Journal of Network and Systems Management (2023) 31:36
https://doi.org/10.1007/s10922-023-09728-1

Assisted Monitoring and Security Provisioning for 5G Microservices-Based Network Slices with SWEETEN

Rafael de Jesus Martins[1] · Juliano Araújo Wickboldt[1] · Lisandro Zambenedetti Granville[1]

Received: 17 July 2022 / Revised: 14 December 2022 / Accepted: 30 January 2023 /
Published online: 12 February 2023

Abstract

5G networks have imposed a drastic shift in how mobile telecommunications must operate. In order to comply with the new requirements, solutions based on network function virtualization (NFV) and network slicing must be carried out. Regarding NFV in particular, the trend towards pulverizing the monolithic software in a microservices-based one carries network management challenges to operators. The deployment and integration of one or more network management software with the managed services is as important as it is delicate, as stringent requirements of 5G applications must be respected. In this paper, we propose SWEETEN as a solution for automating the deployment and transparently integrating network management solutions from different management disciplines, in this case, monitoring and security. Demonstrating its usability through a intelligent healthcare use case, SWEETEN is shown to transparently provide monitoring and security solutions for a complete network slice, enabling compliance with privacy requirements through minimal low-level interventions from the network slice tenant. The results show how SWEETEN integration of monitoring and security disciplines can assist users in guaranteeing the correct operation of their deployments regardless of the underlying software solutions used.

Keywords 5G · NFV · Network management · Microservices

Juliano Araújo Wickboldt and Lisandro Zambenedetti Granville have contributed equally to this work.

✉ Rafael de Jesus Martins
rjmartins@inf.ufrgs.br

Juliano Araújo Wickboldt
jwickboldt@inf.ufrgs.br

Lisandro Zambenedetti Granville
granville@inf.ufrgs.br

[1] Institute of Informatics, Federal University of Rio Grande do Sul, Av. Bento Gonçalves, 9500, Porto Alegre, RS 91509-900, Brazil

1 Introduction

Telecommunications have been undergoing massive evolution in the last years with the specification and launch of 5G networks. While previous generations mostly focused on improving customers data rate, 5G networks cover a wide range of applications with diverse requirements by offering disruptive improvements in reliability, device density, and coverage, to cite a few. In order to comply with such requirements, 5G networks must employ modern techniques for network slicing and network function virtualization (NFV) [1].

Since its inception, NFV has drawn attention from academia and industry because of the benefits it offers in comparison to traditional middlebox appliances (*e.g.*, firewalls, deep packet inspectors (DPIs)). NFV decouples the proprietary hardware from the associated software, enabling the network functions to run on top of commodity hardware [2]. This shift enables dynamism and scalability much needed for 5G networks, all the while reducing operational costs for mobile carriers. In turn, these virtual appliances represented by virtual network functions (VNFs) present their own challenges, in particular when a function is pulverized in multiple microservices that must cooperate to deliver the network service appropriately [3]. In this case, the 5G application's requirements must be carefully considered by the VNF manager, for the VNF as a whole and for the individual microservices comprising it.

Network management is a complex process that can include disciplines such as monitoring, securing, and configuring network devices. With the virtualization of network functions and further with the shift from monolithic software to microservices-based ones, networks become increasingly reliant on proper management of their components [4]. More then ever, individual pieces of the network must be properly managed so that the myriad of applications that 5G enables can be truly experienced by the end user. Newer well-rounded management solutions such as service meshes can provide a variety of network management capabilities to multiple applications interconnected through microservices. These technologies often rely on deploying a separate container with every microservice for the managed applications, which interfaces all connections from and to the application container, and therefore are known as sidecar proxies [5]. Such solutions, however, are mostly fit for complex deployments, and often introduce network overhead (due to the additional hop per microservice in a flow) that can hinder their adoption depending on applications requirements.[1] Extreme cases such as edge applications could even suffer from the computational overhead from the additional containers deployed. Specific requirements for each application must therefore be considered when provisioning the network management capabilities, as there is no one-size-fits-all solution for all different applications thus far.

Considering that 5G networks are envisioned to support upcoming mission-critical applications, security becomes a primary concern. Targets can range from governments and industries to ordinary citizens. For example, eavesdropping e-health

[1] https://medium.com/@pklinker/performance-impacts-of-an-istio-service-mesh-63957a0000b.

devices can leak confidential information regarding their users, and thus impose an important security requirement for the setup [6]. Additionally, battery limitations from these devices can result in minimal computational overhead being acceptable for management solutions. A different application with similar security challenges that runs in the cloud, conversely, could make use of more robust management artifacts that would incur in greater overhead overall. Since applications and resulting requirements can vary significantly, and because there is an increasing number of management tools offered for various contexts, correctly choosing and configuring a set of tools for each scenario becomes a challenging task even for experts.

In this paper, we present how SWEETEN (a**S**sistant for net**W**ork manag**E**m**E**n**T** of micros**E**rvices-based V**N**Fs) can help VNF operators and network slice tenants by including management features from multiple disciplines (*e.g.*, monitoring and security) in operators' and tenants' deployments in a transparent manner. This work is a direct evolution on our previous one [7], where managed entities were limited to VNFs and only monitoring was implemented as a management discipline. Now complete network slices are covered by SWEETEN, which is also able to provide both security and monitoring solutions. By augmenting their deployment specification with high-level management features request, tenants can enrich their deployed entities with automatically chosen and configured management tools and have their management needs fulfilled. When deemed necessary, the user can specify lower-level configuration parameters in order to fine-tune how the complete solution should be put together by SWEETEN. Natural language processing (NLP) is employed to extract meaningful tags from users deployment descriptions, which are then used to provide tailored solutions for each deployment. After the deployment phase, the system provides the user with an integrated dashboard and an API that allows the operator to manage their VNFs and network slice from a high-level perspective, regardless of the specific management tools selection and their interfaces at the lower-level. In this respect, SWEETEN's main contributions presented in this paper are:

- Automated configuration and deployment of network management tools following user's high-level specification, offering management solutions for novice users with ease;
- Varying abstraction levels in the specification are allowed, enabling fine-tuning of the solution by advanced users;
- Integrated dashboard, allowing transparent management for all entities of interest regardless of underlying software configured to realize the network management required.

