ATHENA: An Intelligent Multi-x Cloud Native Network Operator

Alireza Mohammadi and Navid Nikaein

***Abstract*—This paper presents ATHENA, a novel design and a new generation of MANO/OAM that fully adheres to the cloud native principles, while fostering innovation and sustainable deployment for 4G, 5G, and beyond. It elicits an agile and intelligent, dynamic control over a variety of vendors and radio stacks (multi-x) coexisting on the same network with built-in observability and at the scale. With an intent-based, declarative, and distributed constitution, authentic to the cloud native pillars of isolation, scalability, and observability, we have established a scalable and efficient design and implemented its concrete proof-of-concept platform that is able to simplify the adaptation of cloud native for telecommunication. ATHENA automates both the semantics and synthetics of the lifecycle of telco workloads while attending to the performance and sustainability requirements. Accompanied by intensive evaluation on a concrete implementation, we show how several uses cases including private networking, Open RAN, and green computing would be facilitated and sustained with a low footprint and green management and operation. In particular, we improve the agility by 75% on Day-1 and 60% on Day-2 in comparison to OSM, while reducing over 93% overhead in Operation, 70% in Management, and 90% in Orchestration. ATHENA shows less than 2% performance loss for high throughput, with less than 50μs jitter. It demonstrates 99.9995% availability for immutable Day-2 upgrades and zero down-time for mutable reconfigurations. And for energy efficiency, we show improvements of maximum 17.4% per UE and 78.3% per gNB using the proposed decision-making framework.**

***Index Terms*—Mobile networks, Automation, Cloud Native, Service Management and Orchestration, Private Networking, Green**

I. INTRODUCTION

Automation is proven to be the key for any industry to grow in the scale and revenue; telecommunication is no exception, and by early releases of 5G standards, automation has been considered as a necessity to achieve the goals foreseen for the technology. Transformation to software implementations of Network Functions (NFs) paved the road to automation by means of Management and Operation, where solutions such as Open Source MANO (OSM) and Open Network Automation Platform (ONAP) were introduced to automate the lifecycle of NFs based on standards from European Telecommunications Standards Institute (ETSI) and Open Network Foundation (ONF) on Network Function Virtualization (NFV). After the release of OSM [14] and ONAP [23], the research community turned its emphasis to the higher level operations such as multi-clustering [20], service onboarding and management [11], or multi-domain orchestration [5], ignoring MANO itself with otherwise minor modifications to support Containerized NFs (CNFs). However, some works like [1] have recognized that mere inclusion of Kubernetes (K8s) as a virtual infrastructure manager or naively re-packaging Virtual Machines (VMs) as containers do not suffice to achieve the desired and promised functionalities of the cloud native. Furthermore, dismissing the differences between the containers and VMs, in terms of lifecycle, runtime behavior, and management has led to inefficient and ineffective solutions. On the other hand, owing to the rapid evolution of technology, research materials on the cloud native and MANO dating before 2015 (marking approximate date of Kubernetes becoming popular) are now either irrelevant, inadequate, or obsolete, yet being referred in the recent works such as [4, 7]. By itself, this has formed an inconsistent and inaccurate terminology and taxonomy where the misconceptions are escalated in the direction of the cloud native. The further the research community develops, the inconsistencies are more pronounced, often by irregular intakes of various generations of the cloud native principles and technologies, scattered over the time.

Approaching cloud native telco in 3GPP standardization of 5G Core Network (CN) has mainly led to Helm-based solutions such as OpenAirInterface (OAI) Helm charts[1] or Robin Smart Helm[2]. Even though the standard aims for dynamicity, the resulting designs and implementations are of static nature, even for simple matters such as IP address assignment and resolution. This ignorance impels these setups to rely on human operators while they continue to flounder at a larger scale. Such malpractices are often justified as minor engineering issues, but they are in fact a reflection of carrying over a common practice from either legacy Physical NFs (PNFs) or comparably outdated design architectures with little to none regard to the nature of the cloud native and cloud deployments.

In recent years, several new design principles and architecture were born alongside 5G or as a part of it, especially for the Radio Access Network (RAN). Open RAN, private networking, network slicing, and sustainable and green networking are a few noteworthy examples. Various communities have been formed around these topics, and O-RAN Alliance to date remains one of the most prominent ones, with active participation from major vendors and operators. The O-RAN Alliance iteration of MANO, revived as Operation and Maintenance (OAM) [22], builds upon the same disputable amalgamation of ETSI-NFV without addressing any of the issues discussed

Manuscript submitted on January 7th, 2023. (*Corresponding author: Alireza Mohammadi*)

A. Mohammadi and N. Nikaein are with the Communication Systems Department, EURECOM, 06904 Sophia Antipolis, France (email: alireza.mohammadi@eurecom.fr; navid.nikaein@eurecom.fr)

[1]https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed

[2]https://robin.io

in this paper. In fact, all the existing O-RAN stacks use either OSM or ONAP as their OAM, including but not limited to the O-RAN Software Community (OSC) version of the O-RAN SMO. In summary, all the efforts to date towards cloud native telco share the same flaws and shortcomings, which, in the authors' opinion, are mainly due to misinterpretation. The core concepts are often renamed and recycled, proving them to be valid recurring ideas, but they are not manifested properly in the modern context of cloud native telco.

ATHENA bases its foundation upon the Operator Framework [10] to assist or replace the human in the loop, approaching an intelligent, ultra-dynamic, and flexible automation, not just for the CN or edge applications but also for the RAN and its hardware resources. Since the management of the RAN is arguably the most challenging task of the OAM, in this paper we mostly focus on the examples and evaluations of the RAN. However, it should be noted that all the examples have the CN and edge components deployed and managed by ATHENA as well. Operator Pattern, as the foundation of the Operator Framework, dates back to 2018, when Declarative Operators (DOs) made their way as opposed to the imperative Event-Driven (ED) MANOs. The former method focuses on matching the desired and observed state via idempotent actions without extracting specifications of the trigger or maintaining a state machine [10], but the latter adds listeners to specific events and assumes the state is kept consistent between the operations. In distributed systems, however, a consistent and highly-available state machine is not a safe assumption [13], which has forced OSM and its descendants to use dedicated message brokers [14], whereas ATHENA solely utilizes K8s APIs to handle communication between its components *asynchronously*. When it comes to deployments with K8s, the former practice not only involves substantial overhead, but also means the pre-existing cloud native tools need to be reinvented, especially on lifecycle management and resource control. O-RAN OAM has the tendency to follow the same path and use the same implementations for reference. Despite the admiration for virtualized and software-defined RANs, the O-RAN OAM design is more in favor of PNFs [21]. The defined services and interfaces become less and less relevant and compelling as one moves from PNFs to VNFs to CNFs and finally cloud native RANs. This tendecy roots in the formation the O-RAN from Public RAN operators with unforeseen considerations for Private5G sector. Proven by demonstration of the use cases in this paper, ATHENA responds to this cavity by positioning itself mostly from the perspective of Private5G and its relevant use cases. This rather specialized focus helped us to maximize the efficiency of the design and implementation.

In summary, this paper makes the following contributions:

- A multi-vendor, -radio, -runtime, -cluster, or in short multi-x design as generalization of Open RAN;
- A novel, declarative MANO/OAM design and abstraction which adheres to cloud native principles, supports multi-x deployments, and enables micro and macro decision-making, applicable to the new use cases in 5G and beyond such as green MANO/OAM, improving automation and management of Open RAN in Private5G sector;
- An extensible multi-level Operator Plane that enables End-to-End complex operations such as multi-x cost optimization by separating network custom resources (CRs) from its deployment composition model and creating End-to-End logical networks and concepts by (recursively) exposing new set of CRs at each level effectively abstracting the resulted networks and the related concepts.

II. RELATED WORKS

Before comparing the state-of-the-art with ATHENA, one should recognize that a MANO/OAM needs to carry out tasks beyond mere installation or creation of the NFs with a one shot configuration. In that perspective, K8s and its distributions and extensions such as OpenShift[3] and Rancher[4] are not MANO or OAM, but a container orchestrator and indeed ATHENA could interact with any of them to orchestrate the pods. Moreover, an orchestrator is not concerned with the tasks expected from the MANOs and OAMs, such as optimizations specific to telco, semantic understanding of the network, or specific lifecycle management of the NFs. Following the same reasoning, Helm-based solutions such as OAI or Robin Helm charts are not proper examples of MANO/OAM but simply packaging of K8s YAML files. In this context, K8s is analogous to an Operating System (OS) and Helm is analogous to a package manager for it; none of them carry on the intended logic for OAM or MANO. We do not perform further comparisons with the mentioned as they seem irrelevant and unfair.

