacts, via a defined REST API, with a standalone process referred to as Placement Algorithm (PA) server. The latter is the responsible for conducting the abstraction and expansion computations triggering tailored algorithms targeting heterogenous objectives such as efficient use of the overall resources (e.g., WAN bandwidth), minimizing end-to-end latency, etc. 5Growth enhances this architecture where the 5Gr-RL replaces the 5GT-MTP and an improved Resource Allocation (RA) server replaces the 5GT PA server. The REST API between the 5Gr-RL has been enhanced to handle novel advanced mechanisms at the RA server. This allows supporting new capabilities with respect to the 5GT deployment such as restoration, re-optimization, or re-programmability of allocated resources upon decisions derived from innovative closed-loops.

Figure 10 shows the architectural view of the 5Gr-RL interactions with the 5Gr-SO and the underlying controller. The interfaces supporting these interactions (i.e. 5Gr-SO SBI and 5Gr-RL SBI) are mostly based on those already used in 5GT architecture. However, the interworking between 5Gr-RL core and the standalone RA server is based on an enhanced REST API available in [10]. This interface indeed establishes a client-server relationship to request and respond to both expansion and abstraction computations. The new capabilities of the REST API enable that multiple (expansions or abstraction) computations can be requested in a single request-response exchanged messages. That is, a bulk of computation requests can be sent from the 5Gr-RL core and managed by the RA server. To do that, two algorithms have been deployed: i) Class of Service Algorithm (CSA); ii) Infrastructure Abstraction (InA) algorithm. The description of both algorithms is described briefly below.

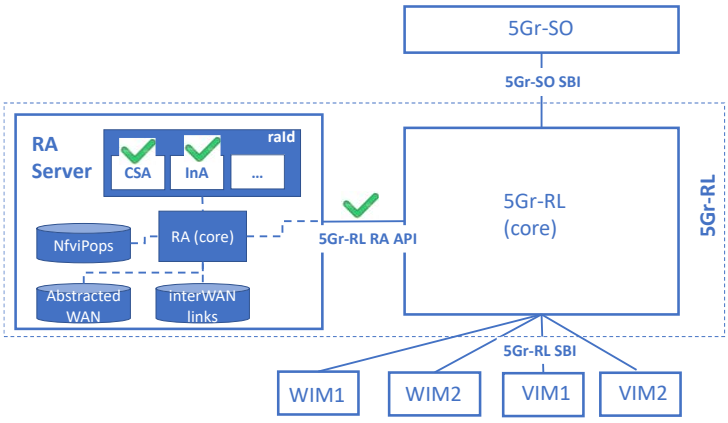

FIGURE 10: 5GR-RL – RA SERVER INTERACTION WITH THE DEPLOYED R1 ENHANCEMENTS

The implementation of the new RA server for the Release 1 and the supported algorithms for both expansion and abstraction computations (i.e., CSA and InA) are highlighted in Figure 10. The source code of the RA server is available as open source, under the Apache v2.0 license, on GitHub [11].

Basic features of the RA server CSA and InA algorithms are described in the following. Further details addressing the required interactions with the 5Gr-SO, 5Gr-RL and WIM are described in Annex 1.

- **CSA**: this algorithm is indicated in the request (POST method) message sent by the 5Gr-RL to the RA server. In the request message it is also carried: i) the endpoint Provide Edge (PE) nodes attached to the source and destination NFVI PoPs ii) the requirements (i.e., bandwidth and latency); iii) the retrieved WAN(s) topologies and resource status gathered by the 5Gr-RL; iv) inter-WAN links' attributes (if two or more WAN domain are involved). The CSA algorithm outputs a feasible WAN path (if such exists) between the PE endpoints fulfilling the connection requirements and attaining an efficient use of network resources (in terms of bandwidth allocation). In brief, the CSA algorithm deals with the expansion computation for a given LL selected by the 5Gr-SO.
- **InA**: this algorithm is indicated in the request (POST method) message sent by the 5Gr-RL to the RA server. The request message may contain a bulk of connections (included in a request list) between multiple PE endpoints attached to different pairs of NFVI PoPs. The remaining information carried in the request message is like that listed for the CSA algorithm. The InA algorithm seeks up to K WAN paths between every requested pair of PEs. Such WAN paths

are computed without specific requirements to be fulfilled since the objective is to output the WAN paths attaining the shortest number of links, having the highest available bandwidth and lowest end-to-end latency. The resulting WAN paths are then used at the 5Gr-RL to derive the required abstraction computation of the pool of LLs between NFVI PoP pairs passed to the 5Gr-SO.

2.4.2. Network slice customization and performance isolation

The 5Gr-RL has been updated to support the application of QoS policies to the networking devices from a given resource domain overseen by the 5Growth platform. More in detail, five software extensions have been implemented at different levels of the 5G-RL, as depicted in Figure 11.

