

---

# Towards the autonomous provision of self-protection capabilities in 5G networks

Alberto Huertas Celdrán · Manuel Gil  
Pérez · Félix J. García Clemente ·  
Gregorio Martínez Pérez

**Abstract** 5G mobile networks are pushing new dynamic and flexible scenarios that demand the automation and optimization of network management processes. In this sense, Self-Organizing Networks (SON) arose to evolve from traditional manual management towards fully autonomic and dynamic processes. Due to the large volumes of data generated in 5G networks, functionalities and capabilities of SON require efficient processes and resource optimization techniques. In particular, self-protection is a critical capability of SON focused on protecting the network resources in a flexible and autonomic way. To achieve self-protection, SON perform different processes ranging from the monitoring of network communications to the analysis, detection, and mitigation of cyber-attacks. In this article, we propose an architecture that combines the Software Defined Networking (SDN) and Network Functions Virtualization (NFV) technologies to optimize the usage of network resources for monitoring services. A use case based on botnet detection in 5G networks shows how our architecture ensures the provision of monitoring services in managing self-protection scenarios. Additionally, we describe a set of experiments that confirm the best time calculated by our solution to deploy or reconfigure

---

Alberto Huertas Celdrán  
Departamento de Ingeniería de la Información y las Comunicaciones, University of Murcia,  
30071 Murcia, Spain  
Tel.: +34 868 884644  
Fax: +34 868 884151  
E-mail: alberto.huertas@um.es

Manuel Gil Pérez  
Departamento de Ingeniería de la Información y las Comunicaciones, University of Murcia,  
30071 Murcia, Spain  
E-mail: mgilperez@um.es

Félix J. García Clemente  
Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, 30071 Murcia, Spain  
E-mail: fgarcia@um.es

Gregorio Martínez Pérez  
Departamento de Ingeniería de la Información y las Comunicaciones, University of Murcia,  
30071 Murcia, Spain  
E-mail: gregorio@um.es

monitoring and detection services. These experiments consider different aspects like the number of zombies shaping the botnet, their mobility, or network traffic.

**Keywords** Network monitoring · Software Defined Networking · Virtualization · Botnets · 5G mobile networks

## 1 Introduction

The evolution of technologies has brought about a radical change in mobile networks and, therefore, in their internal management processes. Nowadays, the incoming fifth generation (5G) of mobile networks is advancing new scenarios in which dynamism and flexibility are essential aspects. These new scenarios are characterized by considering several Key Performance Indicators (KPIs) [1], defined by the 5G Public Private Partnership (5G-PPP). Among these indicators, the number of connected devices (from 10 to 100 times more than 4G/LTE), the volume of mobile data per geographical area (1000 times higher), the end-to-end latency (less than 1ms), and ubiquitous 5G access including low density areas are some of the most relevant aspects that influence the evolution of current networks towards future mobile networks strengthened with the 5G technology.

This new situation requires the automation of management processes currently performed by network administrators. Because of this, Self-Organizing Networks (SON) arose with the goal of moving from traditional manual management towards fully autonomic and dynamic processes without human intervention. These new autonomic aspects provide SON with self-managing capabilities. To reduce network management complexity, the *Software Defined Networking* (SDN) paradigm [2] can help SON to automatically manage and orchestrate the network resources by taking into account the situational awareness of the underlying network given at any time. Specifically, this paradigm enables the flexible and dynamic management of the network communication. It is crucial to allow SON to control the communications from/to their infrastructure, services, and users in real time. To this end, the SDN paradigm establishes three well-defined characteristics:

1. The ability to decouple the *data plane*, where forwarding elements are located, from the *control plane*, where routing decisions are made.
2. The unified control element called *SDN Controller*, which manages multiple network elements belonging to the data plane.
3. The global administration perspective, which avoids making changes on each individual network element.

Furthermore, combining the SDN paradigm with Network Functions Virtualization (NFV) techniques [3] allows decoupling the software implementation of Network Functions (NF) from the underlying hardware, providing and enhancing the flexibility on the management of the network resources.

One of the most challenging tasks of SON is the management of the self-protection capability due to the large volumes of data generated from 5G subscribers' User Equipments (UE), according to the KPIs established by the 5G-PPP. Self-protection is a critical aspect of SON in the latter's protection of the network resources the network resources (infrastructure and services) in a flexible and autonomic way. To achieve self-protection, SON perform different processes ranging from the monitoring of network communications to the analysis,

detection, and mitigation of cyber-attacks. Monitoring services need to provide valuable information in real time about resources and the actual network status, which are essential to fulfill the main functionalities and capabilities of SON: self-configuration, self-healing, self-optimization, and self-protection [4]. In this sense, the orchestration of monitoring and sensing services to achieve the self-protection capability is a critical and complex process that should be carried out in an automatic way. Otherwise, in the case of self-protection scenarios -the focus of the current contribution-, the detection of potential attacks would not be possible due to the huge number of UEs consuming network services, their high mobility between different Radio Access Networks (RAN), and the bandwidth and latency of future mobile communications, as expected for 5G networks, among others factors.

Despite the advantages provided by the NFV and SDN approaches, the mobility provided by future mobile networks as well as the dynamic provision of services have created new complications in the process of achieving the efficient management of network infrastructures. In addition, we believe that these future networks should orchestrate network monitoring services by taking into consideration not only Data-Related Information (DRI), such as the information contained in network flows, but also the Control-Related Information (CRI) to optimize the use of resources in the monitoring processes. Aspects belonging to the SDN and NFV control plane operations, such as the number of gathered flows per second or the percentage of CPU and storage consumed by network resources at a given time are some of the essential keys to ensure the correct provision of monitoring network services.

In order to face the previous challenge, the current article substantially extends the solution proposed in [5]. In that contribution, an architecture was presented to autonomically orchestrate monitoring services of a 5G-oriented network, managing network resources with information that belongs to the network control plane, CRI. This extension has two main contributions:

1. The definition of a self-protection use case that shows how the proposed architecture autonomically manages monitoring services to proactively detect potential attacks conducted by botnets in a 5G scenario.
2. The design and conduction of different experiments with the aim of demonstrating the usefulness of our architecture. The experiments performed indicate the best moment to deploy or reconfigure monitoring and detection services in a given scenario. Among the different aspects considered by those experiments, we highlight the number of bots shaping the botnet, their mobility, the traffic generated by them, and the available network resources together with their capabilities when running in a 5G-oriented network.

The remainder of the article is structured as follows. Section 2 discusses related work on other SDN-oriented solutions that monitor and orchestrate network elements as well as the process made by our solution to enforce the actions required to manage network resources. Section 3 shows the components that make up the proposed architecture, while Section 4 presents the self-protection use case that will be used throughout the article to demonstrate how the architecture manages monitoring services autonomically to detect botnet attacks. The management policies and how they manage the virtual and physical network elements are presented in Section 5. Section 6 shows the experiments performed so as to test the useful-

ness of our solution. Finally, conclusions are drawn and future work is suggested in Section 7.