TABLE I: Taxanomy of the generations of MANO/OAM

G#	Solution	Workload	Management	Operation
1	OpenStack[5] Family	VNF	-	ED Triggers
2	Juju[6] Family	VNF/CNF	Sidecar Charms	ED Operator
2+	OSM-based MANO	PNF/VNF/CNF	External EM	ED Charms
3	Operators Frameworks	CNF	-	Application DO
4	ATHENA	CNF	Sidecar Manager	Logical DO

Workload Types: ATHENA orients itself around supporting CNFs that are packaged with the particular wireframe of ATHENA to (a) make it compatible with the cloud native models [16], (b) improve its lifecycle, and (c) to align itself most-efficiently with Private5G use cases where CNFs would be the dominant workload type. By preserving the support for VMs, solutions like OSM [14] are condemned to lag behind the advanced CNF capabilities or otherwise make the platform heavy or over-complicated. On the other hand, O-RAN's specifications define inapplicable interfaces and services for CNFs which are rather useful for PNFs. Examples of such services are those concerned with changes in the network interfaces. In common cases, the containers do not have the permissions to modify the network interfaces, and they are indeed configured externally via the network plugins in the orchestrator. This example also shows how based on the workload type, the scope and the formation of the MANO/OAM evolves. To lower the complexity of the architecture and to propose a truly cloud-native solution, ATHENA focuses only on the CNFs

[3]https://www.redhat.com/en/technologies/cloud-computing/openshift
[4]https://rancher.com/
[5]https://www.openstack.org/
[6]https://jaas.ai/

and leaves behind the PNFs and VNFs. This should give the platform the desired momentum to skip VMs, which are deemed superfluous given the superior performance and lower resource usage of container workloads compared to VMs. Furthermore, ATHENA could be used by network operators with little or no infrastructure in place who could benefit from a fully CNF-based environment without an interim VNF adaption step, significantly shortening the time to harvest benefits of the deployment. This pattern is very common in the Private5G sector due to its infancy.

Operation: During our literature study, we have identified four MANO/OAM generations presented in the Tab. I, grouping related solutions under a single generation. ATHENA's conceptualization of the Operators differs with the two well-known Operator patterns, i.e., Redhat Operator Framework[7] and Canonical Charmed Operators[8]. The aforementioned patterns are intended to be generic, allowing for onboarding of any application. In their design pattern, usually each application from each vendor is associated with its own Operator. However, by relying on the standards, rather than encouraging each application vendor to develop its own Operator, we provide Operators for logical entities that are formulating concepts such as network terminals, functions, or slices. In this view, the NFs could be mix-and-matched which was otherwise impossible or troublesome to achieve since each vendor was providing its own (perhaps patched or tweaked version) MANO/OAM. Solutions like Kube5G [3] lack the same insight, and are grouped with the Operator Frameworks as another application Operator. Unlike works such as [18] that are supposed to gain insights and take semantic decisions considering RAN or CN for specific scenarios, ATHENA is more generic and concerns with synthetics and structure of the network rather than what happens in a particular case. ATHENA delivers a built-in autonomous observability stack as opposed to the traditional monitoring systems which required human intervention. The processing in those systems is done by an ED which would take centralized and imperative actions. However, ATHENA takes the approach of intent-based and decentralized decision-making, utilizing sidecar Managers to realize a declarative idempotent update on the cluster.

The Operator Framework and inherently ATHENA follow up the fundamental control logic principles from Kubernetes [9]. In short, this class of the control logics for the Operators relies on *level-based* designs rather than the *edge-triggered* ones that are common with the EDs. On the other hand, these control systems assume *Open World* conditions, meaning they acknowledge the presence of uncertainty, variability, and unknown factors that can affect the system's behavior. The declarative nature of the Operators in ATHENA are manifestations of these principles. What in particular ATHENA offers on top of that is to adaption and porting of the mentioned designs to the specific field of telco by introducing the concept of the Operator Plane. This plane allows a structural and semantic extension of ATHENA for unforseen use cases and scenarios.

Management: ATHENA Manager is onboarded in each pod as a sidecar container, following the paradigms of sidecar containers in K8s [16] and ETSI-NFV Element Manager (EM) [14]. However, the sidecar pattern in K8s [16] is barely used for management purposes in terms of controlling the lifecycle of an internal application and ETSI EM [14] performs arbitrary operations, some of which are out of the scope of our Manager (e.g., billing), or they are related to VM environments (e.g., installation commands). O-RAN OSM has recently adapted the sidecar pattern for synchronization services [2]. While they have recognized the values of the pattern, it seems its application remains limited to the synchronization. ATHENA Manager performs agile sub-lifecycle operations, observability proxying, configuration management, and dependency resolution. It should be noted that unlike network proxy sidecars, the sidecar model for the Manager in ATHENA is not abolished by the advent and adaptation of the extended Berkeley Packet Filter (eBPF) and Xpress Data Plane (XDP) [25] technologies. However, if the functionalities of the Manager are implemented otherwise, it could be discarded for a particular NF. ATHENA remains agnostic to the way the Manager functionalities are provided, and indeed, the Manager is intended to aid rather traditional and non-cloud native applications to be aligned with the cloud native paradigms.

III. OVERVIEW

ATHENA is formed around the idea of **multi-x** as its motivation, whereby "x" stands for vendor and radio in telco context and container runtime, OS, or cloud provider in the cloud context. ATHENA embeds support for simultaneous deployment of workloads from different vendors relying on the different radio devices (VI-B), while supporting multi-node, multi-cluster (by BGP networking), multi-tenant (by K8s namespacing[10]), and multi-runtime deployments on top of K8s. We have foreseen multi-x as the natural generalization of Open RAN which seeks beyond RAN to CN and MANO/OAM itself. Thus, we believe ATHENA's multi-x would enjoy the same positive expectations of Open RAN as briefed, for example in [9, 28]. In the Fig. 1, we have demonstrated the variety of vendors as the most important dimension of multi-x by different shapes. What comes next in this paper implicitly assumes multi-x support by design. This also includes interaction with external elements, not directly under the control of ATHENA (marked by the "External" label in Fig. 1).

In accordance with the multiplayer ecosystem of telco, we have separated the concerns on the network parameter, including radio, identity, and slicing with the composition of the network and how it scales. In this way, the vendors and integrators could evolve and upgrade their software, independent of the network operators. The networks are **declaratively** defined as a Network Custom Resource (CR) in K8s[11], whereas the composition of such networks are described in Composition Model CRs that abstract the cloud-related parameters, such as

[7]https://operatorframework.io/what

[8]hthttps://ubuntu.com/engage/collection-of-charmed-operators-whitepaper

[9]https://github.com/kubernetes/design-proposals-archive/blob/main/architecture/principles.md

[10]https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

[11]https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/

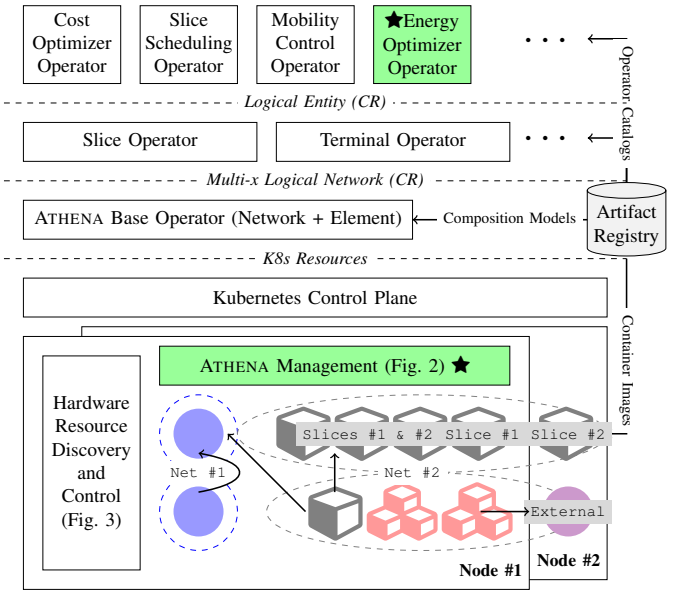


Fig. 1: ATHENA overview in a nutshell

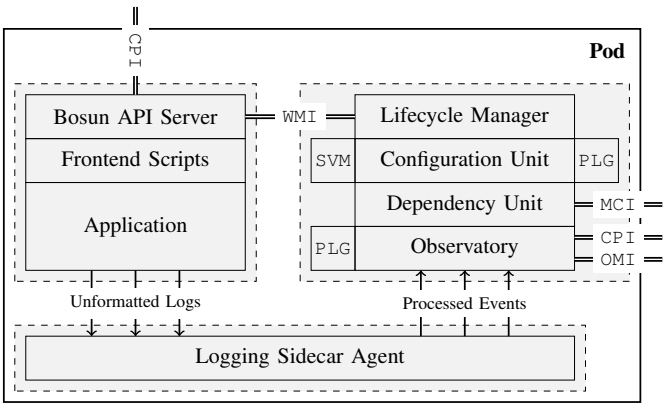


Fig. 2: Sidecar Manager and Agents in ATHENA

images, resources, and networking as well as the configuration of the workloads. The Composition Model CRs also define the observations that need to be collected from the Workloads as well as the scaling policies for the NFs. This separation distinguishes between *synthetics* and *semantics* of (multi-x) logical networks. The adaptation of simple yet extensible Composition Models would enable the vendors and the service operators to independently innovate on top of ATHENA. These CRs are defined using CR Definitions (CRDs) that are registered in the K8s API server. CRDs in K8s are based on the OpenAPI v3 specification with some extensions. In that sense, one could imagine defining a CRD is the next generation of defining a REST API using OpenAPI, which is the common approach in the O-RAN's North-Bound Interfaces (NBIs). In the former case, K8s takes care of a mature and well-tested implementation of the API, with all the features expected of a modern API, such as verification, authentication, and authorization.