- First, a Smart Orchestration Algorithm has been implemented at the RO component, which decides the optimal QoS policy - in this case the "baseline policy" is performance isolation - to apply to a certain Network Slice depending on specific performance requirements.
- Second, two external Software-Defined Networking applications have been provided to configure different QoS mechanisms at the data plane level, the ONOS Slicing application for OpenFlow switches and the ONOS-P4 Slicing application for P4-programmable software/hardware targets coupled with the corresponding P4 program that implements the policy at the target.
- Third, the SBI component has been extended to support the communication between the Smart Orchestration Algorithm and the external QoS applications, providing the Smart Orchestration Algorithm with the possibility to choose among a set of QoS parameters and request their enforcement to the external QoS applications.
- Fourth, due to the fact that the Resource Orchestrator is stateless, the Database component has been extended to allow storing the QoS policy applied to a specific Network Slice. Maintaining such info at 5Gr-RL is necessary, in case that, at a later point in time and due to changes in the network state, the QoS Policy assigned no longer satisfies the associated SLA Parameters. In this case, the QoS Policy needs to be retrieved from the Database and updated accordingly, in order to enforce again the mentioned performance SLA.
- Finally, the Monitoring Driver component is extended, with the purpose of configuring alarms using the existing REST API exposed by the 5Growth Monitoring platform. These alarms are set to monitor the performance requirements enforced by the QoS Policies, assigned by the Smart Orchestration algorithm. If these requirements are no longer met, due to changes in the network state, the alarm is triggered and the Monitoring Driver sends a notification to the Resource Orchestrator, along with the corresponding measured value(s) and the Network Slice(s) affected. This notification triggers again the performance isolation workflow, in order to update the QoS policy assigned to the Network Slice, in order to adjust it to the new data plane state.

The source code of the 5Gr-RL extensions and the control/data plane components are available as open source, under the Apache v2.0 license, on GitHub [12][13].

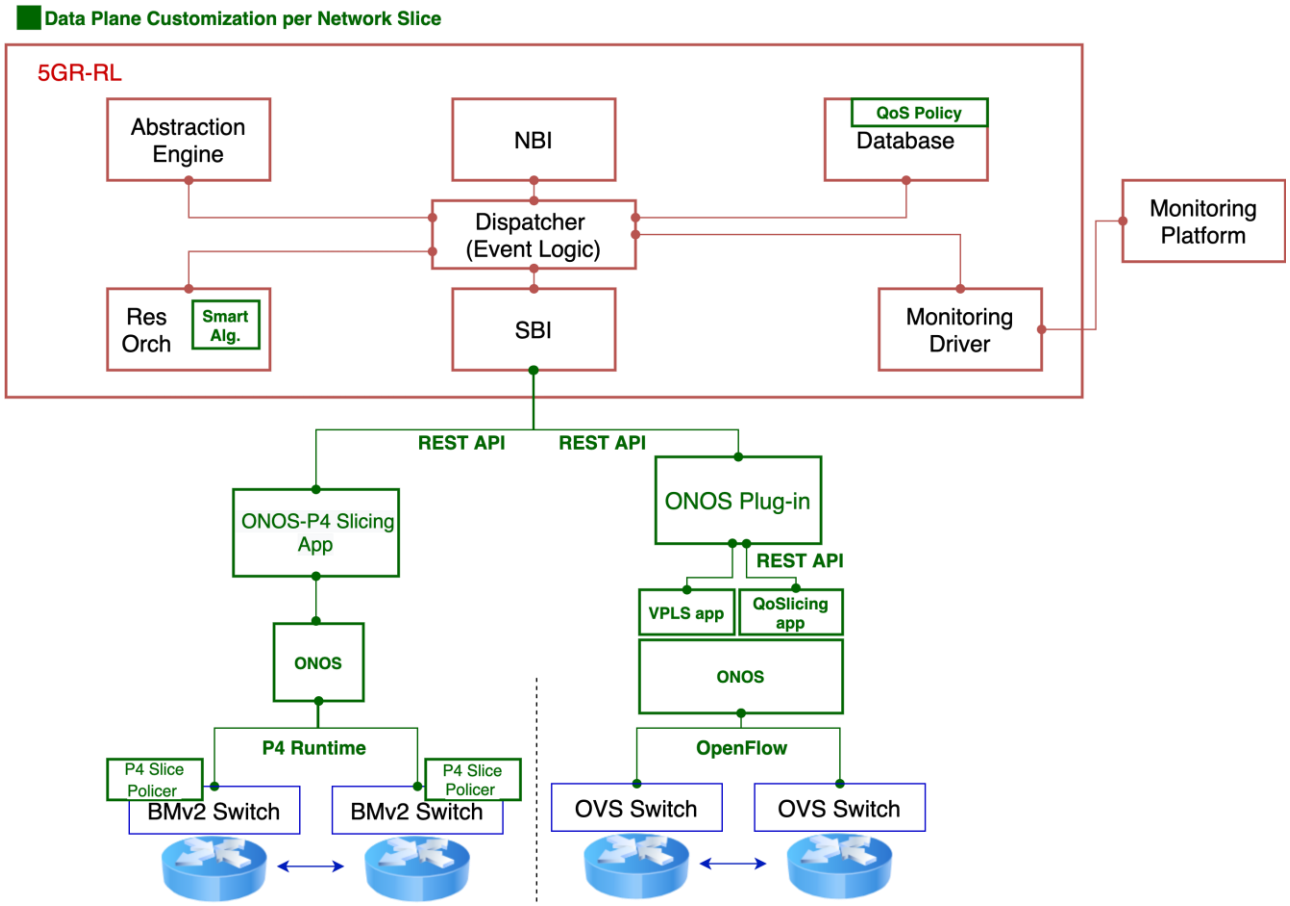

FIGURE 11: PERFOMANCE ISOLATION: 5G-RL AND CONTROL/DATA-PLANE EXTENSIONS

The description of the extended software components is provided below:

- **Smart Orchestration Algorithm**: Slice isolation in terms of performance and management (not only in terms of connectivity isolation) is an essential requirement for network slicing. Service level key performance indicators should be met for each slice, independently of the congestion level of other slices sharing the same physical resources. Slices should be independently managed/controlled and customized, to meet their functional and performance requirements. Performance isolation for each slice is triggered by the RO, upon receiving a LL allocation request containing its maximum capacity. In order to apply this baseline QoS policy, the RO simply (i) stores the "network slice - QoS policy" association to the 5Gr-RL Database (ii) triggers LL creation through the SBI, annotating the request with low level information if needed (e.g., rate limit, max burst size) and (iii) sets up the corresponding alarms through the monitoring driver (e.g., link failure, packet drop rate/throughput thresholds). These parameters will be measured using different networking probes that will be provided by the 5Growth monitoring platform for R2.
- **ONOS-OpenFlow slicing**: this solution uses OpenFlow protocol to deploy network slices providing both connectivity isolation and performance isolation. The current ONOS distribution (i.e., version 2.4) already includes an application to create network slices enforcing connectivity isolation (i.e., the ONOS VPLS app in Figure 11). However, the VPLS app does not

allow to specify requirements for the requested slices and does not implement performance isolation among the created slices. Therefore, we have implemented a novel ONOS application (i.e., the ONOS QoSlicing app in Figure 11) that interacts with both the ONOS core and the VPLS app for applying the needed QoS guarantees (i.e., reserving a bandwidth for each slice) to the slices created using the VPLS app. The two ONOS apps have been integrated into the 5GR-RL system utilizing a dedicated Plug-in (i.e., ONOS plug-in in Fig. 10).

- ONOS Plug-in: This plug-in implements the integration between the SBI of the 5Gr-RL and the ONOS SDN controller. The plug-in uses REST to expose four IFA005 interfaces which are the Query Virtualised Network Resource (), the Query Virtualised Network Resource Capacity (), the Allocate Virtualised Network Resource () and the Terminate Virtualised Network Resource (). The client side of such interfaces is integrated into the 5Gr-RL while on the server side, each interface is translated into a set of instructions capable of directly interacting with ONOS. More specifically, the plugin is able to receive a request message (POST method) from the 5Gr-RL with the parameters required to set up a network slice with specific QoS characteristics. Once the request is parsed and the necessary parameters are extracted, the plug-in sends two different requests (POST method) to ONOS: one for setting up the slice and one for enforcing the QoS guarantees (as detailed in the next points).
- ONOS VPLS App: This ONOS application is included in the ONOS distribution. It allows creating network slices including an arbitrary number of endpoints. The end-points located in the same slice can exchange both broadcast and unicast traffic; end-points located in different slices are not allowed to communicate. For this project, this app has been extended with a REST API interface; this interface has been contributed to the ONOS community and already merged in the official distribution (version 2.4). Specifically, the current parameters that can be configured through the body of the exposed POST REST method are the following:
  - **Slice ID**: identifier of the network slice.
  - **Communication endpoints**: identifiers of the endpoints of the Network Slice (i.e., not limited to two endpoints). In the current implementation, the identifiers are the interfaces of data-plane devices; the VLAN tag of incoming traffic and a list of associated IP addresses can be also specified.
- ONOS QoSlicing App: This ONOS application has been specifically developed for this project. The QoSlicing app enforces performance isolation to slices previously created using the ONOS VPLS app. Performance isolation is applied dynamically creating OpenFlow meters in the OpenFlow-based switches. The app exposes a REST interface toward the ONOS Plug-in to receive the slice parameters to be enforced on the data plane. Specifically, the current parameters that can be configured through the body of the exposed POST REST method are the following:
  - **Slice ID**: the identifier of the network slice previously created with the VPLS app to which the performance isolation has to be applied

  - **Maximum capacity**: the maximum bitrate, expressed in KByte per second, that is assigned to the Network Slice.
  - **Burst size**: the maximum amount of traffic that can exceed the maximum capacity assigned, expressed in KByte.

  The integrated solution including the ONOS Plug-in, the ONOS VPLS app and the ONOS QoSlicing app has been tested in an emulated scenario using the Mininet tool. Some patches have been applied also to the ONOS core for supporting the applications of OpenFlow meters to the Intents generated by the ONOS VPLS app. Therefore, a full release of ONOS v2.4 has been uploaded to the project git, including the required modifications.