## 2 Related work

Data security and privacy is a critical aspect in the self-protection capability of SON [6]. In this sense, in recent years, different approaches have been proposed according to different approaches such as cryptography [7], secure attribute-based data sharing [8], and network communications. Focusing on the last aspect, the Software-Defined Networking (SDN) is a recent paradigm developed to facilitate network management in diverse scenarios such as IoT [9] and industry [10], by decoupling control and data planes [2]. In the last few years, many efforts have been made to provide dynamism to network management through the SDN paradigm, which is showcased in a recent review of this paradigm [11]. This review focuses on the current research status of multi-domain SDN implementations as well as its future challenges. Among the diversity of SDN-oriented proposals, monitoring services are crucial for many network management tasks, such as load balancing, traffic engineering, Service Level Agreement (SLA), and security, among others. In this sense, OpenNetMon [12] is a POX OpenFlow controller that monitors all flows between predefined link-destination pairs on throughput, packet loss, and delay. By querying the switch about the number of bytes sent as well as the duration of each flow, OpenNetMon is able to calculate the effective throughput per flow. To this end, this solution compares the flow statistics to compute the packet loss; particularly, the packet counters from the first and the last switch of each path between the link-destination pairs. Delay is measured by injecting probe packets directly into the switch data planes, and determining a realistic delay for each flow. On the other hand, PayLess [13] is an efficient network statistics collector framework for the SDN paradigm. It is built on top of OpenFlow controllers to monitor, aggregate, and select information gathered from the switches belonging to the data plane, in accordance with the high-level requirements expressed by applications. Furthermore, this solution provides a flexible RESTful API for flow statistics collection at different aggregation levels. It uses an adaptive statistics collection algorithm that delivers highly accurate information in real time, without significant network overhead.

It is worth noting that the previous proposals are oriented to monitor the DRI of the network, without taking into account the management of network resources autonomically in cases, for example, of network congestion or when a given service cannot be offered. In order to improve this situation, the solution presented in [14] proposes management to monitor, visualize, and configure the elements belonging to the three planes of the SDN paradigm. SDN-specific metrics are considered by network administrators to make decisions and configure/reconfigure SDN-related parameters according to their needs. Another approach aimed at controlling the SDN paradigm was presented in [15]. Specifically, it proposes a SDN-based load balancing solution, in which flow capacity is defined by the number of requested physical resource blocks. The results showed a drastic reduction of the number of unsatisfied users in the network and a substantial improvement of resources allocated per user.

The solutions outlined above are able to control resources in SDN-oriented networks in real time, but without considering virtualization techniques and the flexibility obtained when the software implementation of NFs is decoupled from the underlying hardware. In this sense, Software-Defined Network Virtualization (SDNV) [16] is a framework that integrates the SDN and NFV techniques. This considers the SDN principle of separating data and control planes with NFV, decoupling service functions from infrastructures. Furthermore, key technical challenges to make SDN and NFV integration are discussed in this proposal. Following the SDN and NFV integration approach, another solution [17] proposed a management and orchestration architecture for multi-tenant transport networks, which allows deploying virtual optical transport networks and virtual SDN Controllers as Virtualized Network Functions (VNF) in data centers. One of the main challenges addressed by this solution consists in integrating the orchestration of distributed cloud and network resources to dynamically deploy virtual machines and VNF instances to supply the required network connectivity.

On the other hand, regarding solutions devoted to detecting and mitigating attacks in the cyber security space, several approaches are proposing new ways of dealing with the challenges of 5G networks need to face. For example, a list of potential cyber threats in current networks is presented in [18], including attacks that could also be used to disrupt the operation of future 5G networks. The authors note that botnets, especially the mobile ones, are expected to continue to exist in 5G mobile networks as one of the most powerful cyber threats nowadays [19]. In this context, the work presented in [20] claims that SDN and NFV technologies are becoming key enablers for 5G networks. As an example, BotGuard is presented in [21] as a framework for real-time botnet detection in SDN-enabled network. BotGuard uses a Convex Lens Imaging (CLI) graph to extract botnets' topology characteristics with the most significant features so as to detect Command and Control (C&C) channels. Finally, the works presented in [22] and in [23] propose the tightly combination of both NFV and SDN technologies to provide effective detection and mitigation of cyber-attacks in 5G networks.

Despite the important progress made by the previous solutions, they manage the SDN and NFV resources considering only information related to the data plane, DRI. In this sense, it is highly required more efforts focused on considering also information belonging to the control plane. Missing this important plane means that autonomic solutions are not able of ensuring critical capabilities of SON like self-configuration, self-protection, self-optimization, and self-healing. In this context, to ensure the provision of the monitoring service (critical for the self-protection capability), we believe that it is critical to consider also information related to the control plane, CRI. This section has demonstrated that nowadays, there are no solutions integrating SDN and NFV technologies to manage and orchestrate network resources' behavior by considering DRI and CRI. The proposed solution fills this gap, monitoring CRI to orchestrate the behavior of network elements belonging to the SDN and NFV technologies. This will then allow strengthening critical capabilities of 5G mobile networks, enhancing detection and mitigation procedures in case of self-protection scenarios like the one proposed below.

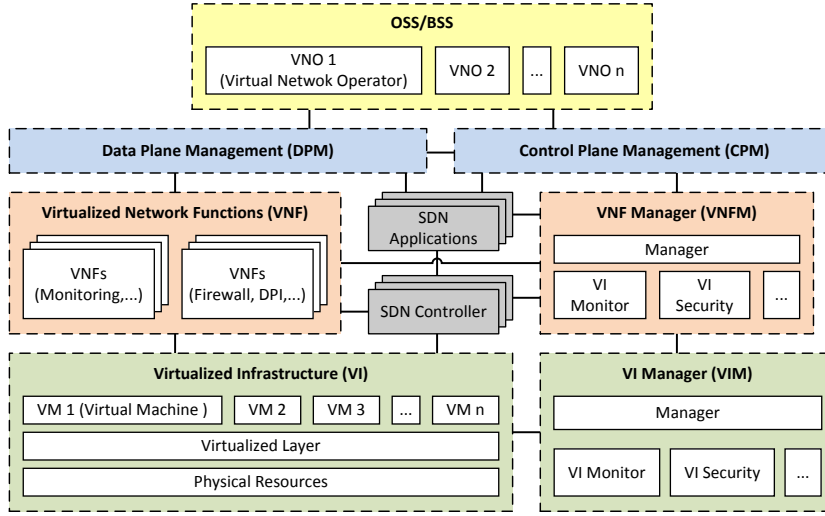


Fig. 1 Architecture to manage the network resources efficiently

### 3 5G-oriented architecture for control-related information

In this section, we propose an architecture oriented to 5G networks [24] that integrates the SDN paradigm with the ETSI NFV proposal [25]. This architecture autonomically manages network resources with the goal of ensuring the provision of network monitoring services. Monitoring services are essential to detect cyber-attacks of potential botnets in self-protection scenarios. Network monitoring services are deployed in our solution as VNF Monitoring, and are characterized by gathering information from different and heterogeneous sources; for example, network infrastructure (physical or virtual), network management services, and communications between users and network services. Fig. 1 shows the five layers that make up our architecture: Virtualized Infrastructure (VI); Virtualized Network Functions (VNF); Software-Defined Networking (SDN) paradigm; Data and Control Plane Management (DPM and CPM); and Operations and Business Support Systems (OSS/BSS).

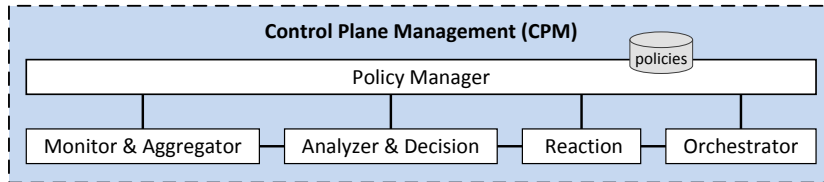
At this point, it is important to highlight that one of the novel contributions of the proposed architecture is the CPM layer (explained below in detail). The components and interfaces of the CPM are in charge of acquiring specific data, analyzing it, making decisions according to predefined policies, and orchestrating those decisions to ensure the provision of the monitoring services. The proposed architecture uses virtualization techniques, provided by the ETSI NFV architecture, to enforce the decisions made by the CPM layer and ensure the provision of the monitoring service in a flexible and cost-efficient way. Therefore, the integration between the CPM layer, the ETSI NFV proposal, and the SDN paradigm is also a novel contribution. In summary, the SDN paradigm and the NFV technologies simply provide standard mechanisms to enforce the decisions made by the CPM control layer in a dynamic and flexible way. By taking into account the previous information, there is a lack of standardization mechanisms in the acquisition, analysis, detection, and reaction processes.