According to Fig. 1, ATHENA transforms presentations of the network resources, network functions, and containers from a traditional K8s resource such as pod or service to a logical network, through a Base Operator capable of controlling and abstracting the networks and their elements. The abstracted format encapsulates the properties of concern for the network operators, tailored to the telco terminology and use cases, without the complexity of the underlying infrastructure.

One of the core contributions in ATHENA is the concept of the Operator Plane where several Operators cooperatively process logical presentations and entities of the network in various aspects. The functionalities of ATHENA in the Operator Plane could be expanded in two levels. Level-1 of the Operators consume logical network CRs to expose a new set of CRs targeting a logical entity such as slice or network terminal. The level-2 is built on top of those logical entities, to enable sophisticated and perhaps E2E Operations, like cost optimization. Through the Operator Plane, concepts and functionalities are transcended from physical to logical, abstracting both the network itself and the related concepts. To illustrate the facility of development and innovation on top of this novel design model, we have already implemented the slicing functionalities and network terminal Operators via the same framework while also providing examples for the upmost layer of the Operators: Cost Optimizer (VI-A) and Energy Optimizer (VI-C).

ATHENA Management Plane works as an extension of its Operator Plane, but in a distributed manner to take local, short-term decisions rather than global, longer-term ones. The former supports macro-decisions, whereas the latter defines micro-decisions, as presented in IV-B. A distinguishing example of these oppositions is captured in the green capabilities of ATHENA, marked with a star in the Fig. 1. The Energy Optimizer Operator would take decisions concerning day-long Operations using AI/ML algorithms, whereas the Manager supports micro-decisions to control a fine-grained, semantic lifecycle of the NFs for energy savings. The energy saving capabilities are integrated to the ATHENA's design, perfectly fitting in the distinction of the Manager and Operator. In the definition of the interfaces between the Manager and Operator or the Manager and the Workloads, we define open interfaces with simple, yet strict protocols that are less prone to protocol ossification [12] while being modernized and cloud native, preferring *de facto* standards over the *post facto* practices.

The RAN Intelligent Controller (RIC) platform could particpate in the Operator Plane to connect the Control Plane to the Operator Plane. In result the RIC could request and observe xApps or rApps using the CRs provided by the Base Operator. Beyond that, the logics of the other Operators could be influenced and extended jointly by the RIC. For example, the Energy Optimizer Operator would not only consider the RAN as a mere CNF, but it could also trigger control loops in the RIC for fine-grained optimizations. In this situation, from the RIC perspective, the Energy Optimizer Operator behaves as an rApp.

ATHENA unifies the device management for telco devices in a cloud native manner. What makes this device management particularly outstanding is its level of automation and consecutive isolation that it brings to the access network workloads. By the mindset of VMs, one might wrongly associate isolation

with lower performance, but we have proven this to be fallacious using our concrete prototype (V-B). The abstraction in the device management is based on the device capabilities to make pools of homogeneous devices that are distinguished by their geographical locations. Section IV-C digs deeper into the device management and its implications.

IV. Design and Implementation

We section ATHENA's design into four distinct planes that impact different aspects of the workloads' lifecycle, namely the Operator Plane, Orchestration Plane, Management Plane, and Functional Plane. This concept of plane separation might be viewed as an extension of the Control and User Plane Separation (CUPS) utilized in 4G/5G networks [19] into MANO/OAM. The Functional Plane itself is decomposed into the Control Plane and the Data Plane, according to the CUPS.

A. Operator Plane

Operator Plane in ATHENA enables customized Operations on the logical networks or the logical entities within it. Each Operator is composed of a few CRs and the Custom Controllers (CCs) which upon invocation by the requests from the orchestrator would *reconcile* the corresponding CRs. On this Plane, with a few fundamental Operators in action, one could build arbitrary control loops that interact with the networks established by ATHENA.

1) Base Operator: As the first level of abstraction, the Base Operator transforms K8s objects into expressive logical presentations of networks and their associations with each other and the cluster, linked by the provided Composition Models. This Operator hence could deploy and manage end-to-end 4G/5G networks that are requested via declarative intents in the form of CRs. The base Operator includes two CCs: (1) A Network CC which governs the collection of Elements forming a logical access, core, or edge network as well as how slices are associated with them. (2) An Element CC which controls individual NFs and their related components such as Kubernetes services, configurations, and pods;

Network CC: ATHENA's Network CC consumes a Network CR that contains the description of the slices as well as the list of access, core, and edge networks, possibly multiple instances of each. In the access networks definition, one also specifies the radio parameters including the device and signal features and the cell parameters including the center frequency, the band number, subcarrier spacing and the bandwidth. Upon invocation, the CC looks up for the Composition Models of each network. Based on the obtained information, the CC builds a topology of the network which later is used by the Management Plane to resolve the dependencies in a distributed manner. This topology is scoped and merged with the any provided custom DNS records to form a localized view of the network for each NF and supports Multi-Access Edge Computing (MEC) applications. Finally, the Network CC issues Element CRs to be later picked up by the Element CC that works in parallel. For the observability, the Network CC aggregates the status of all of its wrapping Elements to indicate the status of the network itself.

Element CC: The Element CC builds the corresponding pods and Kubernetes services with the proper configuration. During this process, connects the pods to the proper container scheduling parameters that are defined for the Element. It would also continuously probe the pods for custom Conditions that are provided by the Composition Models. These custom Conditions use the Kubernetes Probe interface and might be used to determine the readiness of the pod, if marked in the Composition Model. One particular example is used in the Terminal Operator to associate the readiness of the pod with its ability to reach the Internet from its radio interface. The Element CC also records and aggregates the Conditions exported by the pods for the observability.

2) Terminal Operator: To handle the network terminals and achieve an E2E control loop that also contains the UE, we have introduced a Terminal Operator. It is capable of handling four particular modes of the UEs: (1) The simulator mode where deploys a simulated 4G/5G UE to connect to the corresponding simulated eNB/gNBs, regardless of the level of the simulation (RF, L2, S1/NG, etc.); (2) The external mode where it is just a presentation of a handset outside the cluster and no container would be deployed in this mode; (3) The internal mode where a container is deployed attached to a physical UE module on the cluster; (4) The backhaul mode to support external network formations that shall use this terminal as a backhaul. The identity of the UE is automatically injected into the databases of the corresponding core networks, given their identification.

B. Pod Design and Management Plane

The pod in the proposed sidecar design depicted in the Fig. 2 is composed of up to four containers, two of which are mandatory and two are optional. (1) The Workload container that has the application with some frontend helper scripts that are application dependent with a generic API server that implements the Workload Manager Interface (WMI); (2) The Manager sidecar; (3) An optional logging agent that is responsible for collecting the logs from the application and forwarding processed events to the Observatory in the Manager; (4) An optional test suite container not shown in the figure that provides some testing utilities such as ping, traceroute, or iperf. Bosun is the name of the default implementation of the WMI server in the Workload container. The Workload and Manager containers share a common volume where the Manager would place the configuration files for the application to consume.

ATHENA's Manager, depicted in the Fig. 2, is the distributed part of the MANO/OAM with several new interfaces required to facilitate Day-2 operations and observability. Having such a sidecar design is important for (1) **Decoupling** the Management Plane from the functional plane, which is a common practice in the cloud native; (2) **Accelerate** adaptation to cloud native for the rather traditional NFs; (3) Keeping the workloads **lightweight** and **portable** to any environment by keeping the cloud-specific interfaces independent of the Functional Plane. The Manager is chiefly responsible for performing seven tasks: (1) Dependency resolution through its Management and Composition Interface (MCI); (2) Generate the configuration

files for the Workload and its applications by calling the right configurator plugins (PLG); (3) Lifecycle control (init, start, status, stop, etc.) for the corresponding application in the Workload container via the WMI in the form of an infinite observe-decide-act loop; (4) Consume the events from the logging agent and Condition plugins (PLG) and expose the overall and detailed status of the pod for probing via Container Probing Interface (CPI); (5) Gather, group, and pre-process metrics defined for the workload and expose them to metric collectors such as Prometheus[12] using Open Metrics Interface (OMI); (6) Listen for Linux filesystem notifications via Shared Volume Manager (SVM) and call for reconfiguration and restart if necessary; (7) Performing micro-decisions including, but not limited to, short-term green optimizations via WMI. The Lifecycle Manager is in an infinite *observe-decide-act* loop that is responsible for the lifecycle of the application inside the Workload container. Evident by the results in the Sec. V-A, this method has the maximum agility for managing the lifecycle of the application.

