Computer Communications 145 (2019) 284–308

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Detection of economic denial of sustainability (EDoS) threats in self-organizing networks

Marco Antonio Sotelo Monge [a,*], Jorge Maestre Vidal [b,*], Gregorio Martínez Pérez [c]

[a] Department of Software Engineering and Artificial Intelligence, School of Computer Science, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases 9, Ciudad Universitaria, 28040 Madrid, Spain

[b] Indra Digital Lab, Avenida de Bruselas 35, 28108 Alcobendas, Madrid, Spain

[c] Department of Communications and Information Engineering, Faculty of Computer Science, Universidad de Murcia, Murcia, Spain

ARTICLE INFO

Keywords:
Cloud computing
Economic denial of sustainability
Information security
Intrusion detection
Network function virtualization
Self-organizing networks

ABSTRACT

This paper reviews the threat of economic denial of sustainability on recent communication networks and discusses their adaptation to emergent scenarios suited for self-organization and network function virtualization. Thorough the performed research two novel threats were defined: workload-based EDoS (W-EDoS) and Instantiation-based EDoS (I-EDoS). W-EDoS is characterized by executing expensive requests in terms of computational resources at the victim system, hence exhausting its workload and forcing operators to contract additional resources. On the other hand, I-EDoS occurs when the cloud management software deploys more instances of virtual network functions than needed as a response to requests that resemble legitimate, but are malicious, thus increasing the cost of the hired resources. In order to contribute to their mitigation, a security architecture that incorporates network-based intrusion detection capabilities for their recognition is proposed. It implements strategies that lie on predicting the behavior of the protected system, constructing adaptive thresholds, and clustering of instances based on productivity. An extensive experimentation has been conducted to demonstrate the proposal effectiveness, which includes case studies and the accuracy assessment when considering different adjustment parameters. Under the most intense conditions, the highest AUC performed above 98% when assessing the I-EDoS detection accuracy, being the same reading higher than 99% in the case of W-EDoS.

1. Introduction

The complexity of network architectures is growing day by day and they demand faster, more efficient and more scalable management paradigms. In the last years, 5G has emerged as the promising technology to deal with the challenging requirements of future generation networks [1]. It empowers a smart integration of the most innovative advances on Network Function Virtualization (NFV), cloud computing, Software-Defined Networking (SDN), Artificial Intelligence (AI) and Self-Organizing Networks (SON). In particular, the synergies between SDN and SON networks in 5G are addressed as an important research topic [2] to enable the fulfillment of the disruptive key performance indicators (KPIs) promoted by 5G [3]. Recently, several 5G research projects have integrated cloud computing, SON and cognitive network concepts to design modern and automated management architectures [4], which are able to monitor heterogeneous network infrastructures. It is achieved by the incorporation of analysis, decision-making and learning capabilities about the status of complex network scenarios, so that, the automatic enforcement of actions to mitigate risks and optimize operations in 5G networks is possible. An outstanding example is the SELFNET project (Framework for Self-Organized Network Management in Virtualized and Software Defined Networks) [5] which targets on developing a 5G Self-Organizing Network management framework by combining the advantages of supportive 5G technologies; thus conforming a scalable, extensible and smart architecture aimed to create a network management loop that decreases capital and operational expenditures derived from the efficiency of autonomic management tasks.

The research described throughout this paper focuses on the Self-Organizing Networks (SON) as promising solutions to address the aforementioned challenges. Self-Organizing Networks have originally emerged in the telco industry as an effort of the Third Generation Partnership Project (3GPP) to meet the challenges of modern mobile LTE networks by shifting from manual to automated management aimed to ensure operational efficiency [6]. The original 3GPP standardization of a SON framework [7] remarks the reduction of operational costs due to automatization procedures, which entails the transition from an operation controlled (open loop) to an autonomous (closed

* Corresponding authors.
E-mail addresses: masotelo@ucm.es (M.A. Sotelo Monge), jmaestre@indra.es (J. Maestre Vidal), gregorio@umu.es (G. Martínez Pérez).

https://doi.org/10.1016/j.comcom.2019.07.002
Received 27 February 2018; Received in revised form 17 February 2019; Accepted 7 July 2019
Available online 10 July 2019
0140-3664/© 2019 Elsevier B.V. All rights reserved.

loop) architecture. SON architectures gained importance along the last years since the characteristics of modern network infrastructures; such as their complexity, quality of service demands, traffic growth, virtualization capabilities, traffic isolation, among others; became more crucial for operators. Aside from that, it is worth recalling the original functionalities of autonomic management envisaged by IBM [8] that have laid down the principles of self-organizing networks, defined as: monitoring, analyzing, planning and executing. Hence, it is to remark that the major emphasis of this research is posed in the analysis stage since its final goal is focused on EDoS threats detection for accomplishing self-protecting capabilities. Monitoring functions are addressed but in a lesser extent for information gathering, whereas planning and executing of response actions are succinctly implemented as incident notification. Hence, the introduced EDoS detection methods rely on self-organizing principles, and thus paves the way to a full implementation of a closed-loop architecture. Notwithstanding, dealing with the complexity of planning and executing are postponed for future investigation since they truly conform the mitigation part.