From bottom to top of Fig. 1, the *Virtualized Infrastructure Manager* (VIM) is in charge of creating, controlling, and monitoring the whole life-cycle of *Virtual Machines* (VM), instantiated on generic *Physical Resources* through the *Virtualized Layer*. On top of the VIM, the *VNF Manager* (VNFM) is able to create, manage, monitor, and dismantle VNFs running on VMs located at the VI layer. VNFs are services oriented to assist in the network management tasks. Examples of VNFs can be monitoring services, Intrusion Detection Systems (IDS) deployed as Deep Packet Inspection (DPI) tools, or firewalls, among others. At the same level as the VNF, but in another layer, the SDN paradigm decouples the control plane from the data plane. Thus, the SDN Controller and the SDN Applications are able to monitor and manage the control plane of physical and virtual network resources.

In order to manage the information considered by the previous layers, our architecture proposes a management layer with two planes: data and control. This proposal is oriented to the *Control Plane Management* (CPM), which handles CRI provided by SDN Applications and from the virtualization managers (VNFM and VIM). The *Data Plane Management* is outside the scope of this work, which manages DRI received from the data plane of SDN Applications as well as from existing VMs and VNFs. Regarding the control plane, there are several components in charge of monitoring and aggregating CRI, analyzing and making decisions to ensure the provision of VNFs, and reacting and orchestrating the decision-making results according to a given set of internal rules and policies defined by the *Virtual Network Operators* (VNO) belonging to the *OSS/BSS* layer.

### 3.1 The Control Plane Management to handle CRI

In order to show how the proposed architecture is able to ensure the autonomic provision of the VNF Monitoring, Fig. 2 shows in detail the internal communications between the different components belonging to the CPM.



**Fig. 2** Components shaping the Control Plane Management

Specifically, the *Monitor & Aggregator* component gathers CRI from:

- The **VNFM**. This manager provides metrics with the status of the VNF Monitoring (e.g., the number of network flows gathered per second).
- The **SDN Applications**. They send CRI about the status of elements in the three planes of the SDN paradigm. The number of flow entries stored in each switch, the number of SDN Applications running in the SDN Controller, and the logical/physical/geographical location of a given application or network

resource could be examples of CRI gathered by the Monitor & Aggregator component.

- The **VIM**. This component provides CRI about the status of the virtual and physical infrastructure in which the VNF Monitoring is running (e.g., the percentage of CPU and memory used by physical and virtual machines).

Once the previous CRI has been gathered, this is aggregated and sent to the *Analyzer & Decision* component to be analyzed and decide the actions needed to ensure the VNF Monitoring provision. For this decision-making procedure, the Analyzer & Decision considers internal management policies (discussed in Section 5) as well as high-level policies defined by VNOs to set specific thresholds used by management policies. Both kinds of policies are provided to the Analyzer & Decision through the *Policy Manager*. Next, we define a set of possible actions to ensure the VNF Monitor provision made by the Analyzer & Decision:

- Regarding the VNF layer, our solution is able to virtualize a new VNF Monitoring or configure internal parameters of an existing one to guarantee the service provision.
- In the SDN layer, we can indicate when SDN Applications should add, modify, or delete flow entries in the switches to balance or filter the network traffic. Moreover, we can also indicate specific actions of the SDN Applications oriented to the control plane of given network elements (e.g., firewalls or IDSs).
- Regarding the VI layer, our solution indicates when it is required to create new VMs with VNF Monitoring capabilities or add virtual resources like memory, storage, or computation power to existing VMs.
- In the physical layer, actions like switching on/off physical resources to guarantee the VNF Monitoring.

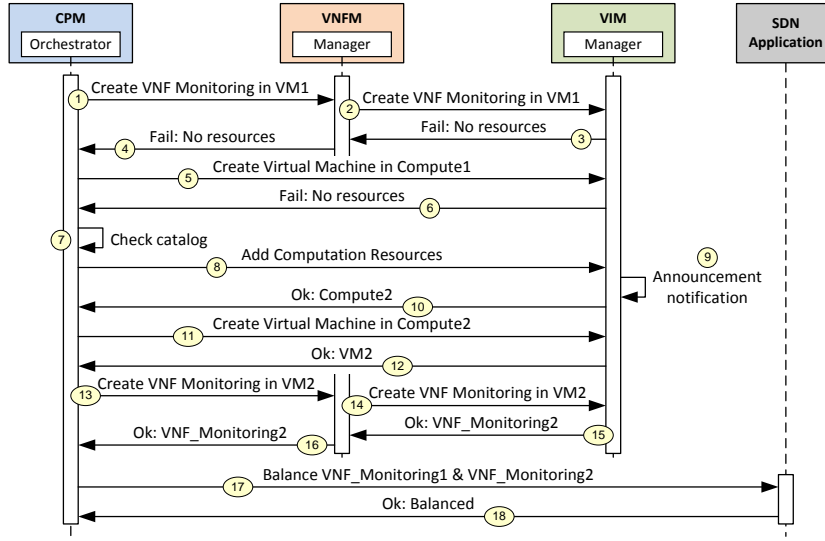
The *Reaction* component in Fig. 2 is in charge of verifying that the actions previously decided to be performed can be applied without provoking conflicts with the existing ones. Moreover, when there are several actions to be performed, the Reaction component is in charge of establishing different levels of priority for them. Finally, the *Orchestrator* knows the components responsible of applying the different actions and it communicates with them (VNF, VIM, and SDN Applications) to enforce those actions.

### 3.2 Orchestration to manage monitoring network services

This section shows the steps performed by the Orchestrator of our architecture in an autonomic way to deploy a new VNF Monitoring. As Fig. 2 shows, once it receives new actions to be enforced, as indicated in the previous section, it interacts with the different components belonging to the SDN, VNF, and VI layers for their actual enforcement.

In order to show how the Orchestrator performs the different steps associated to a given action, we defined a simplistic virtualized scenario containing a SDN-oriented network to monitor the network traffic by means of a VNF Monitoring (VNF\_Monitoring1), which is running on a virtual machine (VM1) deployed on top of the physical infrastructure (Compute1). At an initial stage, the network is providing VNF\_Monitoring1 to ensure the quality of service (QoS). However, at a given time the number of Monitored Flows Per Second (MFPSs) of





**Fig. 3** Diagram of sequence showing the interaction between the Orchestrator and the SDN/NFV resources

VNF\_Monitoring1 starts increasing until it exceeds a given threshold. This situation brings about congestion in VNF\_Monitoring1, which is detected by the Analyzer & Decision component thanks to the policies presented in Section 5 below. As a result, the proposed solution could decide to keep providing the service by creating a new VNF Monitoring (VNF\_Monitoring2).

With this scenario in mind, Fig. 3 shows the different steps performed by the Orchestrator and the SDN/NFV resources to create the new VNF\_Monitoring2. To this end, the Orchestrator interacts with the VNFM to create VNF\_Monitoring2 in the existing virtual machine, VM1 (Step 1 in Fig. 3). The VNFM then communicates with the VIM to deploy the VNF (Step 2), finding out that VM1 has not enough resources when the VIM returns a fail message (Steps 3 and 4). Once the Orchestrator receives the message, it directly asks the VIM to deploy a new virtual machine (VM2) in Compute1 (Step 5). The VIM checks that the available physical resource (Compute1) is not sufficient to instantiate VM2, and it notifies the situation (Step 6). The Orchestrator then checks the catalog maintained by the VIM, so as to find out if there are more available physical resources to extend the existing one. If so, the Orchestrator notifies the VIM that it is required to add new computation resources (Compute2), being it in charge of switching on Compute2 (Step 9).