- **ONOS-P4 Slicing:** this solution uses data-plane programmability as an alternative for deploying network slices providing performance isolation. The ONOS-P4 Slicing App is an ONOS application capable of populating dynamically the programmable forwarding tables of P4-based switches, in order to classify the traffic in different network slices with different QoS parameters. Currently, it supports the configuration of a P4 meter per slice to perform end-to-end rate limiting. A P4 program (P4 Slice Policer) is coupled with the application, which defines the forwarding tables and pipelines of the P4-based switches. This definition includes the application of P4 meters, which are leveraged to perform rate-limiting traffic policing, that is monitoring network traffic for compliance with the maximum rate of traffic received (number of packets per second and maximum burst size) on an interface for a particular network slice and taking steps to enforce that rate. Network slice instance identification at the data plane (local to the switch) is used to make local decisions on forwarding policies, QoS mechanisms, etc. The performance requirements of a network slice instance can, therefore, be met by making the correct decision. Common practices (i.e., macvlan) and tunneling using programmable data plane techniques have been adopted to ensure connection isolation. The ONOS-P4 Slicing App uses the P4Runtime protocol to populate dynamically the forwarding tables defined by the coupled P4 program. Additionally, the ONOS application exposes a REST API that allows external components to configure the data plane to achieve slice performance isolation. Specifically, the current parameters that can be configured through the body of the exposed POST REST method are the following:
  - **Communication endpoints**: identifiers of the source and destination endpoints of the Network Slice to which apply the performance isolation. In the current implementation, the identifiers are the IP addresses of the hosts communicating through the Network Slice.
  - **Slice ID (optional)**: identifier of the Network Slice used locally at the data plane to identify the end-to-end communication flow between the communication endpoints provided above. The Slice ID is needed to identify different communication flows between the same endpoints (i.e., to apply different performance isolation policies to different types of communication between the same endpoints, e.g. video and voice). If none is provided, a locally generated identifier is used to perform slice identification.

- **Path**: local physical topology of the Network Slice. List of identifiers of the P4 switches that form the physical path of the Network Slice.
- **Maximum capacity**: the maximum sustainable bitrate, expressed in bits per second, that is assigned to the Network Slice.
- **Burst size**: the maximum number of bits that can instantaneously exceed the sustainable bit rate assigned.

When receiving these parameters, the ONOS application selects a meter ID among the P4 meters pre-configured in the P4 switches (i.e. mapping between meter ID and performance parameter values *maximum capacity* and *burst size*) that best adjust to the *maximum capacity* and *burst size* parameters received in the POST method and populates the P4 tables of the switches given in the *path* parameter to 1) perform the routing between the *communication endpoints* and 2) rate-limit the capacity of the Network Slice.

2.4.3. Roadmap

Table 6 summarizes the Release 1 deployments covering (i) the RA server and the embedded mechanisms/algorithms along with the supporting functionalities (e.g., 5Gr-RL RA server REST API); and (ii) 5Gr-RL extensions supporting customizing network slices at the data-plane and the corresponding applications at the control and data plane of the forwarding infrastructure. This table also provides the planned deployments for Release 2, focusing on the 5Gr-RL. These include advanced algorithms and mechanisms for supporting closed-loop management for heterogenous use cases such as anomaly detection within the 5Gr-RL, as well as the investigation of more complex QoS policies compared to the baseline case (performance isolation) provided in R1. To do that, novel interactions, functions, information models, etc. will need to be designed and implemented at the 5Gr-RL with other elements in the 5Growth stack such as the AI/ML platform, the monitoring platform, etcetera.

TABLE 6: RELEASE PLAN FOR RA SERVER AND SUPPORTED CAPABILITES

Release 1	Release 2
5Gr-RL: • Support requesting expansion/abstraction computations via the new REST API with the RA server • Support of LLs bound to different Class of Services (e.g., Gold, Silver, Bronze) used for the CSA algorithm • For the InA algorithm, creation of the request among all the NfviPop pairs and deriving the LLs consumed by the 5Gr-SO • Support deployment of baseline QoS policy (performance isolation).	**5Gr-RL:** • Support for retrieving the model from the AIML platform to support anomaly detection (e.g., degraded WAN links) • Resolving affected WAN connections due to an anomaly detection decision • Support for deploying monitoring jobs of the underlying WAN infrastructure • Support of triggering asynchronous re-optimization mechanisms of the existing WAN connections • Interaction with AI/ML platform • Enable alarms on QoS policy from monitoring platform. • Support more complex QoS policies (e.g., data plane arbitration via programmable scheduling)
RA server: • Support processing and serving abstraction computations through new REST API with the core 5Gr-RL • CSA algorithm for computing (expanding) feasible WAN paths bound to a specific LL CoS • InA algorithm for computing WAN paths among pairs of NfviPops used to derive the abstraction of LLs towards the 5Gr-SO	**RA server:** • Support of an algorithm targeting global concurrent optimization of multiple WAN connections impacted by a detected network anomaly • Support of an algorithm addressing network resource re-optimization of a bulk of existing WAN connections demanded by the 5G-RL
• Support of the new designed 5Gr-RL and RA server REST API enabling bulk of expansion/abstraction computations • ONOS applications and P4 programs for programmable software targets to support the deployment of QoS policies that ensure network slice KPIs are met.	• Support of the 5Gr-RL and RA server REST API to cover restoration and re-optimization algorithms at the RA server: Extending the REST API (at both client and server sides) carrying information about WAN paths to be restored / re-optimized • Extend ONOS and P4 programs for programmable software targets to support the more complex QoS policies (e.g., data plane arbitration via programmable scheduling)
	5Gr-SO/5Gr-VS: • Non-Real Time Orchestration of next-generation RAN resources • Advanced vertical slice arbitration algorithms • Interaction with AI/ML platform

2.5. 5Growth CI/CD and containerization

CI/CD generally refers to the combined practices of Continuous Integration (CI) and either Continuous Delivery or Continuous Deployment (CD).