We implemented a prototype to evaluate SWEETEN in terms of providing management capabilities to an intelligent healthcare use case. In this use case, patients can be monitored by a number of resource-constrained Internet of Things (IoT) health monitors and other resourceful devices in real-time, enhancing quality of human life through the automated execution of mundane tasks [8]. To achieve that, data collected by said devices is sent to a deep learning module hosted in the cloud, which processes the data from a patient and triggers alarms when events happen. Due to

the sensitive nature of the traffic exchanges by the healthcare devices, security in the terms of privacy is a foremost concern for all communication. It should be noted that, since these devices can be resource-constrained, solutions should balance the defensive mechanism effectiveness and the overhead they produce. Moreover, IoT devices in this use case utilize narrowband IoT (NB-IoT) for their radio-access technology as it offers improved coverage and efficiency in terms of cost and power consumption [9]. Cloud radio access network (C-RAN) is used to deliver connectivity to the application's devices, imposing stringent latency and data rate requirements that must be met throughout the deployment life-cycle and thus implying the need for careful monitoring.

By using SWEETEN, the results show that operators can request management features and come up with solutions from a high-level perspective, with no need for expertise in specific tools and configurations that would typically constitute a burdensome work for experts. Through the inclusion of annotations to their specification, users receive management solutions configured for their needs. The additional management entities deployed represent a significant overhead regarding the deployment time, but considered acceptable for the benefits offered and due to its infrequent occurrence. Minimal computational overhead was also perceived for when the solution is deployed, while a much more prominent overhead is present regarding network overhead. Notwithstanding, the overhead is much more due to the included management entities themselves and not due to SWEETEN usage, and would thus also be present if a similar management solution were to be manually included by the user, therefore presenting a net gain for the user.

The remainder of this paper is organized as follows. In Sect. 2, we present background and related work. Then, we describe a proposed architecture in Sect. 3. In Sect. 4, we present a use case that illustrates management challenges faced by novel applications in 5G networks management. Then, we evaluate our proposal with the use case, discussing the results in Sect. 5. Finally, we present our conclusions and future work in Sect. 6.

2 Background and Related Work

Because this study spans over a few non-trivial concepts, contextualizing each of them and how they relate to each other is necessary. To this end, Sect. 2.1 presents the main characteristics of the microservices paradigm and current efforts towards managing software that follows such paradigm. Network slicing as a technique for delivering 5G use cases is discussed in Sect. 2.2.

2.1 Microservices

Monolithic software can be defined as a software composed by modules that cannot be executed independently [10]. Software design has generally followed a monolithic paradigm in which an indivisible software is responsible for realizing a complex service in an integrated manner. Monolithic software still can and

should be designed through a composition of specialized modules. However, the different modules in a software following the monolithic paradigm still rely on resource sharing (e.g., memory, CPU) for running in the same machine, which tightly ties all the components as one atomic application. While the monolithic architecture is viable for many applications, recent off-premise and distributed computing offered by cloud services impose the need for a more flexible paradigm in software design. In the microservices paradigm, systems are designed through independent components called microservices, which provide a system with cohesive and well-defined functionalities [4]. Context sharing between microservices is done through network messaging, allowing microservices to be deployed along a distributed infrastructure, as well as completely decoupling implementation details and choices (e.g., programming languages) between modules. Microservices introduce many benefits regarding continuous integration and delivery, for example, as updates for individual microservices may be gradually rolled out. However, the design also imposes new management challenges that must assert the correct operation of each microservice, and of the composed software as a whole.

Designing and developing software through the microservice paradigm can quickly become hard to manage as complex connection schemes are required among hundreds of microservices. Service meshes recently emerged as a solution for that through the automatic management for microservices connections, as reviewed in the study by Li et al. [5]. Among their benefits, service meshes can provide service discovery for the microservices and load balancing among different containers (even using different software versions). On the implementation side, these solutions usually employ an array of lightweight network proxies, which are deployed alongside the application containers and can provide an interface for all incoming and outgoing connections. Some specific scenarios that are much relevant to 5G, such as multi-tenancy, can however present specific challenges that were not part of service meshes design. SWEETEN is designed to provide management for VNFs and network slices with various requirements, such as the minimal computational and network overhead for IoT applications, and thus can provide the appropriate management services and configuration based on the user specification and high-level feature annotations.

Chowdhury et al. [3] highlighted the importance for the NFV ecosystem to have VNFs designed through a microservice architecture. In Service Function Chains (SFCs), for example, having monolithic VNFs incurs in unnecessary processing overhead from redundant functionalities. Instead, the redesign of these functions through microservices enable fine-grained resource allocation and independently scalable components, as elements for the orchestration of VNFs in an SFC become also present for the VNF-Components (VNF-C) for any VNF. Among the research challenges documented in the literature, the adequate monitoring of these functions is underlined, as well as questions pertaining performance profiling and overhead trade-offs, all occurring topics in our present research.

2.2 Network Slicing

Network slicing refers to the slicing of a single physical network into multiple isolated logical networks [11], and thus has emerged as a cornerstone for evolving 5G networks. While 4G and previous generation relied on a one-fits-all architecture to serve mobile network costumers, 5G covers a plethora of services with diverse requirements, and so the system itself must be customized to meet each customer's needs. With network slicing, the common network infrastructure can be harmoniously shared among multiple tenants, allowing their diverse requirements to be met while providing isolation between slices. End-to-end network slices can span over various network layers and heterogeneous technologies (e.g., RAN, core, and cloud), and can facilitate service delivery to customers while also enabling efficient networking and service convergence [12]. Network slicing helps providing the much needed dynamism and scalability in 5G networks, as customized end-to-end slices can be created on-demand and thus provide a cost-efficient manner to serve customers.

Slamnik-Kriještorac et al. [13] presented an extensive survey on the distributed and heterogeneous resource sharing that is taking place in 5G networks. The sharing model for 5G and other networks proposed by the authors is classified in three distinct models: technical, business, and geographic. Specifically, the technical model is structured in three layers: infrastructure, orchestration, and service. Although some management concerns for network slices are discussed, they primarily focus on the infrastructure layer, while the examples for service (e.g., Healthcare) and orchestration (e.g., Kubernetes) are left for each layer and case to solve individually. Our study, in contrast, proposes that the network management should consider all layers jointly, and also that this management is a complex task that operators should be assisted with when deploying new network slices.

Kist et al. [14] proposed a virtualization scheme that allows technologies and instances for different radio access networks (RANs) to be provided as services for network slice tenants. The proposal offers programmability and adaptability for service providers while maintaining isolation between tenants and their slices. An experimental scenario evaluated by the authors comprised of LTE and NarrowBand-IoT (NB-IoT) clients showcases how the proposed system allows the provisioning and management for virtual RANs (vRANs) for providers' network slices. The management aspect is however limited to the RAN infrastructure, and any additional required management aspects (e.g., regarding the application, or other VNFs included in the slice) are left to be determined and deployed by the slice owner.