Once the physical resources have been extended with Compute2 (Step 10), the Orchestrator communicates with the VIM to instantiate VM2 in Compute2. When completed, the Orchestrator catalog is updated (Step 12) and communicates with the VNFM to deploy and configure VNF\_Monitoring2 (Step 13). Finally, the Orchestrator notifies the SDN Application that it should balance the workload between the two VNF Monitoring instances, modifying the flow tables of the existing switches for that purpose (Step 17).

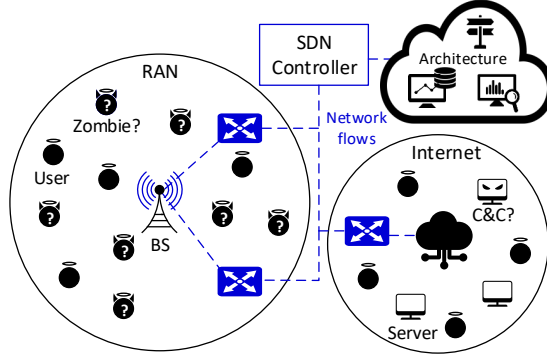
#### 4 Self-protection use case

This section describes a use case that points out the necessity of autonomically managing the monitoring services deployed by our 5G-oriented architecture in real time. In particular, the proposed use case shows the importance of having a flexible and efficient architecture able to ensure the provision of the monitoring service to detect cyber-attacks conducted by botnets in a proactive way [26]. In 5G networks, the detection and mitigation of botnets present an even greater challenge due to the massive number of connected devices and higher data rate. Botnets have been recognized by Joseph Demarest, director of the FBI cyber area, as one of the most powerful threats on Internet. He stated in [27] that “each second, 18 computers are recruited by botnets” (around 567 million of computers compromised annually) estimating losses around 110,000 million dollars per year worldwide.

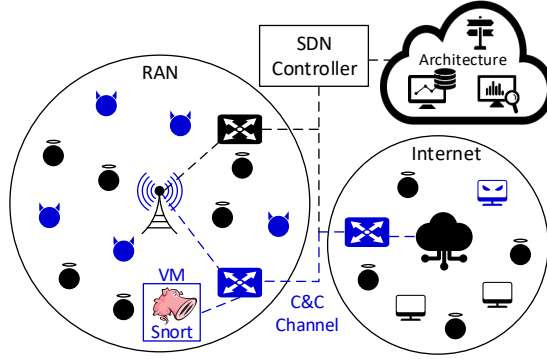
To this end, we propose a use case based on decoupling traditional botnet detection procedures into two phases at two complementary levels of abstraction. The initial phase carried out by our architecture is a high-level detection phase in charge of identifying suspicious C&C channels established between a given C&C server and its recruited zombies or bots. These channels are established by the C&C server with the goal of controlling the zombies and indicating when they have to initiate a given attack. In this first phase, our architecture makes use of the SDN paradigm to monitor and analyze traffic network flows, in order to detect suspicious C&C channels very quickly. In 5G networks, where the number of packets traveling through the network infrastructure is 1000 times higher than in current networks, it is not possible to make DPI at this level. Because of this, monitoring and detection tools like Snort [28] are not able to analyze the sheer number of network packets managed by 5G networks. It should be borne in mind that we use Snort as a DPI tool because it is the only well-known DPI tool supporting GTP (GPRS Tunneling Protocol) encapsulation traffic belonging to 5G networks. Fig. 4 shows a basic scenario of this initial phase, which comprises a Radio Access Network (RAN) where several users and suspicious zombies are connected to the Internet through a base station (BS1) and several routing elements (switches). Additionally, the figure also shows the Internet side where different servers, included suspicious C&C servers, provide network services. In this scenario the proposed architecture is in charge of monitoring the network flows thanks to the elements in blue in the figure.

Once the first phase observes suspicious C&C channels, our use case launches a second low-level detection phase in charge of confirming that the suspicious C&C channels are real and a given botnet is then in place. To confirm this, our architecture dynamically deploys and configures a Snort VNF in real time by using the NFV techniques explained in Section 3. In addition, thanks the SDN paradigm the network traffic between suspicious zombies and the C&C server is mirrored to the Snort VNF. In this case, the Snort VNF is effective due to the lower number of network packets to monitor and analyze (only those belonging to the C&C channels previously identified as suspicious). Fig. 5 represents the phase when the proposed architecture deploys a VM with an Snort VNF to make a deep package inspection and detect the existing zombies and C&C servers (in blue).

Up to this point, we have seen the importance of having a flexible and efficient architecture able to manage the network resources and ensure the provision of monitoring services. In this context, this section has shown different actions took



**Fig. 4** Initial phase to make a high-level detection of suspicious zombies



**Fig. 5** Second phase to confirm suspicious C&C channels between real zombies and the C&C

by our architecture during the two phases of the proposed use case. However, it is also important to define an autonomic mechanism in charge of deciding the exact moment when it is necessary to deploy a new Snort VNF or reconfigure the existing ones. In this sense, Section 5 shows the policies managed by our architecture in charge of defining *what*, *when*, *where*, and *how* to manage the VNF Monitoring autonomically.

## 5 Policies to ensure proper provision of monitoring network services

This section shows how the proposed architecture uses policies to make decisions about the network infrastructure and communications. These actions will ensure the provision of monitoring services in the self-protection use case. Among the different existing policies, we use management-oriented policies, which are defined in the Policy Manager component of the architecture presented in Section 3. These policies decide the list of potential actions to be taken to guarantee the VNF Monitoring provision, in accordance with dynamic contextual information. Regarding the previous use case oriented to detect botnets, contextual information could be the number of suspicious zombies or C&C appearing in our 5G-network, the mo-

bility of zombies around different RANs, the overloading of existing Snort VNFs due to the number of suspicious zombies, their traffic, etc.

Policy actions influence the behavior of the components and layers of the proposed architecture. Considering this influence, we categorize our management-policies into four different groups, which are detailed in the following subsections; namely: Software-Defined Networking, Virtual Network Function, Virtual Infrastructure, and Physical Infrastructure policies. These policies are comprised of the following elements: the *type* of policy; the network *resource*, whose information is currently being managed; the *metric* with which the network resources are evaluated; the *location* or region where the policy will be enforced; the *date* or period of time in which the policy will be applied; and the *result* or set of actions to be carried out on the network once the policy is applied. The previous elements are shown in the next policy structure:

$$Type \wedge Resource \wedge Metric \wedge Location \wedge Date \rightarrow Result$$

The previous categorization of policies as well as the elements comprising the policies have been defined to be extensible. This extensibility enables the adaptability of the proposed solution to cover other open challenges of SON such as mobility, QoS, or high availability. In order to reach the desired extensibility, we use technologies based on semantic web. On the one hand, we use ontologies to represent formally the information considered by the policies. On the other hand, the policies are modeled using semantic rules that consider the elements shaped by the ontologies.

Policies have different actions according to the family to which they belong. However, the reaction of a given policy can affect different elements belonging to the network infrastructure. For example, Step 1 of Fig. 3 shows the moment when a Virtual Network Function Policy decides that it is necessary to deploy a new VNF. At that point, this decision implies not only the deployment of a new VNF, but also a new Virtual Machine (where the VNF will run) and add new physical resources (where the VM will be hosted). In this kind of situations, the orchestrator of the proposed architecture is responsible for knowing the specific action associated with policies decisions. For that, the Orchestrator interacts with the different managers (VNFM, VIM, SDN Controller), and maintains several catalogs with the status of the network scenario.