- **Continuous Integration**: CI is a practice responsible for processing developers code all the way from the repository commit till containerizing application, including building, testing, and accepting new code (merge) stages.
- **Continuous Delivery**: CD is a practice to deliver or deploy merged code as a part of a product in the environment.

5Growth CI/CD implementation goals are to enable all consortium developers to release a constant flow of software updates into testbed or demo environments to quicken release cycles, lower time costs, and reduce the risks associated with development. Also CD part is responsible for the environment's life cycle to bring testbed management onto a new level allowing to make changes in a software configuration promptly reflecting the requirements that appeared during the research and development process.

Release 1 implementation of the CI/CD and containerization innovation consist of the following components:

- Private 5Growth project repository to store components source code and configurations for CI/CD and containerization;
- 5TONIC environment with deployed OpenStack Rocky release;
- CI/CD server, Jenkins2 deployed on 5TONIC premises;
- Cloudify orchestrator for virtualized infrastructure management deployed on 5TONIC premises;

Figure 12 shows the full CI/CD pipeline from the moment of committing new code till deploying changes to the environment.

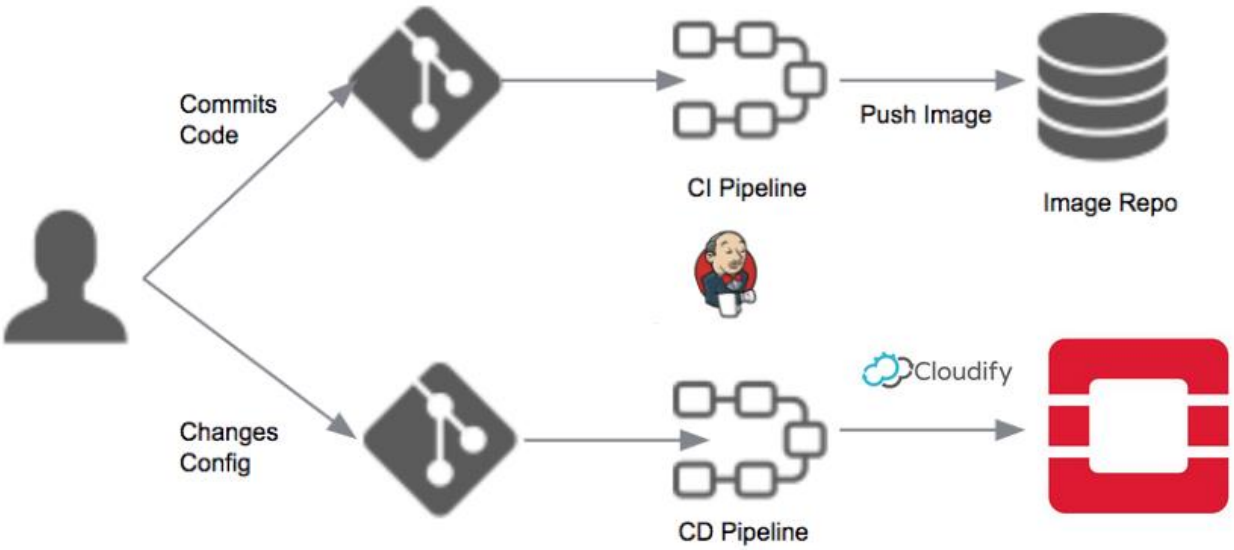

FIGURE 12: HIGH-LEVEL CI/CD ARCHITECTURE VIEW

In order to support current project requirements 5Growth CI/CD and containerization innovation currently provide the following functionality:

- Triggering by a developer's commit in private project repository (SCM polling);
- Creating a virtual environment on OpenStack using pre-created Cloudify blueprint;
- Cloning new changes from the repository, building 5growth components, containerizing components in a newly created environment;
- Testing a containerized application;
- Pushing Docker images to the private registry for future use in deployment in case of success or notifying via email in case of failure;
- Deployment 5Growth components on a new or previously instantiated environment.

Continuous Integration pipeline is explained in Figure 13:

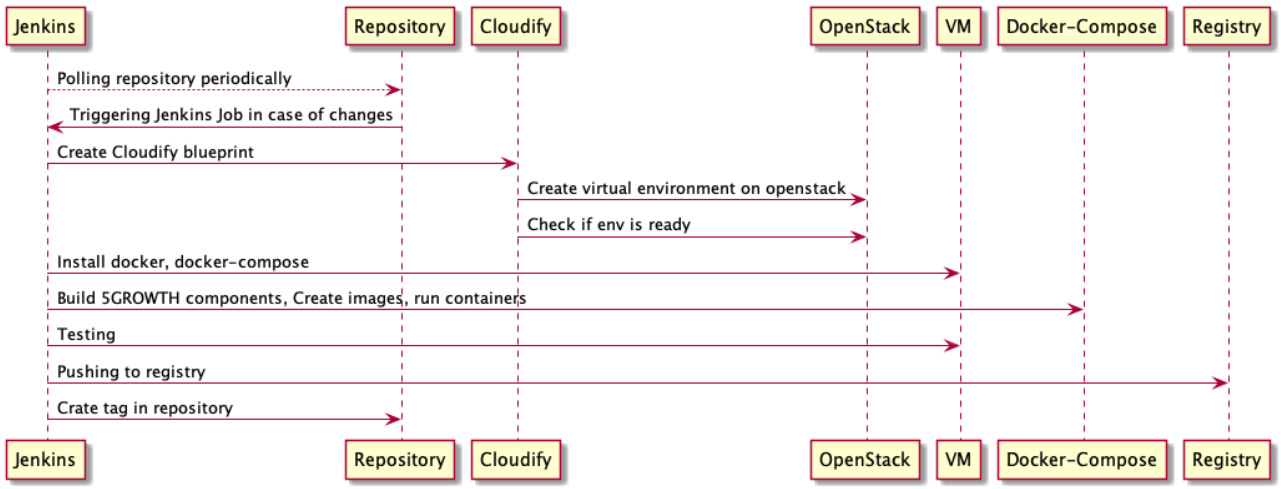

FIGURE 13: CI PIPELINE WORKFLOW

- Jenkins server monitors periodically each 5growth platform component repository (Polling function);
- Jenkins triggers a Job in case of any changes detected;
- Jenkins Job containerizes Cloudify CLI (Configuration required for connection to Cloudify manager and generated Cloudify blueprint mounts to containers);
- Containerized Cloudify CLI connects to Cloudify manager and creates new WM in 5TONIC OpenStack;
- Instantiated VM, once up and running, used by a next stage of the job to clone configuration files and 5Growth components source code from private 5Growth repositories;
- Jenkins Job installs all required software for containerization and building 5Growth components;
- Build 5Growth components as Docker containers.
- Built docker images with 5Growth components start as containers on the same VM;
- Once started, containers are ready for the tests, and once passed successfully, images are pushed to the registry, and can be used whenever needed.

Continuous Delivery pipeline once manually triggered, allows deployment of 5Growth components to the environment – existing 5TONIC VM or instantiate VM as part of the Continuous Delivery pipeline.

TABLE 7: RELEASE PLAN FOR CI/CD AND CONTAINERIZATION

Release 1	**Release 2**
Containerization: • 5Growth component containerization (Docker)	**Containerization:** • Containerization extension to Kubernetes deployment
Environment management: • Reworked LCM pipeline for environment management (5TONIC OpenStack cluster installation and removal)	**Environment management:** • Standardized developers environment (minikube on top of devstack) • Infrastructure as Code approach for describing environments (develop, testbed, demo) • Automation for Kubernetes deployment
QA: • Automated testing pipeline	**QA:** • Extend test coverage

3. References

[1] 5Growth. "Business Model Design." Deliverable D1.1, September 2019.

[2] 5G-TRANSFORMER, D1.3, 5G-TRANSFORMER refined architecture, May 2019.

[3] 5Growth "Initial Design of 5G End-to-End Service Platform." Deliverable D2.1, December 2019

[4] The source code of the 5Gr-VS is available as open source, under the Apache v2.0 license, on GitHub: https://github.com/5growth/5gr-vs

[5] The source code of the 5Gr-RL is available as open source, under the GPL license, on GitHub at https://github.com/5growth/5gr-rl

[6] The source code of the 5Gr-SO is available as open source, under the Apache v2.0 license, on GitHub: https://github.com/5growth/5gr-so

[7] 5Gr-VoMS GitHub repository available at the project software repository: https://github.com/5growth/5gr-mon

[8] Apache Spark. "Lightning-fast unified analytics engine for large-scale data processing", Available at: https://spark.apache.org/

[9] Apache Kafka, "A distributed streaming platform", Available at: A distributed streaming platform

[10] The interworking between 5Gr-RL core and the standalone RA server is based on an enhanced REST API available in https://github.com/5growth/5gr-rl/blob/master/5gr-rl-ra-server/API/5gr_rl_ra_api_v1.1.1.json

[11] The source code of the RA server is available as open source, under the Apache v2.0 license, on GitHub: https://github.com/5growth/5gr-rl/tree/master/5gr-rl-ra-server/src

[12] The source code of the ONOS OpenFlow Slicing application is available as open source, under the Apache v2.0 license, on GitHub: https://github.com/5growth/5gr-rl/tree/master/rl/plugins/WIM/ONOS-OpenFlow-Slicing

[13] The source code of the ONOS P4 Slicing application is available as open source, under the Apache v2.0 license, on GitHub: https://github.com/5growth/5gr-rl/tree/master/rl/plugins/WIM/ONOS-P4-Slicing

[14] Yen, Jin Y. "An algorithm for finding shortest routes from all source nodes to a given destination in general networks." Quarterly of Applied Mathematics 27.4 (1970): 526-530.

4. Annex 1

Herein we provide further details about the implementation of the CSA and InA algorithms deployed within the RA server.

CSA

The use of this algorithm has some implications over the interactions between both the 5Gr-SO and 5Gr-RL and the 5Gr-RL and WIM controllers. Basically, it is assumed that the 5Gr-RL, as a network provider, offers to the 5Gr-SO a pre-defined set of LLs bound to a specific CoS (e.g., Gold, Silver and Bronze). Every CoS has its own capabilities in terms of maximum supported bandwidth, end-to-end latency, etc. As an example, Gold CoS offers up to 100Mb/s connection flows tolerating a maximum end-to-end of 3 ms; Silver CoS supports flows demanding 50Mb/s without exceeding a maximum end-to-end latency of 5 ms; Bronze CoS supports 25Mb/s and 10 ms for maximum data rate and end-to-end latency, respectively. Those supported LLs and their CoS attributes are passed by the 5Gr-RL to the 5Gr-SO. It is important to mention that those LLs at the 5Gr-RL are not explicitly tied to a specific pre-computed WAN path. Additionally, observe that it is likely that not all the defined LL CoS types can be offered for all the NfviPop pairs.