Coelho et al. [15] looked into formally defining the network slice designing problem, proposing a framework that considers nested slices and network functions decomposed in smaller services and models the relationship between radio splitting, control and data planes isolation, and core network function placement. Leveraging the reusability of smaller network function services and network slices subnets, a variety of sharing policies that range from total isolation to flat sharing can be used to realize 5G services of any class, and fulfilling the stringent requirements each of them impose. The study therefore focuses on producing a network slice, including necessary network functions, their split and placement,

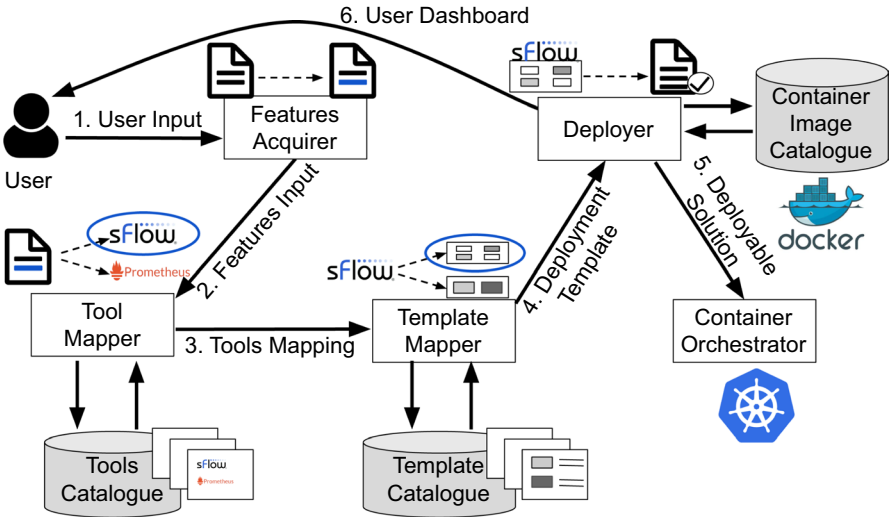

Fig. 1 SWEETEN architecture [7]

to deliver the demands posed by a number of services. The slice management itself, including the appropriate monitoring of the deployed network functions, is not covered by the slice design and thus is left for the operator to manually determine, configure and deploy the appropriate solutions.

Our proposal compliments the related work regarding microservices as we consider their specificities within a network slice when delivering a management solution, and do so from a high-level perspective. That, on one hand, allows the focus for designing and implementing a microservice not to be taken from the service itself, and on the other hand maintain management and application as separate as possible. The proposal's architecture is further discussed in the following section.

3 SWEETEN Design and Implementation

In this section, we review SWEETEN design, which has been previously introduced and evaluated in the context of network monitoring [7]. Moreover, additional implementation details are now presented in-depth for all the architectural components that comprise the system. An overview of SWEETEN is presented in Fig. 1. The remainder of this section presents the system progressively. General aspects and user input are discussed in Sect. 3.1. SWEETEN pre-processing is discussed in Sect. 3.2. The system's mapping of management tools and configurations is explained in Sect. 3.3. Finally, the process of deploying the mapped solution and returning a customized dashboard to the user is discussed in Sect. 3.4.

Table 1 Network features and respective tools listings [7]

	Monitoring	Security	Administration
Flows	sFlow, NetFlow, Prometheus	Snort (for IDS), OSSEC	–
Traffic	Prometheus, iPerf, SNMP	iptables, nftables	Linux tc
Latency	SmokePing, OWAMP, TWAMP	–	Linux tc
Device	Kubelet (Kubernetes native)	Syslog, antivirus utilities	NETCONF, SNMP

3.1 General Aspects and User Input

SWEETEN can be viewed as an automated assistant intended to help tenants of 5G slices to meet management requirements for their slice components. In particular, when these components are designed following the microservices architecture, network management must be thought of while respecting the modularity and isolation envisioned in this architecture. Moreover, a slice can span over thousands of microservices [16], which makes automated management not only a benefit but a necessity.

The user input in the designed system consists of a specification for user containers, which is augmented by the user to contain requests for management features. These management features can be of multiple disciplines, namely security, monitoring, and administration. Some examples for each discipline are found in Table 1.

With respect to the classification for network management features, three disciplines are considered, as presented in Table 1. The monitoring discipline encompasses all measurements that can be done to assert a network and its components are behaving as expected [17]. These measurements can be either passive (e.g., observing flows in a given interface) or active (e.g., probing a link to check latency and throughput available). The security discipline involves all sensitive aspects in a network, including privacy and resilience requirements and the means to guarantee them at a certain level [18]. The administration discipline is comprised of management tasks and applications that actively alter the network behaviour for one or more device. For example, Netconf protocol can be utilized to reconfigure switches and routers in a network, altering its behavior dynamically [19].

Having presented the user options for management features requests, an excerpt from a user specification with requests for both monitoring and security features is presented in Listing 1. While novice users can request high-level management features more easily, advanced users can specify lower-level configuration parameters that must be observed in the provided solution. This enables operators to promote network management capabilities from a high-level perspective, while also being able to traverse through lower-level configuration parameters when necessary. An example for such parameters addition is presented in Listing 2, where in addition to specifying which tool should be used (i.e., Prometheus), configuration for the measurement intervals and the dashboard are provided by the user.

```
apiVersion: apps/v1
kind: Deployment
metadata:
    management:
      monitoring:
        - flows
      security:
        - cryptography
    ...
```

Listing 1 Excerpt for a simplified user management service with requirement for management features.

```
      monitoring:
        - flows: TCP
          tool: Prometheus
          scrape_interval: 1s
          dashboard:
            - tool: Grafana
              http_port: 3333
          ...
```

Listing 2 Excerpt for a feature request that includes lower-level configuration parameters specification

3.2 User Images Pre-processing

The user input is received by SWEETEN through the Features Acquirer module. This module is responsible for retrieving information on all microservices defined in the user specification as well as the management features requested for each microservice.In addition to the nature of the management desired, requirements for the produced solution can be derived from requirements tags. To achieve that, each microservice with a management annotation undergoes three steps:

1. Management feature retrieval, where the requested features and options are extracted from the user specification.
2. Description enrichment, where the information about the user service is augmented.
3. Service tagging, where tags that better describe the service requirements are appended to the specification.