### 5.1 Software-Defined Networking policies

The Software-Defined Networking (SDN) policies allow managing the elements belonging to the SDN paradigm dynamically. By considering the status of the VNF Monitoring, users' mobility, or the location of the infrastructure, this kind of policies manages automatically the network resources belonging to the data, control, and application planes of the SDN paradigm.

One of the most important and useful tasks of the SDN paradigm consists in controlling, in real time, the forwarding of packets traveling across the network infrastructure. The following SDN policy shows an example oriented to manage the control plane of switches belonging to the data plane of the SDN paradigm.

$$\text{Type}(\#SDN) \wedge \text{Resource}(\text{?switch}) \wedge \text{isConnected}(\text{?switch}, \text{?vnfMonitoring}) \wedge$$

$$\text{integer}[\text{MFPS in } \#MFPSRedAlert] \text{ hasStatus}(\text{?switch}) \wedge \text{hasLocation}(\text{?switch}, \text{?area}) \wedge$$

$$\text{locatedResources}(\text{?area}, \text{?nearSwitches}) \rightarrow \text{balance}(\text{?switch}, \text{?nearSwitches})$$

This policy modifies the flow entries of any congested switch that provides network statistics to a given VNF Monitoring. That congestion means that the number of MFPSs is above a predefined threshold called *MFPSRedAlert*. This situation needs to balance the network traffic between the close switches in order to guarantee the provision of network statistics to the VNF Monitoring. The *metric*, *location*, and *date* parameters are optional in this kind of policies.

In our self-protection use case, the control of the network communications is of great importance in order to detect C&C channels. During the first phase of the use case, network packets are mirrored to the Snort VNF in order to make a deep packet inspection. Once the Snort VNF detects that there is a real zombie and the corresponding C&C server, we can block the channel dropping their packets through the SDN Controller and the network switches.

## 5.2 Virtual Network Function policies

Through the Virtual Network Function (VNF) policies, our proposal is capable of managing and configuring the internal behavior of the VNFs dynamically, with the goal of guaranteeing the VNF Monitoring provision. Furthermore, these policies allow instantiating/dismantling VNFs to migrate or balance the workload of monitoring services and optimize their use.

As an example of VNF policies, the following policy indicates that when a given VNF Monitoring (*resource*) is congested (e.g., the percentage of CPU *metric* –CPUPct– is above a predefined threshold *CPUPRedAlert*), the *result* consists in creating a new VNF Monitoring whose *location* is close to the congested one.

$$\text{Type}(\#VNF) \wedge \text{Resource}(\text{?vnfMonitoring}) \wedge \text{integer}[\text{CPUPct in } \#CPUPRedAlert]$$

$$\text{hasStatus}(\text{?vnfMonitoring}) \wedge \text{hasLocation}(\text{?vnfMonitoring}, \text{?area}) \rightarrow$$

$$\text{createVNF}(\#Monitoring, \text{?area})$$

Oriented to the self-protection use case, these policies also allow our architecture to configure the existing Snort VNFs when there are new suspicious C&C channels. In this sense, Section 6 describes different experiments performed with Snort to find out the time required to add a new rule by taking into account the users' mobility and the number of suspicious C&C channels managed by a Snort VNF.

The next policy configures an existing Snort VNF (*snortVnf*) with a new detection rule when a suspicious C&C appears and *snortVnf* is not overloaded. To ascertain where the overload threshold is, we have performed several experiments, which are presented and explained in Section 6. Specifically, Fig. 9 shows how the number of detection rules accepted by Snort –NCCchannel– affects the Snort performance. In this sense, according to the output of our experiments, we have established that 100 detection rules is the overload threshold of Snort.

$$\text{Type}(\#VNF) \wedge \text{Resource}(\text{?snortVnf}) \wedge \text{integer}[\text{NCCchannel under } \#100] \\ \text{hasStatus}(\text{?snortVnf}) \wedge \text{hasLocation}(\text{?snortVnf}, \text{?area}) \wedge \text{hasC\&C}(\text{?area}, \text{?zombie}) \rightarrow \\ \text{configureSnort}(\text{?snortVnf}, \text{?zombie})$$

### 5.3 Virtual Infrastructure policies

The Virtual Infrastructure (VI) policies are aimed at managing the virtual network infrastructure automatically, in order to ensure the provision of a given VNF Monitoring. Among possible actions enforced by this kind of policies, we highlight several of them like the creation/dismantling of virtual resources (e.g., computation, networking, storage, etc.), instantiation/termination of VMs, and even the relocation of virtual resources in existing VMs, among others.

The following VI policy, as an example, shows that when a given VM, in which a VNF Monitoring is running, is congested (the percentage of used memory –MemoryPct– is above a *MemoryRedAlert* threshold), the reaction consists in instantiating a new VM located close to the congested one to later migrate or create a new VNF Monitoring with a VNF policy.

$$\text{Type}(\#VI) \wedge \text{Resource}(\text{?vm}) \wedge \text{hasResource}(\text{?vnfMonitoring}, \text{?vm}) \wedge \\ \text{integer}[\text{MemoryPct in } \#MemoryRedAlert] \text{ hasStatus}(\text{?vm}) \wedge \text{hasLocation}(\text{?vm}, \text{?area}) \\ \rightarrow \text{instantiateVM}(\#MonitoringVM, \text{?area})$$

Regarding the self-protection use case, Virtual Infrastructure policies allow our solution to deploy new VMs in specific locations that host Snort VNF when the existing ones are overloaded or far away from the zombies' locations. Section 6 shows different experiments performed with Snort to discover the best moment to deploy a new VM with a Snort VNF by taking into account the number of suspicious C&C channels that it is able to monitor in a given moment, as well as the amount of traffic generated by these channels, and the zombies' movements.

According to the results obtained in our experiments, the next policy deploys a new VM with a Snort VNF in the same area as the existing one (snortVnf) when the number of detection rules managed by Snort (channels monitored and analyzed) is more than 100 (so it is overloaded).

$$\text{Type}(\#VI) \wedge \text{Resource}(\text{?snortVnf}) \text{ integer}[\text{NCCchannel over } \#100] \\ \text{hasStatus}(\text{?snortVnf}) \wedge \text{hasLocation}(\text{?vm}, \text{?area}) \rightarrow \text{instantiateVM}(\#SnortVM, \text{?area})$$

### 5.4 Physical Infrastructure policies

The Physical Infrastructure (PI) policies allow controlling the physical resources belonging to the network infrastructure in order to ensure the provision of a VNF Monitoring. The actions of these policies focus on adding or removing physical resources such as processors, memory, disk, network interfaces, etc. These policies are used when it is necessary to add or remove physical resources required to accomplish the demand of the monitoring service in a proactive way. Usually,

these actions focus on ensuring the QoS or availability of the monitoring services provided by the self-protection scenario.