Using the collected LL's CoS the 5G-SO triggers the PA computation. The output of this algorithm selects the set of NfviPops to allocate the required network slice and service as well as the LLs providing the connectivity among those NfviPops fulfilling the service requirements as shown in in Figure 14. For the LLs, those are passed to the 5Gr-RL core requesting the allocation of the resources (WAN paths) supporting such LLs.

For each received LL, the 5Gr-RL handles its WAN path computation ensuring the targeted requirements such as bandwidth (in b/s) and end-to-end maximum latency (in ms). For the sake of completeness, these requirements are individualized on per LL basis and contained in the same message as the requested LLs. As said the WAN path computation is delegated by the 5Gr-RL to the RA server. Prior to that, as reflected in the workflow shown in Figure 14, the 5Gr-RL updates its WANs view through the underlying WIM controllers. Next, the 5Gr-RL requests (for each LL) to the RA server the WAN path passing the endpoint LL's PEs, the requirements (i.e., bandwidth and maximum latency of the network service) and the updated view of the WANs infrastructure including inter-WAN links. Upon receiving the request, the RA server triggers the CSA algorithm.

CSA is based on a K-Constrained Shortest Path First (K-CSPF) mechanism based on a modified Yen algorithm (see [14]). That is, the CSA computes up to *K* shortest paths aiming at fulfilling the requirements. The selected and returned WAN path to the 5GR-RL core (out of the computed *K* paths) is the one providing to i) minimize the number of traversed links; ii) the end-to-end latency; and iii) foster efficient use of the available bandwidth, e.g. attaining an efficient use of the overall WAN available link bandwidth. The latter is attained through favoring the selection of the WAN path having the largest residual (available) bandwidth over the most congested path link.

The successfully computed WAN path is then used by the 5Gr-RL to conduct the resource allocation over that WAN path. To this end, the 5Gr-RL communicates with the involved WIMs (via the SBI) controlling the selected resources forming the WAN path.

As said before, the WAN path computation and allocation of the resources is done sequentially for all the LLs used for the network slice and service. Once all the LLs are expanded in their respective WAN paths, the 5Gr-RL notifies the 5Gr-SO about the successful network slice and service deployment.

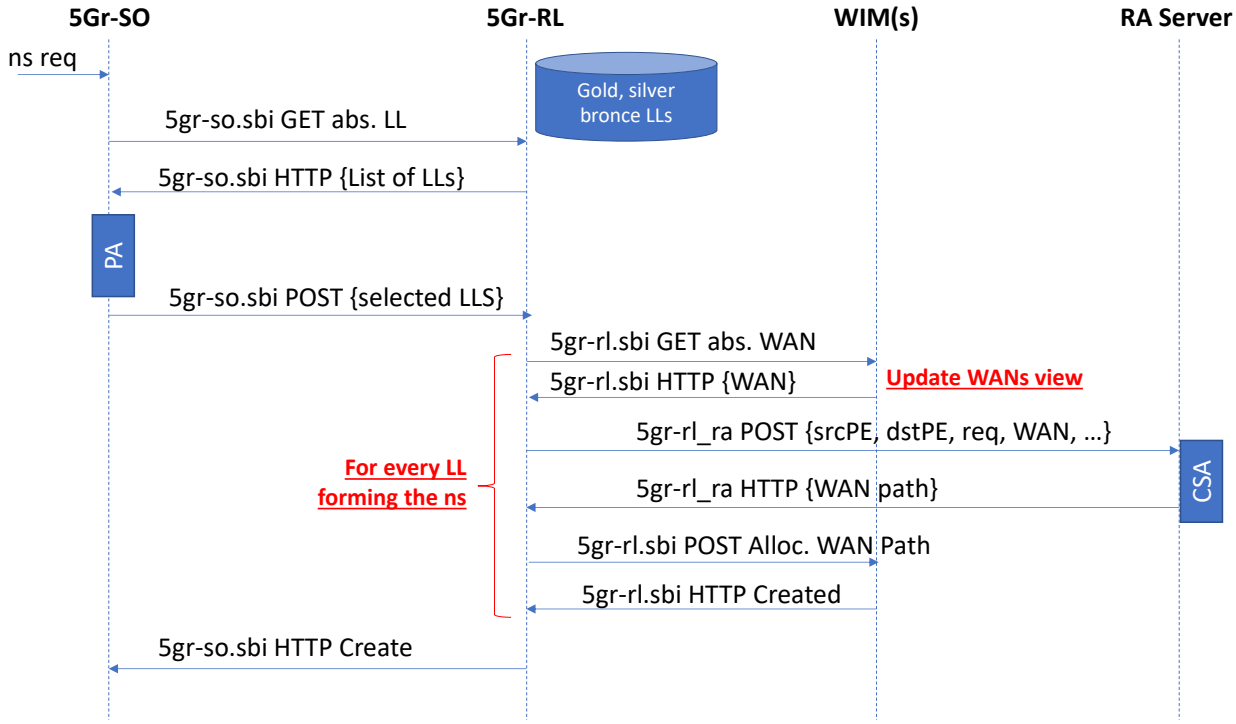

FIGURE 14: IMPLEMENTED WORKFLOW FOR RA SERVER USING CSA ALGORITHM

InA

The InA algorithm is devoted to providing the computation of the WAN paths that then are derived on an abstraction made by the 5Gr-RL to compose feasible LLs connecting each NfviPop pairs. The workflow for this approach is shown in Figure 15.