Each requested feature obviously can be realized by a number of tools. In order to allow the best possible match for tools selection and configuration in a later stage within SWEETEN, user's microservices undergoes a tagging process composed by the description enrichment and service tagging processes previously mentioned. For each component in the user input for which a feature is requested, SWEETEN appends tags that can provide some insight about the type of service and its requirements. Tags are later used to differentiate the tools and configuration choices for monitoring a cloud-hosted service from an IoT one; for example, while the former

can leverage network and processing resources to employ a robust solution, the later must realize the management necessities with minimal overhead.

To alleviate the burden for the user, SWEETEN can automatically derive tags from the user specification alone without any additional user input. To achieve that, the description for each container that composes a service is processed in the service tagging step. This description is rarely present and descriptive for most containers, so the description is enriched before tags are derived. For the purpose of our proof-of-concept, the description enrichment process is obtained through a simple Google web search query that is automatically requested by the system. This query is composed of the words "define" and the container image name, and the text for the first result is considered by the system. The service tagging process can then run the enriched description through a Natural Language Processing sub-module to extract the aforementioned tags, as described next.

Natural Language Processing (NLP) is an area of computer science that employs algorithms for learning, understanding, and producing of human language content [20]. NLP has lately been a tool in various areas with promising results, for example, by providing enterprises with network security insights and suggesting solutions when paired with a neural network model [21].

Among the many available algorithms for NLP, an important aspect that differentiates them is whether they require supervised learning or not. In SWEETEN's case, since we want not only to include current management features but also to allow the system to easily evolve and include new ones, unsupervised learning is preferred. Our model of choice is based on Latent Dirlecht-Allocation (LDA) [22], an unsupervised machine-learning algorithm that can help find common topics between multiple text documents. In this way, we initialize a database with enriched descriptions for three of the top containers for each category in DockerHub,[2] and stipulate seven different topics to be found. Each topic will contain a weighted list of words that indicates the prevalence of the main words for each topic. For a new document, i.e., the enriched description for the user microservice that is being processed, LDA associates a percentage for each pre-determined topic; later, the relevant words for the selected topics (e.g., indicating the resourcefulness of an object) are used to determine which solutions SWEETEN should prioritize, as explained next.

3.3 Management Tools and Templates Mappings

Management features required by the users must be realized by a set of management tools. The Tool Mapper module thus is the first one to make selections based on the Features Acquirer output. For each feature required by the user, this module maps to one or more tools that are capable of realizing such features. The listing for these mappings are provided through a Tools Catalogue, which has been already pre-populated by an expert. In the event that more than one tool are fit for a certain request, tags are considered so that the best fit can be

[2] https://hub.docker.com/.

provided. The algorithm for matching the tags to the available tools is a greedy one, so the solution that matches the most tags from the user input is selected each time.

Additionally, each tool must be configured and deployed so that it can perform the intended task correctly. For example, a firewall must be placed in front of a targeted back-end service, while an active latency monitor must be placed alongside the monitored microservice. Moreover, the previously appended tags must be considered when determining the configuration parameters for a given network management tool. A second stage for selection is thus performed by the Template Mapper. A template represents the configuration required by a management tool to be deployed, both with respect to the tools internal configurations and with any necessary cluster definition [23]. The configuration aspect is also covered by the tags appended to the user input, in a process analogous to the one described for the tools matching.

Occasionally, the correct configuration for some management tool might require some breaching of microservices architectural design during execution. For example, monitoring the active connections for a microservice requires for the the management tool not only to be placed alongside the managed service, but to share its network context too. This is achieved by namespace [24] sharing between managed and management services, but which is only performed when necessary. In this way, microservices design can be maintained for all applications, and specificities are configured and treated with templates designed for such cases.

3.4 Solution Deployment and User Dashboard

The elements of the solution must be put together in a deployable specification. Because we chose to use Kubernetes as the system's container orchestrator, the result is composed of two separate YAML [25] specifications: one for the user services that did not require management features, i.e., services that were already part of the user specification, but that did not require any management feature; and one for the remainder of the user specification plus all the network management tools included by SWEETEN. We chose Kubernetes because it is the most widespread platform for containers in industry and academia alike [26], which in the one hand offers an active community and a rich environment, and the other hand enables SWEETEN to be used by a wide audience.

Finally, during the slice lifecycle, the user can manage their services through a customized dashboard. Based on the user input, SWEETEN can deploy dashboards from different software. An example for a monitoring dashboard by Prometheus [27] provided by SWEETEN for a novice user is depicted in Fig. 2, while a more advanced Grafana dashboard provided by SWEETEN is depicted in Fig. 3. Non-visualizing features, such as cryptography introduced by security features, are presented textually for the user's knowledge. Additionally, the user can interact directly with the configured management container through a terminal, so they can still have control after the deployment phase for their slice. Thus, in comparison to our previous work, the ability to manage a complete network slice instead of being limited to

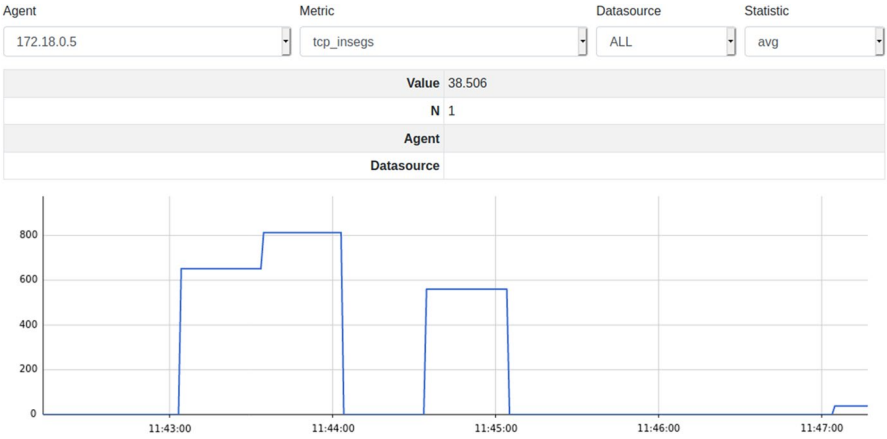

Fig. 2 User dashboard generated from the novice user's specification [7]