As an example, the following policy indicates that when a VM is congested, the reaction implies to switch on a new physical machine located close to the congested one so that it (the machine) can later use the previous policies to ensure the VNF Monitoring.

```
Type(#PI) ∧ Resource(?physicalMachine) ∧
hasResource(?vnfMonitoring,?physicalMachine) ∧
integer[StoragePct in #StorageRedAlert] hasStatus(?physicalMachine) ∧
hasLocation(?physicalMachine,?area) → switchON(#MonitoringPM,?area)
```

## 6 Performance results

In this section, we report on the results of several experiments conducted with the aim of testing the usefulness of our solution. These experiments were intended to allow our proposal to change the network configuration in the best moment by taking into account the following key aspects:

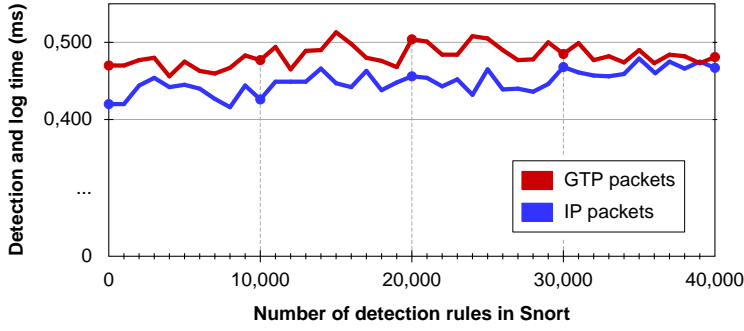
- Overload of Snort when managing GTP as opposed to IP packets.
- Start-up time of Snort for its setting in GTP and IP detection.
- How zombies' mobility and CRI affect the number of deployments and configurations of Snort.

As experimental setting, the tests were conducted in a dedicated PC with an Intel Core i7-3770 3.40 GHz, 16 GB of RAM, and an Ubuntu 12.04 LTS as operating system. The results shown in this section have been obtained by executing the experiments 100 times and computing their arithmetic mean.

### 6.1 Test 1: Overload of Snort when managing GTP as opposed to IP packets

The first experiment focuses on checking the overload involved in introducing a new type of network traffic to a DPI tool for its deep analysis. In particular, GPRS Tunneling Protocol (GTP) is an Internet Protocol (IP) based-protocol suite used to carry general packet radio service (GPRS) within 5G networks. GTP allows users to move while staying connected to the Internet. This protocol can be used with Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). 5G encapsulates current IP packets into frames by following the GTP protocol, which supposes an increase in packet sizes and extra complexity in the search of given string patterns for detection. In these experiments, we used Snort (version 2.9.9.0) as opposed to other tools like Suricata [29], for example, because Snort is actually the only well-known DPI tool supporting GTP packets.

In this experiment, focusing on the time difference between GTP and IP packets being analyzed by Snort, Fig. 6 shows those times when varying the number of detection rules in Snort, because this increase in rules may also imply that Snort needs more time to analyze GTP packets. To this end, the figure depicts the times from the injection of a single GTP or IP packet (in the network interface where



**Fig. 6** Time required by Snort to detect and log alerts when using GTP and IP packets in accordance to the number of rules configured in Snort

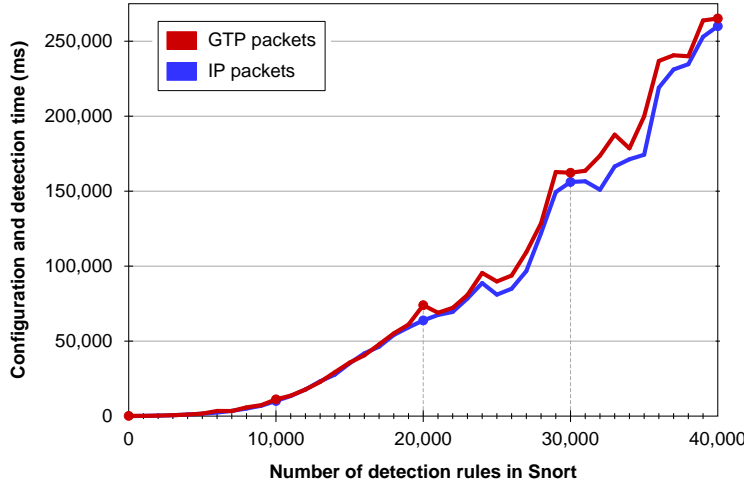
Snort is sniffing) to the logging of the corresponding alert (recorded by Snort internally). These tests were conducted by increasing the number of detection rules being managed by Snort.

As shown in Fig. 6, Snort's analysis procedures do not represent a significant increase in time when managing GTP packets in comparison with IP. This is independent of the number of detection rules configured in Snort. On average, the increase in time between two types of traffic (GTP and IP) is around 0.4896 %, which represents very similar and totally affordable times. This fact confirms that the biggest problem of Snort will continue to be the amount of network traffic it should analyze in real time, regardless of the network technologies being used.

Regarding the amount of traffic that Snort can support, it is well known that it works well on networks up to 1 Gbps, suffering problems from 1.5 Gbps when Snort starts discarding packets due to overload. In [30], a complete study in performance is provided by comparing Snort and Suricata, proving that Snort has a packet loss rate of 3.1 % in networks with a transmission rate of 2 Gbps. The experiments performed by the authors include results ranging from 1 Gbps to 10 Gbps network speed, concluding that Snort dropped more packets at 10 Gbps network speed compared to Suricata. However, Suricata does not allow native GTP traffic management, so additional components would have to be installed on the network for GTP decapsulation. Focusing on Snort, there are additional solutions that improve performance with the use of parallelization techniques such as, among others, those based on ASIC (Application-Specific Integrated Circuit), which achieve speeds close to 7.2 Gbps with negligible packet loss [31]. While these solutions offer substantial improvements, they are also hardware-based approaches with higher implementation costs.

This amount of traffic will clearly affect the number of suspicious zombies that Snort can manage in real time to confirm that they are in place. In case of Snort overloading, as discussed in Section 4, a new Snort VNF should be dynamically deployed to balance the workload in detection. This fact will entail the migration of part of the current Snort VNF to the new one in order to prevent Snort from discarding packets during the analysis phases.





**Fig. 7** Time required by Snort for its configuration when varying the number of rules for analyzing GTP and IP packets

### 6.2 Test 2: Start-up time of Snort for its setting in GTP and IP detection

Following the previous experiment, which concluded the possibility of migrating detection functionalities from an overloaded Snort to another newly deployed Snort VNF, this experiment aimed to analyze how much time it takes to configure a new Snort to continue with the detection processes.

The tests were carried out by increasing the number of detection rules to configure in Snort, and its start-up until detecting the first GTP or IP packet directly injected after Snort has been configured. In this regard, we also divided the tests between traffic types to verify the implication of using GTP packets, if any, as opposed to IP. Fig. 7 shows the times extracted for this experiment, varying the rules up to 40,000.

As shown in Fig. 7, we can quickly check that there is an inflection point when 10,000 rules are configured, with the configuration and detection time drastically increasing from that moment on. At that point Snort needs around 7 seconds for performing that process, which is quite a reasonable time for the configuration of a new Snort VNF. In addition, we can also notice very small variations in time when dealing with GTP and IP network packets.

### 6.3 Test 3: How zombies' mobility and CRI affect the deployment and configuration times in Snort

Once the previous experiments have shown the time required to deploy and configure new or existing Snort instances, this experiment is tried to show the number of deployments and configurations required in a realistic scenario where zombies move around different RANs. In this sense, and to answer the last important aspect pointed out at the beginning of this section, we performed an experiment with several configurations that considers the following representative aspects:

- the number of suspicious zombies,
- the number of detection rules accepted by Snort,
- the mobility of suspicious zombies, and
- the number of Radio Access Networks (RANs) where the suspicious zombies can be located.

Table 1 shows how the deployments of new Snort VNFs and the configurations of the existing ones vary when suspicious zombies increase from 1 to 1,000,000. To perform this experiment as realistically as possible, we established 200 detection rules accepted by Snort, 20 different RANs, and 5 random movements per zombie.