Dependency Unit: The Dependency Unit is responsible for the dependency resolution and fault management through the MCI, which exposes a simple REST API with at least two endpoints called `resolve` and `depends` on managing the weak and strong dependencies and fault tolerance. This interface is actively used by the Managers of the different Elements to solve the dependency and fault tolerance problems in a distributed and decentralized manner. A weak dependency is a mere dependence on the IP address or other parameters of the pod from another element, used to fill in the details in the configuration. On the other hand, a strong dependency means the dependant needs the dependee to be on ready state, so the call is not returned successfully until the managed is running and healthy. The resolution and dependency endpoints could optionally be called for a certain interface or service, either 3GPP or non-3GPP (like O-RAN) to form a finer grain dependency. The *dependant* would regularly call the `resolve` or `depends` endpoints to check if the dependency information is still valid. A readiness in the MCI context means the application was running healthy and the conditions were met for a *continuous* period of time, T, without any interruptions. If the period of calls from the dependant to the dependee is shorter than T, the dependant would always be notified of a change in the readiness status of the dependee, by having at least one call to the `resolve` or `depends` that returns unsuccessfully. Hence, the dependee would be also restarted, and the chain of dependencies would be re-evaluated likewise, as long as there are no loops in the dependency graph. The default reaction to a failed dependency might be changed by a policy defined in the annotations in the Composition Model.

Implementation Aspects: We rely on K8s services for making the communication at the MCI level, hence the Managers would call each other by their corresponding DNS names that are already filtered and scoped on the Operator Plane. In this way, one could enforce various policies on the communication between the Managers using service meshes in K8s. However, for data layer connections, the delay of K8s services is intolerable, managing multiple interfaces with K8s services is challenging, and even some protocols like SCTP are not reliable with every K8s networking stack or service mesh. We call this type of rather placeholder K8s services a Shadow Service as it is not exactly used like a normal K8s service. The Shadow Services are not used for Functional Plane communication, but only Management and Operations. K8s services are limited to two-level DNS subdomains, one for the service name and the other for the namespace name, which turns out to be insufficient for ATHENA. Therefore, ATHENA's Base Operator delivers a basic custom DNS server that supports meaningful subdomains for telco in the form of Pod name, Element name and Network name alongside the namespace name. These records are of the DNS type `CNAME` pointing to the Shadow Services. The configurations for the Manager and the Workload contain the rather meaningful DNS names that are resolved to the Shadow Services by the Base Operator's DNS server, then to the IP addresses of the K8s services by the K8s DNS server, and finally to the IP addresses of the corresponding pods by the MCI. We populate the configurations with the corresponding pod addresses that are resolved via MCI and bypass the shadow services. Still, for the sake of consistency, observability, and applying the network policies properly, the Shadow Services include all the Data Plane ports too. In case of a failure that would cause a pod recreation or any event such as eviction and preemption that results a new pod IP address, the `resolve` and `depends` endpoints would return different addresses compared to their previous calls. This would trigger a reconfiguration, perhaps followed by a restart, in the *dependant* pod(s).

Manager Interfaces: The Manager could be extended via two sets of plugins (PLG). The conditioner plugins implement `probe` and `metrics` function where given the parameters, they generate a related Condition object (exposed via CPI) or metrics (exposed via OMI) respectively. These plugins are used to transform the traditional monitoring information to the cloud native equivalents. The plugins are fed with the observation queries that define a source and a drain for the information with a certain data format. The plugins are responsible for the data collection and conversion, while the Observatory would periodically call them to get the latest metrics and conditions for feeding back to the CPI or OMI. The configurator plugins implement two functions: (1) `configure` is called via the corresponding input files to generate the configuration for the applications in Day-1; (2) `reconfigure` is called upon detection of changes in the input ConfigMaps in Day-2. It should update the configuration files and report back if it would be necessary to restart the application via the WMI. The SVM uses Linux *inotify* to detect changes introduced in the filesystem. The `reconfigure` is the first mile towards the genuine **Service Continuity** in 4G/5G networks by enabling Day-2 actions that require no restarting of the network using specific plugins developed for each vendor. However, still restarting the application should take a considerably shorter time than restarting the container by avoiding the initialization for the containers (V-A). The plugins could assume that all the base files for compiling the configuration are already made

[12]https://prometheus.io

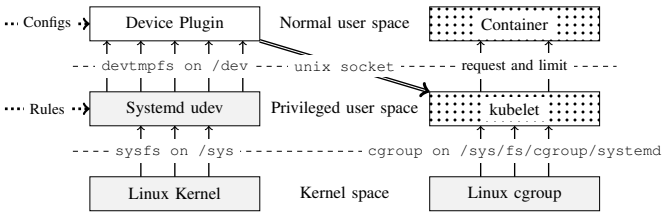


Fig. 3: Device management in ATHENA

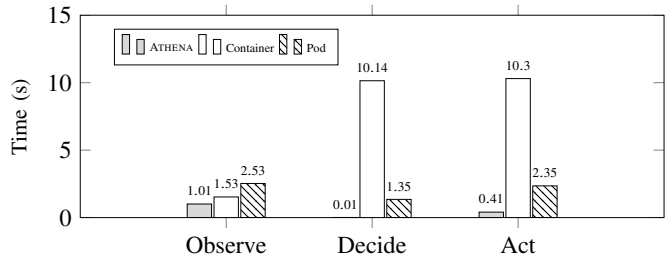


Fig. 4: Comparison of observe-detect-act cycles in ATHENA and vanilla Kubernetes.

available to them by the Configuration Unit in the Manager, regardless of the source of the files. The Configuration Unit would call the plugins separately for each configuration index defined in the Composition Model and passed to the Manager. The Base Operator makes sure that the Manager has all the necessary mount points for the Configurations Unit to function properly. The Manager is shipped with a default set of plugins that are generic enough to be sufficient for most of the use cases.

Manager Extensions: As opposed to the global, longer term decisions of the Operators, ATHENA Manager enables execution of **micro-decisions**. A micro-decision represents a decision that is concluded and intended for a short period of lifecycle and could be invalidated and overwritten by another decision in a short time. These micro-decisions could be used to perform short-time optimizations, in particular significant in green computing [24, 26]. To elaborate on an example of micro-decisions, we have considered an extension to the WMI with two new API calls. A `freeze` endpoint in the WMI intends to freeze the corresponding workload. By default, it is pointing to the `stop` endpoint internally, though each vendor could provide customized actions that would not necessarily stop the instance, but just put it in the sleep or freeze mode. Likewise, the `thaw` endpoint implements the awakening action and by default is pointing to `start`. Applications of these endpoints are explored in the Sec. VI-C.

C. Orchestration Extensions

ATHENA relies on K8s Control Plane as its Orchestration Plane, while introducing specific extensions to fully automate the network lifecycle operations. This also includes automatic device detection and management for a wide class of telco-related devices such as the radio modules, accelerators, and network terminals (essential for the E2E automation).

1) Device Management: ATHENA defines addressable and assignable resources in a form of device resources such as radio devices (whether hot-plug, or network-based, or PCI-based) or accelerators (e.g., GPU, FPGA). These resources are exposed as Kubernetes node resources via a device plugin and could be allocated to any of the pods upon request. Defining a device plugin improves the isolation and the security of the containers by following the Kubernetes security structure, and on the other hand opens up a whole spectrum of possibilities for the logics to be implemented in the Operator Plane.

Implementation Details: Automated device management in ATHENA relies on *systemd* and K8s device plugins, see Fig. 3. Systemd is the most-commonly used init daemon across all the Linux distributions which comes with a device management mechanism called *udev*. Given specific rules to udev, we govern the naming, management, and initialization of the devices on the nodes to achieve a harmonized representation of the resources. Later the ATHENA device plugin automatically and dynamically detects the devices and advertises them as node resources in K8s.

The physical devices in ATHENA are transformed to logical software devices by Linux systemd and udev in the smallest operational units for each device. These units are categorized by their capabilities into homogeneous sets of devices. The compatability of the devices with the NFs is left for the vendors to decide via the Composition Models. Similar to the NUMA topology, we define neighborhood metadata for the devices that are used in the container scheduling. The neighborhoods could refer to the physical connections between the devices, such as belonging to the same physical device, sharing the same PCI bus, and having the same RF ports and antennas. They could also refer to the geographical proximity or synchronization.

2) Probing and Observability: In telco workloads, readiness is a multistage event which might not necessarily be expressible as a binary readiness status. The readiness could rely on various internal state changes that are reacted by the other elements as a dependency. For example, in the O-RAN context, the RAN needs some basic discovery of the RIC before finishing its initialization, wihtout the need for the RIC to be fully operational. Exposing different readiness levels not only improves the observability but helps the Management Plane to handle complicated, multistage dependency matrices. To implement this we have overloaded the ReadinessGate feature of K8s [8] with relevant custom probes that complements the CPI interface defined in Sec. IV-B.

D. FCAPS in Athena

In ATHENA we have refactored cloud native principles in the design of FCAPS.