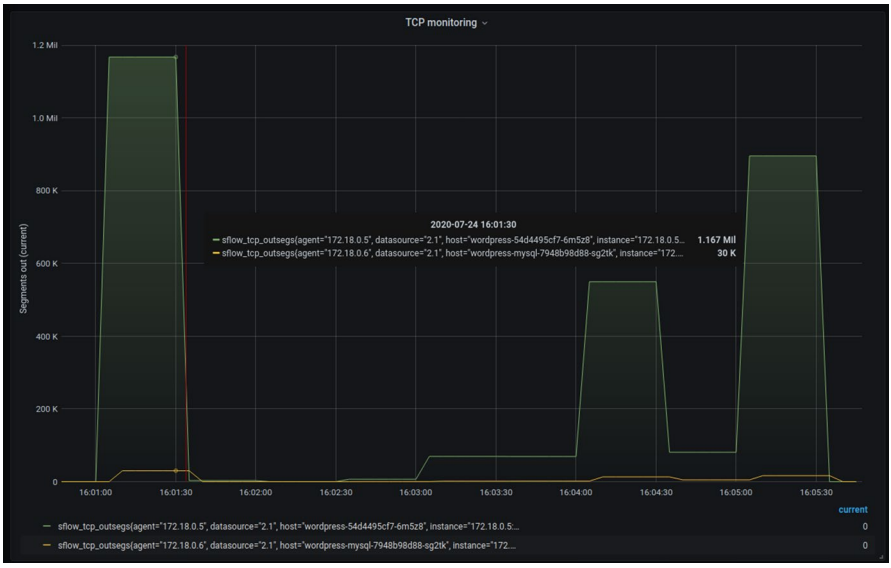

Fig. 3 User dashboard generated from the experienced user's specification [7]

its VNFs, and the inclusion of security coverage in addition to the monitoring ones present important advances towards a more complete management assistant.

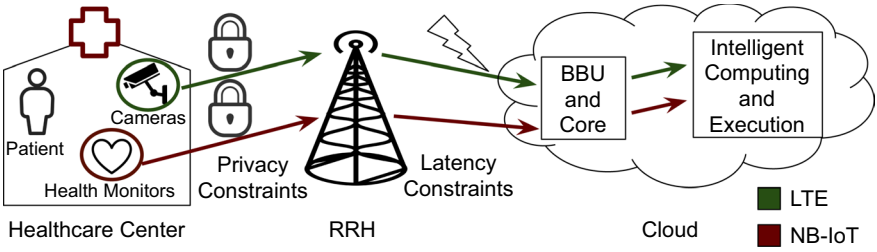

Fig. 4 Use case for intelligent healthcare application

4 Automated Network Management for an Intelligent Healthcare Use Case

5G systems promise a series of disruptive advancements for a myriad of applications typically classified in three scenarios. Enhanced mobile broadband (eMBB) addresses applications centered in multi-media content, services, and data; ultra reliable low latency communications (URLLC) encompasses critical applications that pose stringent requirements such as remote medical surgery; massive machine type communications (mMTC) is characterized by a large number of low-cost devices that transmit a low volume of data [28]. Some of the most technically challenging applications unite requirements from two or even all three scenarios. Healthcare applications can exemplify such a case, where a multitude of health devices of different capabilities and with distinct requirements are used to guarantee the well-being of patients. This use case is depicted in Fig. 4 and further expanded in the following.

Heart rate, respiratory rate, and body temperature monitors are a small sample from a large list of monitoring devices that can be utilized in a patient's health monitoring. The number of IoT devices employed in this scenario can grow significantly, providing abundant data to track patients physiological characteristics but also requiring a more extensive analysis by physicians and other professionals. In this context, these applications can be further benefited by the inclusion of intelligent algorithms to process and automate decisions, triggering alarms and actions whenever abnormalities are detected [8]. Artificial intelligence (AI) techniques based on novel models such as big data mining and deep learning can process large amount of data at real-time, and then predict and automate tasks at a rate impossible before. While the monitoring part must be performed on-premise, i.e., in the healthcare center where the patients are located, the burden of collecting and processing all the data can be effectively run in the cloud.

Because of the sensitive nature of the data monitored and transferred, security, in particular by the means of privacy, is a foremost concern. As hardware solutions are not always feasible and as new legislation advances the levels of privacy requirements for these applications, guaranteeing a certain security level from a software perspective is a necessity. In certain occasions, resourceful devices are used for patients' monitoring, such as 4K cameras that can record their movements, and

paired with deep-learning algorithms can detect facial expressions and gestures of patients and warn healthcare professionals in the event of an anomaly [8]. However, most monitoring IoT devices are constrained in terms of computational power and battery, and so possible security solutions should account for these limitations and prioritize lighter-weight solutions whenever possible. Less constrained devices, in turn, can afford to employ more advanced and intensive defensive mechanisms, and the provisioning for each case should reflect these characteristics. As a manual security approach is not feasible for complex 5G scenarios, security automation is a key principle in securing 5G applications and networks [29].

In recent years, multiple low power wide area (LPWA) radio technologies have emerged as options for delivering the scalability required by mMTC applications. From the alternatives, NB-IoT has been shown to offer promising results for healthcare applications [30]. NB-IoT is fully compatible with long term evolution (LTE), and can be deployed inside a single LTE physical resource block (PRB) of 180 KHz or inside an LTE guard band, potentially serving up to 50k end-devices per cell [31]. The limited 250 kbps data rate is plenty for hear rate and body temperature monitoring that require only 1 byte for payload every 5 minutes [30], but it is impractical for streaming the video from the deployed cameras. Devices as such that require extensive bandwidth must connect over standard LTE-A network, which is capable of meeting their demands.

Metrics collected by all the devices are reported to a remote radio head (RRH), the radio antenna responsible for communications to and from users' devices. The signals must then be processed by the network core, which is performed by the baseband unit (BBU) in a base station (BS). Previously, these functions would be performed exclusively by specific-purpose hardware. With the recent advances in virtualization and the expansion of NFV architectures, virtual base stations have been adopted by pioneering virtual mobile network operators. Such operators do not own the required wireless physical infrastructure, but instead lease it from traditional mobile network operators. The processing modules for wireless services, in turn, can be provided by software running in the cloud, enabling different strategies that benefit customers [32]. In the NB-IoT case, low protocol stack processing requirements and low latency-sensitivity make C-RAN an attractive alternative, so all the BBU processing and higher-layer protocol stacks are implemented by software that runs on the cloud [33].