<b>Zombies</b>	<b>1</b>	<b>10</b>	<b>100</b>	<b>1,000</b>	<b>10,000</b>	<b>100,000</b>	<b>1,000,000</b>
Deployments	4	18	20	20	60	502	5054
Configurations	1	26	385	3,808	37,981	379,963	3,788,933

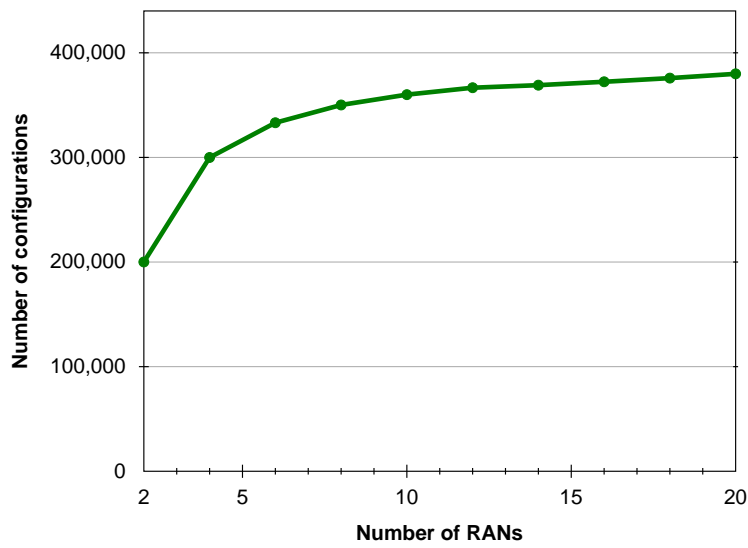
**Table 1** How the number of suspicious zombies affects the number of configurations and deployments of Snort

Table 1 shows a lineal relationship between the number of suspicious zombies and the number of deployments and configurations. When the number of zombies increases, the number of deployments and configurations also increases. It is important to point out that the majority of deployments are performed in the first movement due to the fact that RANS do not have any Snort running in a proactive way. In that sense, the number of deployments (and therefore the time) could be reduced by deploying at least one Snort before moving the first suspicious zombie. In addition, it is also important to know that the number of deployments is much lower than the configurations because when a single zombie moves from one RAN to another, a new configuration is required. However, to deploy a new Snort in a given RAN, at least 200 new suspicious zombies should appear in a specific moment.

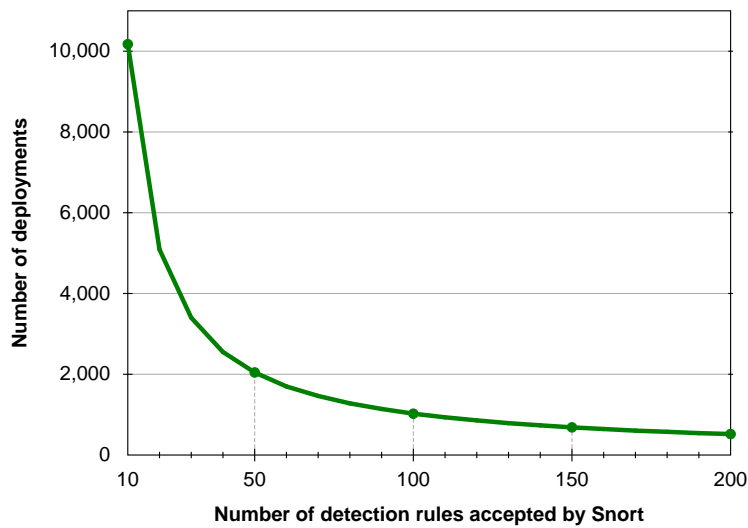
Fig. 8 depicts how the zombies' mobility affects the number of Snort's configurations when the number of RANs increases from 2 to 20. As shown in the figure, when the number of RANs increases, the number of required configurations also increases. This is because having more RANs increases the probability of zombies' movements. Specifically, the interval from 2 to 5 RANs is where the number of required configurations is lower. This is due to the fact that the probability of moving to a new RAN is lower than when there are more RANs. Having more than 5 does not have a high impact on the number of required configurations as the majority of zombies are already changing RAN in each movement.

Finally, Fig. 9 shows that when the number of detection rules accepted by Snort increases, the number of Snort's deployments decreases. From 10 to 100 detection rules, the number of required deployments is higher than having more than 100 rules due to the lower capacity of Snort to detect zombies at the same time, and the number of suspicious zombies considered in our experiment. In addition, it is also important to consider that the more detection rules in Snort, the longer is required for Snort's detection time (as demonstrated in the previous experiments).

In conclusion, this experiment has shown that there is a direct relationship between the number of zombies, the number of rules accepted by Snort, the number



**Fig. 8** Effect of the number of RANs on the number of Snort's configurations (100,000 zombies, 20 RANs, and 5 movements per zombie)



**Fig. 9** Effect of the number of rules accepted by Snort on the deployments of new Snorts (100,000 zombies, 20 RANs, and 5 movements per zombie)

of RANs, and the number of deployments and reconfigurations of Snort, which defines the time required to analyze and detect zombies.

## 7 Conclusions and future work

In this article, we have extended the solution presented in [5] with the definition of a self-protection use case that shows how the proposed architecture autonomically manages Snort VNFs required to monitor and proactively detect potential attacks conducted by botnets in a 5G scenario. In addition, we performed several experiments to ascertain the best configuration of the management policies in charge of deciding what, where, and when to deploy or reconfigure monitoring and detection services. Among the different aspects considered by our experiments to ensure the Snort VNF provision, we highlight the number of bots comprising the botnet, their mobility, the traffic generated by them, and the available network resources and their capabilities.

As future work, we plan to deploy our 5G-oriented architecture in a fully virtualized environment, using OpenStack as VIM to instantiate the virtual resources in which VNFs will run; OpenDaylight as SDN Controller to control the virtual switches deployed in the OpenStack infrastructure; and Open Baton to orchestrate the elements belonging to the SDN and NFV planes. Considering this virtualized environment, we plan to automatically deploy VNFs in real time in different RANs and the EPC (Evolved Packet Core) belonging to 5G mobile networks, in order to ensure the provision of the monitoring services. Finally, we also plan to work in proactive approaches to migrate VNFs before they get overloaded, thus improving the continuity of monitoring services over time.

## Acknowledgements

This work has been supported by a Séneca Foundation grant within the Human Resources Researching Postdoctoral Program 2018, a postdoctoral INCIBE grant within the “Ayudas para la Excelencia de los Equipos de Investigación Avanzada en Ciberseguridad” Program, with code INCIBEI-2015-27352, as well as European Commission FEDER funds, under grant TIN2015-66972-C5-3-R and the European Commission Horizon 2020 Programme under grant agreement number H2020-ICT-2014-2/671672 - SELFNET (*Framework for Self-Organized Network Management in Virtualized and Software Defined Networks*).

## References

1. The 5G Infrastructure Public Private Partnership (5G-PPP), Key Performance Indicators, <http://5g-ppp.eu/kpis>.
2. S. Singh, R. K. Jha, A survey on software defined networking: Architecture for next generation network, *Journal of Network and Systems Management* 25 (2) (2017) 321–374. doi:10.1007/s10922-016-9393-9.
3. R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, R. Boutaba, Network function virtualization: State-of-the-art and research challenges, *IEEE Communications Surveys Tutorials* 18 (1) (2015) 236–262. doi:10.1109/COMST.2015.2477041.
4. L. Jorgueski, A. Pais, F. Gunnarsson, A. Centonza, C. Willcock, Self-organizing networks in 3GPP: Standardization and future trends, *IEEE Communications Magazine* 52 (12) (2014) 28–34. doi:10.1109/MCOM.2014.6979983.
5. A. Huertas Celdrán, M. Gil Pérez, F. J. García Clemente, G. Martínez Pérez, Automatic monitoring management for 5G mobile networks, in: *12th International Conference on Future Networks and Communications*, 2017, pp. 328–335. doi:10.1016/j.procs.2017.06.102.