1) Fault Management: The faults in ATHENA are grouped into the direct and indirect faults. Direct faults are when the application itself crashes or fails to start. The direct faults are detected and resolved using the WMI and MCI as discussed in Sec. IV-B. Probes and CPI would ascend the visibility of the faults to the Operator Plane. If the fault is rooted in the Managers themselves, the Operator Plane takes the responsibility to resolve the fault, and if none of the methods are effective, the container orchestrator would aggressively

restart the corresponding pods. The time to detect and resolve the direct faults are studied in the Sec. V-A. The indirect faults are the cause for service degradation and disruption. They appear in the logs or the metrics of the workloads and are left for the Operators in the Operator Plane to handle. ATHENA does not directly react to the indirect faults, but it provides the means for the other Operators in the Operator Plane to do so via its observability stack.

2) Configuration Management: The configurations in ATHENA emerge from the CRs in the Operator Plane and are then transported using the Orchestration Plane to the Management Plane where the plugins do the last mile configuration. ATHENA focuses on a unified NBI for the configuration rather than enforcing the workloads to implement a specific interface. The proposed unified NBI is neutral in the sense that it allows to support customized or standardized configuration interfaces such as O-RAN O1. However, having a sidecar with a shared volume annihilates the requirements for a remote configuration interface such as O1, while remaining generic and flexible. Also, by definition, the configurator plugins tie the lifecycle of the Workloads to the reconfiguration at Day-2.

3) Accounting, Performance, and Security Management: Accounting, Performance, and Security Management in ATHENA are built upon its observability stack. The metrics in ATHENA are grouped into the *outbox* and *inbox* metrics. The inbox metrics are collected from inside the boundaries of the application and differ from one NF to another. Example of inbox metrics are those that are mainly collected and processed by the controllers such as RICs. The outbox metrics are collected from outside the boundaries of the application and are common for all the applications, such as the CPU and memory usage. Some of the outbox metrics like energy or cost are composite, second-order metrics that could be derived from several other inbox or outbox metrics. Methods of performance or accounting management that like O-RAN O1-PM fully or partially delegate the task of the measurement to the NFs themselves are potentially biased, inaccurate, and prone to the side effects of the measurement itself. Moreover, operating legacy NETCONF-based interfaces at scale becomes inefficient and challenging in the medium to large scale cloud-native applications, due to limitations of its transport protocol and improper data models. ATHENA does not directly involve itself in key management, authentication, or authorization. For most cases, the orchestration plane, K8S in this context, provides the means to handle these issues. However, ATHENA's device management and secured, isolated, and rootless containers are the cornerstones of a secure network. Otherwise, any security barriers could be breached by a faulty or malicious workload.

E. Day-2 Operations

ATHENA supports a variety of Day-2 Operations including auto-healing, reconfiguration, and upgrade. We address each of them in a separate subsection.

1) Auto-healing and Idempotency: The declarative design of ATHENA avails the Base Operator of auto-healing properties, i.e., whenever an Element or pod is deleted or evicted, it would reconsider the topology and adjust the associated parameters. The healing in action is merely repeating the deployment, since the control loops of ATHENA are designed to be **idempotent**. Thus, the effect of applying them repeatedly or under failure should result in the exact desired state.

2) Reconfiguration, Upgrade, and Immutability: Each CR has mutable and immutable constructs. Changes in the structure of the containers, such as their image or resources as well as changes in the radio device or identity of the networks, would result to recreation to preserve the immutability of the setup. Keeping the building blocks immutable is crucial to make scaled instances consistent and predictable [6]. We call such actions an **upgrade**, and ATHENA minimizes the down-time of the NFs during an upgrade as proven by example in VI-B. The agility of ATHENA and containers combined is the key to achieving the low down-time. However, a **reconfiguration** action, triggered by changes in mutable constructs, is handled by the distributed processing of the Management Plane with zero down-time. An example of changing the network topology and its effect on the service quality is given in VI-B. Reconfiguration in day-2 is of paramount importance in telco to achieve service continuity and flexibility at the same time.

V. EVALUATIONS

We have completely implemented the mentioned design of ATHENA at all layers. The code is mostly written in Golang with over 10k lines of code for the Operators in total, 3k lines of code for the Manager and its plugins, and another 3k for the extensions to orchestrator. For the evaluation of ATHENA, we have used this implementation and onboarded Amarisoft (AMR), OAI, and Software Radio Systems (SRS) already as vendors. The cluster under the test contains 9 machines with Redhat Enterprise Linux 8, CentOS 8, Ubuntu 20.04, or Ubuntu 18.04 installed on them. The radio devices supported on the cluster are USRP B210, USRP N300, AW2S RRH, AMR SDR50, and AMR SDR100.

A. Lifecycle Improvements and Agility

To simulate the effects of a failure and analyze how the *observe-decide-act* loop of the Manager would behave, we have onboarded OAI gNB workload on ATHENA then performed the following experiments: (1) Stop the RAN process inside the container to simulate a workload failure where in ATHENA, the Manager would detect the issue and act accordingly by restarting the application process; (2) Stop the main process of the container to simulate a full-scale container failure where Kubernetes intervenes to detect the error and then goes through crash loop back-off that takes considerable amount of time for recovery; (3) Finally we stop the pod's sandbox container to simulate an overall pod failure that causes all the containers in the pod to restart. As reflected in the Fig. 4, ATHENA's approach is considerably faster on each part. Besides, the main process of the container is chosen carefully to be the stable Bosun API server for WMI (Fig. 2) that exhibits a very low chance of failure, hence no transition to the longer recovery cycles of K8s. It should be noted that

TABLE II: ATHENA and OSM Timeline

Phase	Delay in OSM MANO Gen. #2+	Delay in ATHENA MANO Gen. #4	Improv.
Operator to Pod Creation	4 second	< 1 second	**75%**
Management Tasks	13 seconds	3 seconds	**77%**
Status Propagation	< 1 second	< 1 second	-
Day-2 Operations	5 seconds	2 seconds	**60%**
Deletion and Clean Up	52 seconds	5 seconds	**90%**

TABLE III: Overhead comparison

Plane	Resource	ATHENA REQUEST	ATHENA LIMIT	Charmed OSM	Improvement
Operator	CPU	10m	500m	151m	**93.38%**
	Memory	64MiB	128MiB	1.8GiB	**96.45%**
	Image Size	-	61.3MB	719.97MB	**91.49%**
Management	CPU	10m	30m	135m	**92.59%**
	Memory	15MiB	64MiB	50MiB	**70%**
	Image Size	-	145MB	183.52MB	**20.99%**
Orchestration	CPU	10m	50m	100m	**90%**
	Memory	30MiB	64MiB	2.67GiB	**98.88%**
	Image Size	-	21MB	694.30MB	**98.88%**

time for observation is composed of the minimum number of probings with the standard period of one seconds each to reach the conclusion about the failure. Killing a container causes a faster detection since it would be triggered with one failed probe, but for the pod the minimum failed probe becomes two. Decisions or acting on the containers takes longer than pods since K8s categorizes the container failures as application failure and goes into crash loop backoff, but a pod failure is counted as of K8s and is recovered faster.

To reflect on the agility of the deployments in ATHENA, we have done a concise analysis of the lifecycle of an E2E deployment of CNFs on both ATHENA and ETSI-OSM [14] release 11. This data is gathered in the Tab. II, where one could observe ATHENA is far more agile than the OSM. The data is averaged over several deployment scenarios with the same Kubernetes version (v1.24). The two clusters have the same machines running Ubuntu 18.04.1 (kernel 5.4.0-107-generic) over 8 virtual CPU cores of Intel Core i7-8550U, at 1.80 GHz baseline, scalable up to 4 GHz and with RAM size of 32 GBs. The data is collected by the timestamp of the corresponding objects or logs. Since Kubernetes does not record timestamps of shorter than one second, for some actions in ATHENA, we could just give the upper bound of 1 second.

We have performed the comparison with Charmed deployment of OSM and for onboarding a Charmed unit to have a similar structure of MANO. On deletion phase, OSM uses long graceful shutdown timeouts by default (30 seconds), but in ATHENA the API server gracefully terminates the process internally by detecting the terminate signal from K8s; hence it does not need to wait for the timeouts. Note that, for the numerous iterations of a deployment, the ratio is more important than the difference, because in large number of sequential deployments the small differences grow to a lot.

B. Performance and Overhead

To form a conclusive image of our platform, we demonstrate the E2E 5G network throughput UDP and TCP throughput achievable in ATHENA, compared to a fully PNF deployment on bare metal and a deployment with Snaps[13], shown in the

[13]https://snapcraft.io

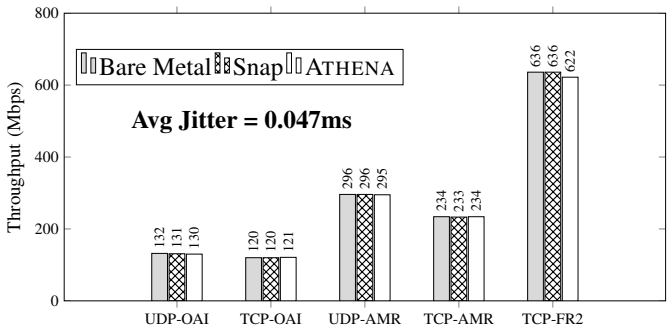


Fig. 5: Comparison of the throughput between bare metal, Snap, and K8s deployments of ATHENA.