A standard LTE-A BBU can be split in different functions [34]. The different split options offer possibilities of alternating between dedicated hardware and function virtualization, allowing the flexible adoption of functional split in time and location. Noteworthy, 5G specifications pose stringent network requirements for their communications, in particular with respect to data rate and latency. Regarding latency, a maximum delay of around 3ms for transmitting and processing the signal is determined by the hybrid automatic repeat reQuest (HARQ) mechanism adopted in LTE [35]. There is thus a stringent requirement (*i.e.*, latency) that must be respected by the network slice, and that must be properly monitored too. While our previous work considered monitoring challenges for microservice-based VNFs in 5G networks, the current study further advances the management scope by including security concerns and measurements in the evaluated slice. Moreover, the complete network slice itself

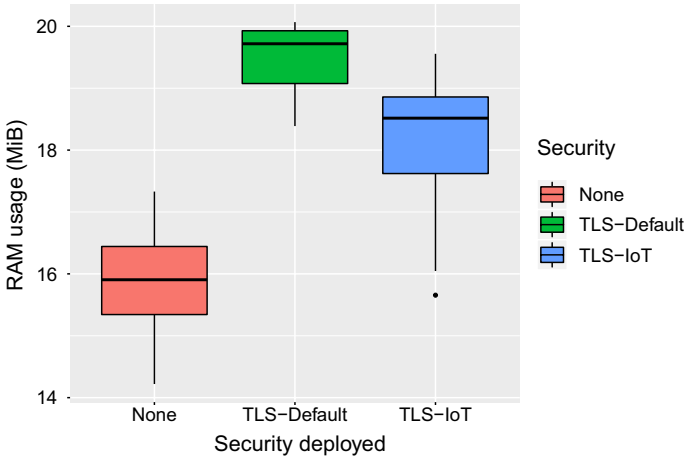

Fig. 5 Computational overhead using different security options

is subject of study here, including different radio access technologies and service applications, with new features and requirements that they come with.

5 Results and Discussion

Although there are already some promising NB-IoT solution in development [33, 36], for stability and reproducibility of our results we do not utilize any specific implementation, and simply consider the traffic patterns for these deployments. With respect to the security demanded in the user's specification, SWEETEN can leverage the previously populated catalogues and produce different results for each requiring application. In the present use case, it is noteworthy that the IoT monitoring devices should try to adopt solutions that incur in minimal overhead because of their resource-constrained nature. Tools and configurations provided by experts to the system's catalogues can therefore feature fine-tuned solutions for systems tagged as such. This way, a TLS configuration using less resource-intensive ciphers can be used for IoT components[3], while more resourceful components can utilize a more robust solution[4]. For comparison purposes, normalized results for the different security options are presented in Fig. 5 for memory footprint, and in Fig. 6 for network overhead.

The results indicate that a small computational overhead is added with respect to memory footprint when either the default or the IoT security solution is included. Albeit small, it is also noticeable that the security option recommended for IoT outperforms the default option with respect to overhead. Similar results are also found

[3] https://docs.aws.amazon.com/iot/latest/developerguide/transport-security.html.
[4] https://www.acunetix.com/blog/articles/tls-ssl-cipher-hardening.

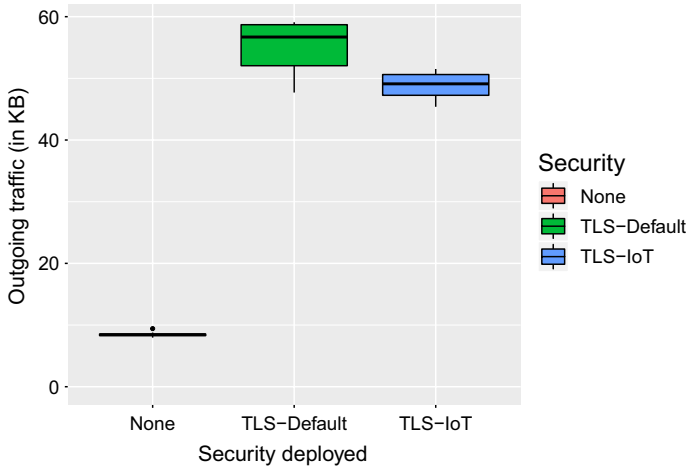

Fig. 6 Network overhead using different security options

for network overhead. While it is clear that the overhead is much more noticeable when comparing either security option (i.e., the default one or the IoT one) to having no security deployed, the IoT configuration still leads to lesser overhead in comparison to the default configuration. While the user should still be mindful that some overhead will be added whenever a security feature is requested, these results show how experts knowledge integrated into the system through different configuration options can be used to produce a more fine-tuned solution for each case.

Another overhead aspect that we consider is the one added to the deployment time of the slice specification. Here, there are two separate types of processing overhead that must be considered. The first one is the overhead introduced by SWEETEN's processing of the user input until the complete solution is produced, as has been explained in-depth in Sect. 3. The second one is the additional deployment overhead due to the inclusion of the management services (realized by containers), that must be instantiated alongside the original specification. A comparison for these times is presented in Fig. 7. We evaluate the system's scalability through varying the number of replicas for each deployment in the slice from one to ten. We divide the time in three steps in order to compare the overhead:

- Base refers to the deployment of the user's network slice without any management functionalities included.
- Complete refers to the deployment of the complete network slice with all management solutions included.
- SWEETEN refers to the overhead added by our prototype processing the user specification and delivering a deployable solution.

The results show that SWEETEN's overhead is approximately constant regardless of the replicas count, and it becomes negligible for larger deployments. Larger deployments are precisely the ones that should benefit the most from SWEETEN,

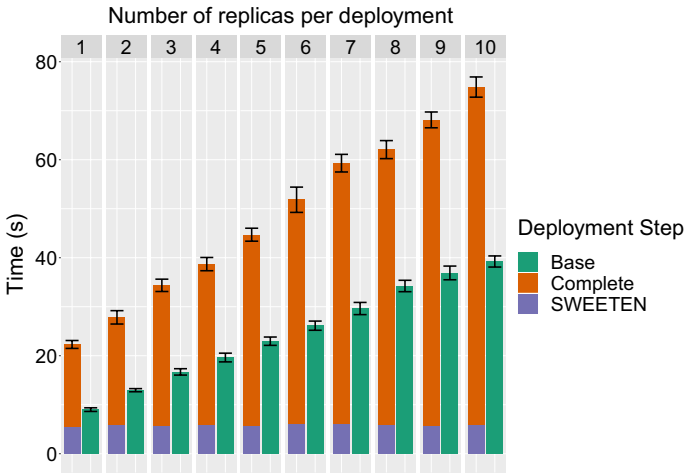

Fig. 7 Deployment time overhead for varying replicas count

as the inclusion of management features throughout a complex slice is burdensome in comparison to a more simplistic slice. Most of the overhead is introduced by the inclusion of the additional containers, with the complete solution taking on average 94% more time to be deployed. Two noteworthy points here are that: (1) this is not a recurring cost, as fresh deployments are less frequent than individual updates, which would present a much lighter overhead; (2) the manual inclusion of management containers by an expert user would incur in similar overhead for the deployment time. Users could minimize this overhead by including the management software directly into their containers, but doing so would breach the microservice architecture and possibly do more harm than good in the process. The overhead results are also aligned to what we observed previously for a different network slice where only monitoring management was featured [7].