Upon receiving the instantiation request of a network slice or service, the 5Gr-SO asks to the 5Gr-RL for the set of the LLs. According to InA, the 5Gr-RL first retrieves an updated WANs view communicating with the underlying WIMs. The 5Gr-RL then determines all the NfviPop pairs along with their endpoint PEs. Using this, the 5Gr-RL commands the request of the WAN path computations to the RA server for interconnecting each pair of determined PE pairs. Specifically, the request carries the bulk of WAN computation requests one per each PE pair as well as the updated WANs information. Notice that no requirements (e.g., in terms of a specific bandwidth of maximum tolerated latency) are specified for each WAN path request. Indeed, the RA server is required to seek the best performance WAN path for each PE pair. The targeted best performance WAN path is attained by the InA mechanism computing the path with the lowest number of traversed links, lowest

end-to-end latency and having the largest residual bandwidth on the most congested path link. To this end, the InA algorithm is like the CSA algorithm (i.e., based on a K-CSPF) but no requirements need to be fulfilled. In InA, more than one WAN path for a given PE pair can be returned by the RA server to the 5Gr-RL. This allows the latter to derive more than one LL (with different capabilities and attributes) to the 5Gr-SO for a specific NfviPop pair. Thus, in this approach, the 5Gr-RL keeps a 1:1 mapping between a LL and its explicit physical WAN path.

The set of derived LLs is then passed to the 5Gr-SO which uses this information to satisfy the network slice and service requirements (i.e., selection of the NfviPop to allocate the VNFs and set of LLs providing the NfviPop connectivity). As in CSA, the set of selected LLs are then passed to the 5Gr-RL to complete the resource allocation. For each received LL, the 5Gr-RL finds the previously computed WAN path associated with that LL and triggers the resource allocation over such a WAN path via the corresponding WIM controller.

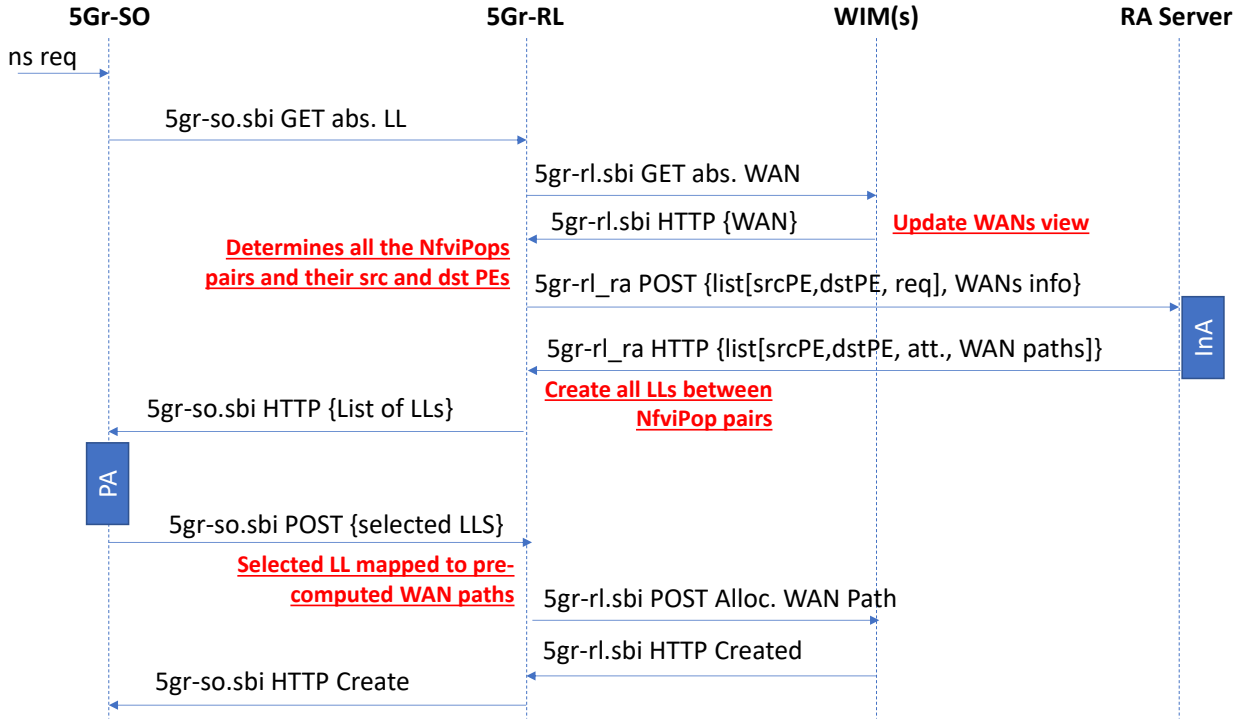

FIGURE 15: IMPLEMENTED WORKFLOW FOR RA SERVER USING INA ALGORITHM