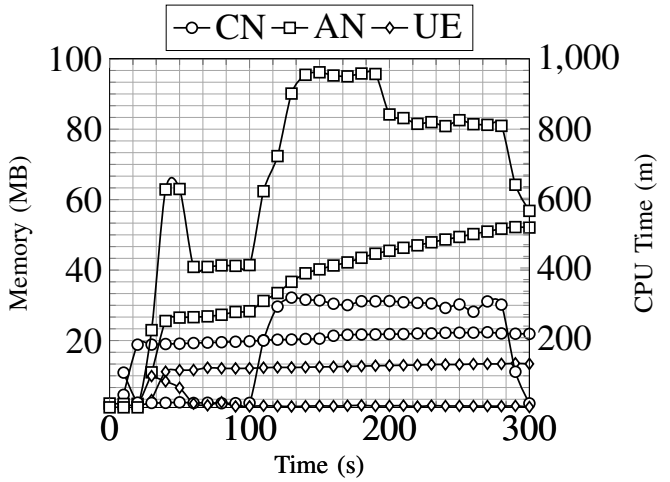


Fig. 6: CPU and RAM usage of option 1 in Tab. IV.

Fig. 5 for OAI on B210 (SISO), AMR on SDR50 (MIMO 2x2), and AMR on SDR100 (MIMO 2x1). All the tests are performed on the same machine, running RedHat Enterprise Linux 8 with 12 cores of CPU of type Intel Core i9-10920X at 3.50GHz base frequency and 64GB of RAM. The configuration of the RAN is the same over all the setups: 5G-SA FR1, 106 PRBs, 40 MHz bandwidth, TDD band 78 with the pattern of 7DL:2UL slots and 6DL:4UL symbols. For the FR2 setup we used 5G-NSA where the LTE cell is configured with 20 MHz bandwidth in FDD band 66, and the NR cell is configured with 100 MHz bandwidth, 120 kHz subcarrier spacing, and TDD band 261 with the pattern of 3DL:1UL for the slots 10DL:2UL for the symbols. In this case, the results show the aggregated throughput of the two cells that are running with MIMO 2x1. The data shows negligible difference between the throughput on different setups, and the slight variation could only be caused by minor aerial differences.

Gathered in the Tab. III, one finds a comparison between Charmed OSM and ATHENA in terms of the computing resources, as well as the corresponding total image sizes. Unfortunately, OSM does not define limits for its resource usage, so we relied on K8s top command. To make a fair comparison, we have grouped the components of both Charmed OSM and ATHENA into the three layers defined earlier in Sec. IV. For the OSM, we have considered the Model and Charmed Operators,

LCM, PLA, and POL as the necessary parts of the Operator Plane, while taking into account an average for Charmed Operators for workloads as replacement for the Management Plane, whereas the RO, databases, messaging system (Kafka and ZooKeeper), and authentication management (Keystone) are counted as part of the Orchestration Plane. We have not counted in the extra overhead of having Juju, LXD, and MicroStack installed as underlying parts of the Charmed OSM setup, but ATHENA needs nothing more than a minimal Kubernetes stack to Operate on top of it.

VI. USE CASES

Though ATHENA's design model encompasses the support for numerous use cases in 4G, 5G, and beyond; we emphasize on three use cases foreseen in 5G and beyond, which are in particular attention of ATHENA too.

A. Private Networking and Optimization

To demonstrate the capabilities of a platform for private networking use cases, one should consider at least the following three substantial domains: (1) Network **Customization**, possibly involving sharing network components; (2) Cost and Energy **Optimization** across variations of deployment decisions, constrained by the desired Key Performance Indicators (KPIs); (3) **E2E** capabilities including observability and automation with the UEs in the loop, since in many cases, the UEs would be deployed and controlled by the owner of the private network. Consider the scenario of an event in a museum to exhibit recently found relics for a short period of 10 days in a town. The network operator wishes a **short-lived** private network to serve some augmented media to the visitors, demanding 80 Mbps DL UDP payloads inside the venue. The goal is to optimize the cost of the deployment while trying to respect the regulations imposed by the city hall for the energy efficiency with 50 Watts power budget per base station. Peeking the data from Tab. II, and III, it can be seen that ATHENA's overhead and agility are suitable for short-lived networking including network leasing and temporary service boosts following the user demands.

ATHENA Cost Optimizer Operator is an optional Operator on the Operator Plane which given a Service-Level Agreement (SLA) object as well as cost constraints, it would iterate on various options in the network deployment to find the minimum cost deployment that still satisfies the SLAs. The Composition Models are incorporated with metadata that is particularly useful for pricing and cost optimization. The cost of a network is calculated by simple summation over the cost of each Composition Model used. These would be computed to the Capital Expenditure (CapEx) while the Operational Expenditure (OpEx) is estimated by the resource consumption over a period of time. Given the scenario, Cost Optimizer Operator sweeps the four options with a tolerable range of $[0, 100]$ Watts for the power usage (twice the regulation, but still fair penalty) and $[0, 2\%]$ for maximum packet loss to improve the QoS.

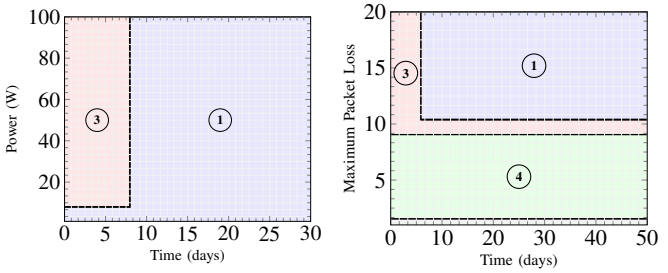
Using its observability capabilities on top of Prometheus over the OMI, ATHENA gets the computing resource consumption of the deployed scenarios in real time. A set of pre-defined, static measurements would not be helpful more than giving some approximate intuitions in general, so Cost Optimizer Operator runs the optimization following the SLA description and on-demand for the scenario. The Fig. 6 shows resource consumption of option 1 in the Tab. IV, where the CPU consumption is summed over the 10 CPU cores of type Intel Core i9-10920X at 3.50GHz, and averaged for intervals of 10 seconds for smoother plotting. Then the overall cost of the scenario would be calculated via an integral over a standard interval of 5 minutes, where the last 3 minutes have a single UE deployed transmitting on 80Mbps downlink via iperf3, placed in a Faraday cage. The integral would include the pricing model specified to the Cost Optimizer Operator as possibly non-linear function. However, for the sake of simplicity and proper reference, we have used Google Cloud Platform (GCP) pricing policy as an example reference in the Tab. IV. The average OpEx reflects the average cost per hour for the entire 5-minute duration, whereas the peak OpEx only considers the portion of the iteration where the UE is active. In the Tab. IV, the power measurements are collected by the Cost Optimizer Operator from a digital Watt-meter deployed on the machines and connected to their power supplier, and it factors out the idle usage of the machine. It should be noted that radio devices consume power even on standby; hence the idle power is measured with no cards or devices connected to the motherboard.

The total cost (C) after n days could be calculated via the formula $C = \text{CapEx} + n \times 24 \times \text{OpEx}$, where 24 is the number of hours per day, and the units are taken from the Tab. IV.

ATHENA Cost Optimizer Operator closes its loop by involving ATHENA Terminal Operator through the Operator Plane via injection of a Terminal object, which also provides the required live testing utilities. Reacting to the corresponding event that Terminal Operator would issue after successful attachment and setup of the UE to the network, the Cost Optimizer Operator starts measuring the KPIs of the E2E deployment. In result, we have the Fig. 7, where the best

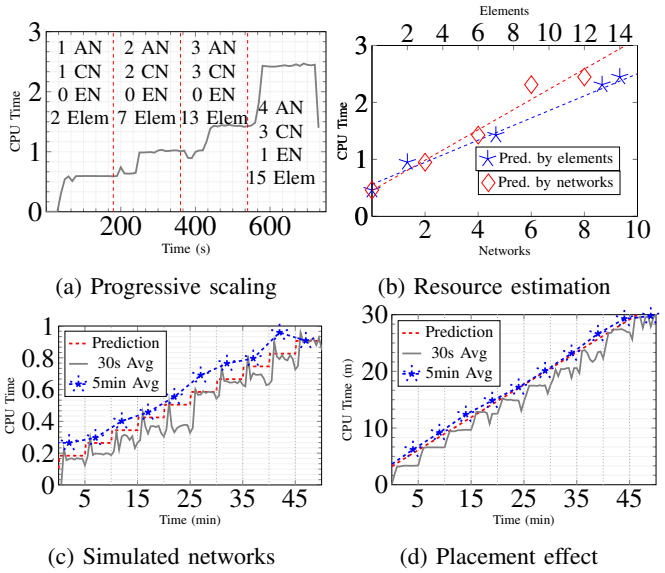
TABLE IV: Example of 4 iterations for Cost Optimization

#	Setup	CapEx (k$)	Avg OpEx ($/h)	Peak OpEx ($/h)	Power (W)
1	AMR SimpleRAN + SDR50	15	73.04	91.87	8
2	AMR SimpleRAN + B200	14	100.25	112.54	9
3	OAI Monolithic + B200	2	141.18	171.20	10
4	OAI O-RAN + AW2S	8	211.69	235.10	90



(a) Energy as criterion (b) Packet loss as criterion

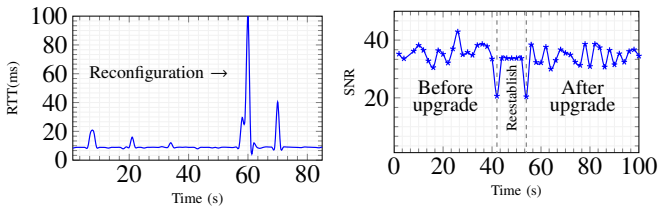
Fig. 7: Phase changes for cost optimization; Maximum Packet Loss is measured per 1000 packets.