6. Z. Huang, S. Liu, X. Mao, K. Chen, J. Li, Insight of the protection for data security under selective opening attacks, *Information Sciences* 412–413 (2017) 223–241. doi:<https://doi.org/10.1016/j.ins.2017.05.031>.
7. B. Gupta, D. P. Agrawal, S. Yamaguchi, Handbook of research on modern cryptographic solutions for computer and cyber security, IGI Global, 2016.
8. J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, *Computers & Security* 72 (2018) 1–12. doi:<https://doi.org/10.1016/j.cose.2017.08.007>.
9. Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, A. Rindos, SDIoT: A software defined based internet of things framework, *Journal of Ambient Intelligence and Humanized Computing* 6 (4) (2015) 453–461. doi:[10.1007/s12652-015-0290-y](https://doi.org/10.1007/s12652-015-0290-y).
10. J. A. Saucedo-Martínez, M. Pérez-Lara, J. A. Marmolejo-Saucedo, T. E. Salais-Fierro, P. Vasant, Industry 4.0 framework for management and operations: A review, *Journal of Ambient Intelligence and Humanized Computing*, In Press doi:[10.1007/s12652-017-0533-1](https://doi.org/10.1007/s12652-017-0533-1).
11. F. X. A. Wibowo, M. A. Gregory, K. Ahmed, K. M. Gomez, Multi-domain software defined networking: Research status and challenges, *Journal of Network and Computer Applications* 87 (2017) 32–45. doi:[10.1016/j.jnca.2017.03.004](https://doi.org/10.1016/j.jnca.2017.03.004).
12. N. L. M. van Adrichem, C. Doerr, F. A. Kuipers, OpenNetMon: Network monitoring in OpenFlow software-defined networks, in: 2014 IEEE Network Operations and Management Symposium, 2014, pp. 1–8. doi:[10.1109/NOMS.2014.6838228](https://doi.org/10.1109/NOMS.2014.6838228).
13. S. R. Chowdhury, M. F. Bari, R. Ahmed, R. Boutaba, PayLess: A low cost network monitoring framework for software defined networks, in: 2014 IEEE Network Operations and Management Symposium, 2014, pp. 1–9. doi:[10.1109/NOMS.2014.6838227](https://doi.org/10.1109/NOMS.2014.6838227).
14. P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, L. Z. Granville, Interactive monitoring, visualization, and configuration of OpenFlow-based SDN, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management, 2015, pp. 207–215. doi:[10.1109/INM.2015.7140294](https://doi.org/10.1109/INM.2015.7140294).
15. S. Namal, I. Ahmad, A. Gurtov, M. Ylianttila, SDN based inter-technology load balancing leveraged by flow admission control, in: 2013 IEEE SDN for Future Networks and Services, 2013, pp. 1–5. doi:[10.1109/SDN4FNS.2013.6702551](https://doi.org/10.1109/SDN4FNS.2013.6702551).
16. Q. Duan, N. Ansari, M. Toy, Software-defined network virtualization: An architectural framework for integrating SDN and NFV for service provisioning in future networks, *IEEE Network* 30 (5) (2016) 10–16. doi:[10.1109/MNET.2016.7579021](https://doi.org/10.1109/MNET.2016.7579021).
17. R. Muñoz, R. Vilalta, R. Casellas, R. Martínez, T. Szyrkowiec, A. Autenrieth, V. López, D. López, Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks, *Journal of Optical Communications and Networking* 7 (11) (2015) B62–B70. doi:[10.1364/JOCN.7.000B62](https://doi.org/10.1364/JOCN.7.000B62).
18. G. Mantas, N. Komninos, J. Rodriguez, E. Logota, H. Marques, Security for 5G communications, in: Fundamentals of 5G Mobile Networks, John Wiley & Sons, Ltd., 2015, pp. 207–220. doi:[10.1002/9781118867464.ch9](https://doi.org/10.1002/9781118867464.ch9).
19. M. Anagnostopoulos, G. Kambourakis, S. Gritzalis, New facets of mobile botnet: Architecture and evaluation, *International Journal of Information Security* 15 (5) (2016) 455–473. doi:[10.1007/s10207-015-0310-0](https://doi.org/10.1007/s10207-015-0310-0).
20. D. Bhattacharjee, Stepping stone detection for tracing attack sources in Software-Defined Networks, Master's thesis, Aalto University, Finland (Jun. 2016).
21. J. Chen, X. Cheng, R. Du, L. Hu, C. Wang, BotGuard: Lightweight real-time botnet detection in Software Defined Networks, *Wuhan University Journal of Natural Sciences* 22 (2) (2017) 103–113. doi:[10.1007/s11859-017-1223-8](https://doi.org/10.1007/s11859-017-1223-8).
22. C. C. Machado, L. Z. Granville, A. Schaeffer-Filho, ANSwer: Combining NFV and SDN features for network resilience strategies, in: 2016 IEEE Symposium on Computers and Communication, 2016, pp. 391–396. doi:[10.1109/ISCC.2016.7543771](https://doi.org/10.1109/ISCC.2016.7543771).
23. M. Gil Pérez, A. Huertas Celdrán, F. Ippoliti, P. G. Giardina, G. Bernini, R. Marco Alaez, E. Chirivella-Perez, F. J. García Clemente, G. Martínez Pérez, E. Kraja, G. Carrozzo, J. Alcaraz Calero, Q. Wang, Dynamic reconfiguration in 5G mobile networks to proactively detect and mitigate botnets, *IEEE Internet Computing* 21 (5) (2017) 28–36. doi:[10.1109/MIC.2017.3481345](https://doi.org/10.1109/MIC.2017.3481345).
24. The 5G-PPP SELFNET project, Deliverable D2.3: Prototype and report framework for enabling the encapsulation of NFV and SDN applications (Apr. 2016). doi:[10.18153/SLF-671672-D2.3](https://doi.org/10.18153/SLF-671672-D2.3).
25. ETSI NFV ISG, Network Functions Virtualisation (NFV); network operator perspectives on NFV priorities for 5G, [http://portal.etsi.org/NFV/NFV\\_White\\_Paper\\_5G.pdf](http://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf) (Feb. 2017).

26. M. Mahmoud, M. Nir, A. Matrawy, A survey on botnet architectures, detection and defences, *International Journal of Network Security* 17 (3) (2015) 272–289. doi:10.6633/IJNS.201505.17(3).06.
27. J. Demarest, Taking down botnets: Public and private efforts to disrupt and dismantle cybercriminal networks (Statement Before the Senate Judiciary Committee, Subcommittee on Crime and Terrorism), <http://www.fbi.gov/news/testimony/taking-down-botnets> (Jul. 2014).
28. Sourcefire, Inc., Snort: An open source network intrusion detection and prevention system, <http://www.snort.org>.
29. Open Information Security Foundation, Suricata: Open source IDS/IPS/NSM engine, <http://suricata-ids.org>.
30. S. A. Raza Shah, B. Issac, Performance comparison of intrusion detection systems and application of machine learning to Snort system, *Future Generation Computer Systems*, In Press doi:10.1016/j.future.2017.10.016.
31. Y.-M. Hsiao, M.-J. Chen, Y.-S. Chu, C.-H. Huang, High-throughput intrusion detection system with parallel pattern matching, *IEICE Electronics Express* 9 (18) (2012) 1467–1472. doi:10.1587/elex.9.1467.