We also illustrate how the user receives their monitoring information. An example for an excerpt of the dashboard provided for monitoring the throughput is illustrated in Fig. 8. Through this interface, the user can easily monitor multiple services of their slice and quickly identify problems as they occur. Different resources can be configured by the system with different parameters in a transparent manner for the user. For instance, a resource-constrained device can be configured with a lower sampling rate than a more resourceful device, thus introducing less overhead. The user can also edit the graphics in the dashboard and add their own, so they can fine-tune the solution to best fit their needs.

We also analyze the system performance with respect to the expressiveness offered to the user. About four lines of high-level feature specification by the user is translated to over 30 lines of management deployment specification with respect to security (disregarding the cryptography keys generated and used in the deployment). The result is even more prominent with respect to monitoring, where four lines of specification are translated into over 100 lines of management specification that add the required monitoring features. These findings are inline with what was reported

Fig. 8 Example dashboard for throughput monitoring

in our previous work [7], further reinforcing the benefits offered by SWEETEN for multiple management disciplines.

6 Conclusion and Future Work

5G networks are in the process of being rolled out around the world and will enable disruptive applications and services that were not previously feasible. To realize that, advances presented by NFV, SDN, and network slicing, for example, must be carefully integrated by these networks. With the increasing number of devices and services, network management plays a central role in delivering the resources and features required by each component. For the same reason (i.e., the increasing number of networks components), the configuration and management of all the pieces must be realized in an automated manner.

In this work, we investigate the assisted management of network slices through the use of SWEETEN. Initially proposed as a system to assist VNF operators, SWEETEN has been demonstrated in this study as a tool capable of delivering management solutions across a diverse network slice. Through high-level annotations in their slice specification, users are able to effortlessly receive fine-tailored management solutions configured for each of their applications and services. The proposed use case demonstrates how a network slice for intelligent healthcare can include monitoring and security features with ease, even considering the different requirements for each application.

Our results show that there is an important expressiveness gain for the user through SWEETEN. Assisting the user in properly deploying complex network

slices is a vital point in achieving the dynamism expected from 5G networks. We also evaluated the overhead of SWEETEN with respect to deployment time, and computational and network overhead. While the deployment time is noticeably affected by the additional management services included by SWEETEN, it is not a recurring cost (i.e., the slice's deployment) and is easily offset by the management functionalities featured in the slice. In the same way, computational overhead was non-negligible, but small enough that it is adequate for the services included. Network overhead, however, was much higher when cryptography solutions were included in the system, which, while expected, to users' discretion is needed to define whether the overhead is acceptable or not.

In the future, we plan to further develop the system through the inclusion of additional management disciplines and solutions. A myriad of different services are coming with 5G, and their network requirements can be as varied as the services themselves. It is thus important for a management assistant to be able to cover a variety of cases so that its usefulness is not limited to a small subset of applications. Moreover, we also intend to evaluate different tagging mechanisms for the Features Acquirer module, so that a more refined tagging system can help SWEETEN to further fine-tune solutions and configurations for every service. Finally, we also intend to enrich the deployment options so that management solutions can be deployed to an existing network slice.

Acknowledgements This study was partially funded by CAPES - Finance Code 001. We also thank the funding of CNPq, Research Productivity Scholarship grants ref. 313893/2018-7 and 312392/2017-6.

Author Contributions Rafael Martins wrote the main manuscript text. All authors reviewed the manuscript, discussed the proposal and designed the experiments that make up for the current work.

Funding As per the acknowledgement section, this study was partially funded by CAPES - Finance Code 001. We also thank the funding of CNPq, Research Productivity Scholarship grants ref. 313893/2018-7 and 312392/2017-6.

Data Availability Relevant source code and data can be found at https://github.com/ComputerNetworks-UFRGS/sweeten.

Declarations

Competing interest The authors have no financial or personal competing interests.

Ethical Approval Not applicable.

References

1. Barakabitze, A.A., Ahmad, A., Mijumbi, R., Hines, A.: 5g network slicing using sdn and nfv: a survey of taxonomy, architectures and future challenges. Comput. Netw. **167**, 106984 (2020)
2. ETSI, NFVISG: GS NFV-MAN 001 V1. 1.1 network function virtualisation (NFV); Management and Orchestration. Dec (2014). https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
3. Chowdhury, S.R., Salahuddin, M.A., Limam, N., Boutaba, R.: Re-architecting NFV ecosystem with microservices: state of the art and research challenges. IEEE Netw. **33**(3), 168–176 (2019)

4. Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow, pp. 195–216 (2017). Springer
5. Li, W., Lemieux, Y., Gao, J., Zhao, Z., Han, Y.: Service mesh: challenges, state of the art, and future research opportunities. In: 13th IEEE international conference on service-oiented system engineering (SOSE), pp. 122–127 (2019). IEEE
6. Zhang, S., Wang, Y., Zhou, W.: Towards secure 5g networks: a survey. Comput. Netw. **162**, 106871 (2019). https://doi.org/10.1016/j.comnet.2019.106871
7. de Jesus Martins, R., Dalla-Costa, A.G., Wickboldt, J.A., Granville, L.Z.: Sweeten: Automated network management provisioning for 5g microservices-based virtual network functions. In: 2020 16th international conference on network and service management (CNSM), pp. 1–9 (2020). IEEE
8. Wang, D., Chen, D., Song, B., Guizani, N., Yu, X., Du, X.: From iot to 5g i-iot: The next generation iot-based intelligent algorithms and 5g technologies. IEEE Commun. Magazine **56**(10), 114–120 (2018). https://doi.org/10.1109/MCOM.2018.1701310
9. Xu, J., Yao, J., Wang, L., Ming, Z., Wu, K., Chen, L.: Narrowband internet of things: evolutions, technologies, and open issues. IEEE Internet Things J. **5**(3), 1449–1462 (2018). https://doi.org/10.1109/JIOT.2017.2783374
10. Di Francesco, P., Lago, P., Malavolta, I.: Migrating towards microservice architectures: an industrial survey. In: 2018 IEEE international conference on software architecture (ICSA), pp. 29–2909 (2018). https://doi.org/10.1109/ICSA.2018.00012
11. Foukas, X., Patounas, G., Elmokashfi, A., Marina, M.K.: Network slicing in 5g: survey and challenges. IEEE Commun. Magazine **55**(5), 94–100 (2017). https://doi.org/10.1109/MCOM.2017.1600951
12. Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., Flinck, H.: Network slicing and softwarization: a survey on principles, enabling technologies, and solutions. IEEE Commun. Surv. Tutor. **20**(3), 2429–2453 (2018). https://doi.org/10.1109/COMST.2018.2815638
13. Slamnik-Kriještorac, N., Kremo, H., Ruffini, M., Marquez-Barja, J.M.: Sharing distributed and heterogeneous resources toward end-to-end 5G networks: a comprehensive survey and a taxonomy. IEEE Commun. Surv. Tutor. **22**(3), 1592–1628 (2020)
14. Kist, M., Santos, J.F., Collins, D., Rochol, J., Dasilva, L.A., Both, C.B.: Airtime: End-to-end virtualization layer for ran-as-a-service in future multi-service mobile networks. IEEE Trans. Mobile Comput., (2020). https://doi.org/10.1109/TMC.2020.3046535
15. da Silva Coelho, W., Benhamiche, A., Perrot, N., Secci, S.: On the impact of novel function mappings, sharing policies, and split settings in network slice design. In: 2020 16th international conference on network and service management (CNSM), pp. 1–9 (2020). IEEE
16. Jamshidi, P., Pahl, C., Mendonça, N.C., Lewis, J., Tilkov, S.: Microservices: the journey so far and challenges ahead. IEEE Softw. **35**(3), 24–35 (2018). https://doi.org/10.1109/MS.2018.2141039
17. Lee, S., Levanti, K., Kim, H.S.: Network monitoring: present and future. Comput. Netw. **65**, 84–98 (2014)
18. Stallings, W.: Cryptography and network security, (2006). Springer
19. Enns, R.: Netconf configuration protocol. RFC **2006**, 6241 (2006)
20. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015)
21. Franco, M.F., Rodrigues, B., Scheid, E.J., Jacobs, A., Killer, C., Granville, L.Z., Stiller, B.: Secbot: a business-driven conversational agent for cybersecurity planning and management. In: 2020 16th international conference on network and service management (CNSM), pp. 1–7 (2020). IEEE
22. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
23. de Jesus Martins, R., Hecht, R.B., Machado, E.R., Nobre, J.C., Wickboldt, J.A., Granville, L.Z.: Micro-service based network management for distributed applications. In: 34th international conference on advanced information networking and applications (AINA), pp. 922–933 (2020). Springer
24. Pahl, C.: Containerization and the paas cloud. IEEE Cloud Comput. **2**(3), 24–31 (2015)
25. Ben-Kiki, O., Evans, C., Ingerson, B.: Yaml ain't markup language (yaml™) version 1.1. Working Draft **11** (2009)
26. Bernstein, D.: Containers and cloud: from lxc to docker to kubernetes. IEEE Cloud Comput. **1**(3), 81–84 (2014)
27. Prometheus Authors: Prometheus-monitoring system & time series database. prometheus.io (2017). https://prometheus.io/
28. Series, M.: Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond. Recomm. ITU **2083**, 10 (2015)

29. Sun, Y., Tian, Z., Li, M., Zhu, C., Guizani, N.: Automated attack and defense framework toward 5g security. IEEE Netw. **34**(5), 247–253 (2020). https://doi.org/10.1109/MNET.011.1900635
30. Malik, H., Alam, M.M., Moullec, Y.L., Kuusik, A.: Narrowband-iot performance analysis for healthcare applications. Proc. Comput. Sci. **130**, 1077–1083 (2018). https://doi.org/10.1016/j.procs.2018.04.156
31. Raza, U., Kulkarni, P., Sooriyabandara, M.: Low power wide area networks: an overview. IEEE Commun Surv Tutor **19**(2), 855–873 (2017)
32. Kamiyama, N., Nakao, A.: Analyzing dynamics of mvno market using evolutionary game. In: 15th international conference on network and service management (CNSM), pp. 1–6 (2019). IEEE
33. Beyene, Y.D., Jantti, R., Tirkkonen, O., Ruttik, K., Iraji, S., Larmo, A., Tirronen, T., a. J. Torsner: Nb-iot technology overview and experience from cloud-ran implementation. IEEE Wireless Commun **24**(3), 26–32 (2017). https://doi.org/10.1109/MWC.2017.1600418
34. Wubben, D., Rost, P., Bartelt, J.S., Lalam, M., Savin, V., Gorgoglione, M., Dekorsy, A., Fettweis, G.: Benefits and impact of cloud computing on 5G signal processing: flexible centralization through cloud-ran. IEEE Signal Process Magazine **31**(6), 35–44 (2014)
35. Marotta, M.A., Ahmadi, H., Rochol, J., DaSilva, L., Both, C.B.: Characterizing the relation between processing power and distance between bbu and rrh in a cloud ran. IEEE Wireless Commun. Lett. **7**(3), 472–475 (2018)
36. Ho, C.Y., Cheng, R.G., Chen, J.W., Liu, C.S.: Open nb-iot network in a pc. In: 2019 IEEE Globecom Workshops (GC Wkshps), pp. 1–6 (2019). IEEE

Rafael de Jesus Martins holds a B.Sc. degree in Computer Engineering (2018) and a M.Sc. degree in Computer Science (2022) by Federal University of Rio Grande do Sul (UFRGS). His research interests include softwarized networking and performance monitoring and optimization.

Juliano Araújo Wickboldt is an associate professor at the Federal University of Rio Grande do Sul (UFRGS) in Brazil. He holds both M.Sc. (2010) and Ph.D. (2015) degrees in computer science from UFRGS. Juliano was an intern at NEC Labs Europe in Heidelberg, Germany for one year between 2011 and 2012. In 2015, Juliano was a visiting researcher at the Waterford Institute of Technology in Ireland. His research interests include softwarized networking, IoT, and 5G technologies.

Lisandro Zambenedetti Granville is a full professor at the Federal University of Rio Grande do Sul (UFRGS) in Brazil. He holds both M.Sc. (1998) and Ph.D. (2001) degrees in computer science from UFRGS. Lisandro was a visiting professor at the University of Twente, the Netherlands, for one year between 2007 and 2008. Lisandro was the previous IEEE CNOM chair and IETF'S NMRG co-chair. His research interests include network programmability, virtualized networks, and network monitoring/measuring.