(a) Progressive scaling (b) Resource estimation

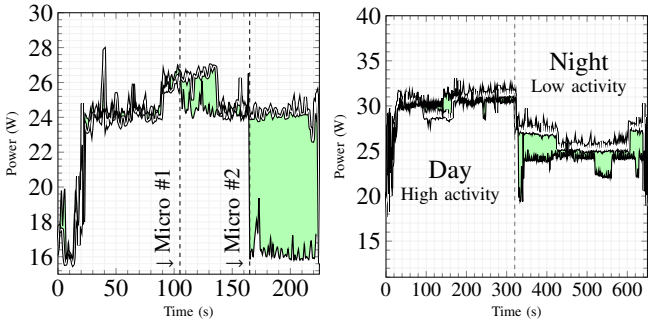
(c) Simulated networks (d) Placement effect

Fig. 8: Resource usage scaling of Open RAN



(a) Reconfiguration (b) Upgrade

Fig. 9: Day-2 Operations



(a) Micro-decisions (b) Macro-decisions

Fig. 10: Power saving in ATHENA.

choices are indicated by their numbers in the Tab. IV. On the Fig. 7a, the plot presents the variation of best option based on the cost of average usage (60% of time in the peak) depending on the period of deployment and power consumption constraints. The temporal dimension is important in this graph since it helps to flatten the CapEx over OpEx in the long term. On the right, we considered the packet loss reported on the iperf3 measurements with the cost of deployment on peak usage. Due to the given time period and SLAs of our scenario, the best choice for a single node setup is to have option 1 which could guarantee less than 1% packet loss. It has to be noted that the conditions and scenario are relaxed for the sake of simplicity, but the procedure is valid for any arbitrary scenario.

It is worth noting that ATHENA is reproducible and consistent by design, hence the whole process of each iteration could be completely automated. Otherwise, the results of the experiments could not be trusted. Besides, the agility of the platform (Tab. II) is crucial since the deployment time would determine the time for each individual iteration, yet still the reconfiguration capabilities of ATHENA could be exploited to reduce deployment time from two highly correlated network deployment options.

B. Open RAN and Emerging Networks

Open RAN is an initiative led by operators to break from vendor lock-ins and mix-and-match components of the network, even beyond the 3GPP specifications through the newly defined open interfaces. We have built the idea of multi-x as the extension to the same doctrine mixed with the fundamental goals of cloud native, which goes beyond the RAN itself to the MANO/OAM. ATHENA defines open interfaces in between its components, which allow them to be replaced at any time with any customized implementation of the same API. ATHENA is agnostic to the cloud provider, K8s distribution, container runtime, or OS.

Aligned with the Open RAN, as a cloud-native O-RAN OAM, ATHENA supports concurrent deployment of Open RAN components as well as dynamic transition between any of the options for deploying the RAN with minimal down-time and human intervention. To illustrate this, we have composed a scenario in which ATHENA progressively deploys a complex network of varying vendors and splits in multiple requests to resemble a growing network of an operator. The Fig. 8 shows how the CPU usage of the whole cluster as an example of computation resources changes during the growth of the network and its correlation with the scale of new workloads introduced. Based on the number of Elements and networks deployed in each phase, we have done an extrapolation of the resource consumption. The results have higher correlation score with Elements, due to the fact of variations in the network compositions and vendors used in the setup. This means to predict and provision resources for an Open RAN deployment; one should rely on the number of Elements to be deployed rather than the number of networks. For the Fig. 8c, we have used OAI RAN in RF simulator mode, and in intervals of 5 minutes, one 5G SA RAN and UE simulators have been introduced. The prediction has been made based on the number of elements which indicates higher non-linearity, despite the larger sample size. We hypothesize in the Open RAN scenario of the Fig. 8a, the variations offered by different deployment models and vendors have absorbed some uncertainty caused by the distribution of nodes in the cluster. However, in a more homogenous scenario of the Fig. 8c, other factors become more dominant. On the other hand, in the Fig. 8d, we have adjusted the pod scheduling via ATHENA Mobility Control Operator to place the simulated UEs on the same machine with the simulated gNB. This has drastically changed the regime of resource usage into a highly linear but significant CPU time.

The RTT experienced by the user also drops from an abnormal amount of 88 ms to 17 ms on average, showing the effect of placement on the QoS. In conclusion, a multi-x platform is the key enabler to unlock the Open RAN potential not only for deployment of tailored networks, but also for performing valid and reliable measurements.

To reflect on the day-2 operations of ATHENA, we present two operations in the Fig. 9: (1) A zero down-time reconfiguration of AMR RAN in response to topology change and addition of a new core network; (2) An upgrade procedure, where the AMR CN that was connected to an AMR RAN instance gets upgraded to a newer version, and in response to the UE performs a connection reestablishement procedure. For the first trial, the Fig. 9a, shows the E2E RTT of the UE, which without a down-time, experiences only 3 seconds of service degradation. The second trial in the Fig. 9b, plots the SNR values observed in the RAN for one particular UE. Since the bearer between the RAN and the CN gets lost, the UE tries a service connectivity request. The request is rejected due to the context loss in the CN, but the UE tries again by establishing a new NAS session over a new RRC. The NAS request fails because the UE tries SUCI instead of SUPI, but the CN is unable to reveal the concealed identity of the UE. Finally, a second reestablishment is successful, and after 12 seconds of lack of E2E service, the UE is back on again. For upgrades of the service once each month, 12 seconds of service outage is equivalent to 99.9995% service reliability. Thus, ATHENA is one of the few platforms able to continuously deliver (CD) with the required service reliability of five nines or six nines.

C. Green MANO/OAM

Sustainability and energy efficiency have been listed already in the requirements for 6G [27, 15], while the foundations to achieve them are already established by the lean-design, fine-grained controlling, and variety of optimizations in 5G. ATHENA's design is solicitous to sustainability and green computing, by providing built-in features in its observability and control mechanisms. To realize a green MANO/OAM one should consider two factors: (1) The extra overhead introduced by the MANO/OAM specifically in terms of computation resources should be minimal (see Sec. V-B); (2) MANO/OAM should provide means by which algorithms on energy optimizations could be applied. The flexibility and Day-2 operation offered through macro-decisions and micro-decisions in ATHENA are the key to resolve over-provisioning and achieve a sustainable deployment. These decisions span over the whole spatio-temporal dimension as demonstrated in the following examples.

The Fig. 10 demonstrates a tradeoff between QoS and energy saving in general; the solid black lines are the optimizations made by ATHENA and the doubly white lines are the baseline. The area in between the graphs is the power saved. The decisions, either micro or macro, would affect user performance, either in terms of service outage or QoS drops. Either way, one could encapsulate this as a Quality Reduction Metric (QRM). The former metric should be normalized by the number of affected users to form a more accurate picture of the impact. The Power Saving Score (PSS) shows the ratio of average normalized power consumption where the baseline power is deducted from the average values to show only the difference. The operator defines these two metrics based on the observed raw data as well as his desired balance between them. The Energy Optimizer Operator configures its decision as well as Manager's parameters to resolve the tradeoff accordingly. The metrics are ratios and have no dimensional units, hence mathematically comparable. Both the following experiments on power consumption are done with OAI gNB over USRP B210.

To demonstrate an example of a micro-decision in ATHENA specifically for green MANO/OAM, we provide a scenario in which the tradeoff between availability and energy-efficiency stands out. Imagine a UE that is moving around a facility to collect and upload some data. A few gNB nodes are deployed along the path, but since the data collection is frequent yet geographically scattered, the energy optimization becomes important. The required UL throughput is 5 Mbps, and any value below it counts as violation of SLA. When the UE's uplink channel gets worse enough to toss the balance between the mentioned metrics, the UE is dropped from the RAN side; otherwise the bad uplink would require heavy computation to provide a weak service. This decision, as shown in the Fig. 10a, saves on average 1.74 W (17.4% PSS) per UE with 61.54% QRM. Also, an unused RAN instance would be temporarily set to standby and remain in sleep mode for the duration on which there is no apparent activity from the UEs. The RAN is awakened at the moment that a UE is activated in its vicinity on a proximate node, and the whole process of detection and activation of the dormant instance all together takes less than 1 second, but it allows us to have on average 7.83 W (78.3% PSS) power saving per gNB. In this case, QRM could be considered almost zero for an agile MANO/OAM.

Moreover, ATHENA exposes some Workload-dependent statistics via its OMI interface in Manager where later could be processed by an Energy Optimizer Operator on the Operator Plane to take decisive actions with respect to energy saving and green computation for longer cycles, like day and night shifts, exploiting patterns discovered by AI/ML [18, 17]. For example, in the Fig. 10b, we have shown a saving of on average 2.24 W (22.4% PSS) per gNB for lowering the bandwidth from 40 MHz to 10 MHz, during the second half of the trial period due to the pattern enforced by the Energy Optimizer Operator, causing 77.36% QRM during the nighttime. If we consider 20% active users during the nighttime, normalized QRM becomes 15.47%. Of course, AI/ML could be handy in processing this information and adjusting the Energy Optimizer Operator's parameters, to make sure the decision is desirable. Weighting the QRM and PSS the same makes this particular decision favorable.

The micro- and macro-decisions may be combined to maximize the energy efficiency. The Energy Optimizer Operator considers a green budget per user as part of its SLA, indicating the maximum tolerable QRM over PSS value. This budget would be first spent on macro-decisions and then for micro-decisions. Since the Manager observes RAN state directly and the QoS in real-time, it could by itself decipher the remaining

budget from the tradeoff parameters and monitoring the RAN. Because of their timescale, the micro-decisions could adapt very fast to the side effects of the macro-decisions.

During our experiments in the Fig. 10, we noticed common tools like powertop are incapable of capturing true power usage of the RAN, because their scope is limited only to the ACPI interface of the CPU, even though the RAN could include RF devices and accelerators that are not reported via the same interface. Thus, we used a battery-powered machine that provides power readings from the battery. The setup is done via USRP B210, which takes all the power from the USB port of the machine. These metrics are exposed to the manager container via SysFS by the device manager for micro-decisions, and another node agent reads the coarser data, with higher periods, exposing them to the Prometheus for the usage in the Operator Plane.

VII. Conclusion

In this paper, we presented ATHENA, the fourth generation of MANO/OAM with innate support of various use cases in 5G and beyond, demonstrating intelligent, light-weight (Tab. III), agile (Tab. II), and green (Fig. 10) automation of CNFs in multi-x environment (Fig. 8a). The design is conclusively cloud native with several abstractions on its Operator Plane which transform K8s resources to multi-x logical networks and further up as logical network entities. The platform is highly extensible with several built-in Operators to support Open RAN, private networking, green MANO/OAM, and slicing. The Operators in this Plane could perform optimizations on cost or energy to facilitate design and deployment of private and sustainable networks. ATHENA preserves the performance while keeping the tight isolation through specialized device control and utmost declarative automation and observability, debunking the established beliefs around cloud native immaturity for telco use cases (Fig. 5). ATHENA is designed to Operate networks of any scale with minimal overhead and footprint, without compromising agility. The Managers in ATHENA form a distributed plane capable of performing zero down-time Day-2 Operations, rapid lifecycle management, and micro-decisions. We expect ATHENA to become the foundation of numerous novelties in research and development studies on 5G and 6G, since it is demonstrably capable of providing all the features demanded from a next generation MANO/OAM .

Acknowledgment

This work has been funded by Davidson Consulting, France and partially supported by the European Commission as part of the H2020 program 5G-Victori project, under the grant agreement 857201; the Horizon Europe 2022–6Green project, under grand agreement 101096925; the ImagineB5G project, under grand agreement 101096452; as well as the LeadingEdge project under the CHIST-ERA SDCDN topic. In addition, the results of this work have been supported by BubbleRAN's cloud-native 5G Open RAN infrastructure.

References

[1] 5G Americas. *5G and the cloud*. Tech. rep. 2019.
[2] O-RAN.WG6.O-Cloud Notification API-v03.00. *O-Cloud Notification API Specification for Event Consumers*. Tech. rep. O-RAN Alliance, 2022.
[3] Osama Arouk and Navid Nikaein. "Kube5G: A Cloud-Native 5G Service Platform". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–6.
[4] Alcardo Alex Barakabitze et al. "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges". In: *Computer Networks* 167 (2020), p. 106984. URL: https://doi.org/10.1016%2Fj.comnet.2019.106984.
[5] G. Biczók et al. "Manufactured by Software: SDN-Enabled Multi-Operator Composite Services with the 5G Exchange". In: *IEEE Communications Magazine* 55 (2017), pp. 80–86.
[6] Brendan Burns, Joe Beda, and Kelsey Hightower. *Kubernetes Up & Running; Dive into the Future of Infrastructure*. Second Edition. O'Reilly Media, Inc., 2019, pp. 3–6.
[7] Massimo Condoluci and Toktam Mahmoodi. "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges". In: *Computer Networks* 146 (Sept. 2018).
[8] Kubernetes contributers. *Pod Lifecycle*. URL: https : / / kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle.
[9] Mavrakis Dimitris and Saadi Malik. *Open RAN: Market Reality and Misconceptions*. Tech. rep. ABI research, 2020.
[10] Jason Dobies and Joshua Wood. *Kubernetes Operators*. O'Reilly Media, Inc., 2020, pp. 67–77.
[11] Sevil Dräxler et al. "SONATA: Service Programming and Orchestration for Virtualized Software Networks". In: (May 2016).
[12] Korian Edeline and Benoit Donnet. "A Bottom-Up Investigation of the Transport-Layer Ossification". In: *2019 Network Traffic Measurement and Analysis Conference (TMA)*. 2019, pp. 169–176.
[13] Seth Gilbert and Nancy Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services". In: *SIGACT News* 33.2 (2002), pp. 51–59. ISSN: 0163-5700. URL: https://doi.org/10.1145/564585.564601.
[14] OSM End User Advisory Group. *OSM SCOPE, FUNCTIONALITY, OPERATION AND INTEGRATION GUIDELINES*. Tech. rep. ETSI, 2019.
[15] Tongyi Huang et al. "A Survey on Green 6G Network: Architecture and Technologies". In: *IEEE Access* 7 (2019), pp. 175758–175768.
[16] Bilgin Ibryam and Roland Huß. *Kubernetes Patterns*. O'Reilly Media, Inc., 2019, pp. 131–134.
[17] China Mobile Research Institute. *C-RAN The Road Towards Green RAN*. Tech. rep. China Mobile, 2020.
[18] Leonardo Militano et al. "AI-powered Infrastructures for Intelligence and Automation in Beyond-5G Systems". In: *2021 IEEE Globecom Workshops (GC Wkshps)*. 2021, pp. 1–6.
[19] Technical Specification Group Core Network and Terminals. *TS 29.244 V17.4.0*. Tech. rep. 3GPP, 2022.
[20] Borja Nogales et al. "Design and Deployment of an Open Management and Orchestration Platform for Multi-Site NFV Experimentation". In: *IEEE Communications Magazine* 57.1 (2019), pp. 20–27.
[21] O-RAN.WG1.O-RAN-Architecture-Description-v07.00. *O-RAN Architecture Description*. Tech. rep. O-RAN Alliance, 2022.
[22] O-RAN.WG10.OAM-Architecture-R003-v08.00. *O-RAN Operations and Maintenance Architecture*. Tech. rep. O-RAN Alliance, 2023.
[23] ONF. *ONAP Architecture Overview*. Tech. rep. ONF, 2018.
[24] Nikolaos Sapountzis et al. "Reducing the energy consumption of small cell networks subject to QoE constraints". In: (Feb. 2015), pp. 2485–2491.

[25] Dominik Scholz et al. “Performance Implications of Packet Filtering with Linux eBPF”. In: *2018 30th International Teletraffic Congress (ITC 30)*. Vol. 01. 2018, pp. 209–217.

[26] Kyuho Son et al. “Base Station Operation and User Association Mechanisms for Energy-Delay Tradeoffs in Green Cellular Networks”. In: *Selected Areas in Communications, IEEE Journal on* 29 (Oct. 2011), pp. 1525–1536.

[27] FG-NET2030; Focus Group on Technologies for Network 2030. *Network 2030 - Additional Representative Use Cases and Key Network Requirements for Network 2030*. Tech. rep. ITU-T, 2020.

[28] Dariusz Wypiór, Mirosław Klinkowski, and Igor Michalski. “Open RAN; Radio Access Network Evolution, Benefits and Market Trends”. In: *Applied Sciences* 12.1 (2022). ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/12/1/408.