Our other object of study is the role of virtualization in cloud computing and networking, which enhances dynamic scalability. This is a mandatory solution not only for dealing with the on-demand provisioning model driven by cloud operators [9] but also for achieving a significant reduction of service deployment times in 5G networks [10]. Nevertheless, auto-scaling policies expose vulnerabilities that can be exploited to conduct attacks targeted to cause a workload-based effect on the existing virtual instances, hence making a cloud service unsustainable due to the generated economical overspending [11]. This effect is referred as Economical Denial of Sustainability (EDoS). This paper reviews this threat from the perspective of Self-Organizing Networks and consequently remarks the strong dependence on virtualization technologies in 5G, hence currently grounded by cloud platforms and the ETSI-NFV [12] management and orchestration framework. In this way the presented research extends beyond conventional EDoS attacks/countermeasures and introduces two previously unknown EDoS variations: Workload-based (W-EDoS) and (Instantiation-based) I-EDoS threats, and it is demonstrated that by exploiting vulnerabilities in the current NFV platforms, their successfull execution is possible. To this day, the authors have no record of previous attempts to cover these new intrusion vectors. A general idea of the EDoS detection approach and its main outcomes have been previously introduced in [13], which is extensively detailed in this paper. Therefore, to cooperate with the research community towards their detection and mitigation, the following main contributions are presented:

- An in-depth review of the EDoS threats and the efforts made by the research community for their detection, mitigation and identification of sources.
- A comprehensive formalization of the distinction between EDoS threats and similar network incidents, among them normal activities, flash crowds and denial of service.
- The definition of a pair of emerging new generation threats: Workload-based EDoS and Instantiation-based EDoS.
- A multi-layered architecture for EDoS attack detection, which describes the management of the acquired information from its monitoring to the notification of possible threats.
- A novel entropy-based EDoS detection approach, which assuming its original definition, allows to discover unexpected behaviors on local-level metrics related with the auto-scaling capabilities of the victim system.
- An evaluation methodology adapted to the singularities of the EDoS threats and the assumptions driven by their original definition.
- Comprehensive experimental studies that validate the proposed detection strategy, in this way motivating its adaptation to future use cases

The paper is organized into eight sections, and the first of them is the present introduction. Section 2 reviews the background of the performed research. Section 4 characterizes the network situations taken into account. Section 5 introduces a novel approach for their detection. Section 6 details the performed experimentation. Section 7 discusses the obtained results; and Section 8 presents the conclusions and future research lines.

2. Background

The following describes the main features of the Self-Organizing Networks, the EDoS threat and the efforts of the research community toward providing effective detection and mitigation approaches.

2.1. Autonomic network management

Network management tasks in early Network Management Systems (NMS) were highly dependent on the intervention of network administrators. NMSs were limited to Command-line Interfaces (CLI) for performing the configuration of each network node, a task that gained much more complexity in an ever-increasing network. Some technological approaches were introduced to partially automate this process (i.e. policy-based management, active networks, mobile agents), but the manual intervention of the network administrator for developing the necessary network policies, code or templates for customizing the network behavior was still mandatory [14]. Even though the NMSs were partially automated, they were static by nature, hence lacking the ability to take decisions for adapting the network with the surrounding context. On the other hand, the concept of autonomic computing emerged in the research land. It was introduced by IBM in 2001 [8] inspired in the biological systems, such as the autonomic nervous system, in the sense that they are capable of self-govern vital functionalities of the human body without conscious intervention. The term Autonomic Computing (AC) refers to computing systems that can perform management operations by themselves given a set of high-level objectives provided from their administrators. Autonomic Computing targets on four aspects of self-management: self-configuration, self-healing, self-optimization and self-protection (they are collectively referred as self-CHOP properties [14]). Autonomic elements are the basis of autonomic systems and they are capable to manage their internal behavior and their relationships with other entities within an autonomic system. More specifically, an autonomic element is conceptualized as set of managed elements (or functional units) that perform operational operations, and an autonomic manager (management unit) that controls the managed elements [15]. According to the IBM's vision, the autonomic manager distinguishes four main functionalities: *monitoring* of the managed elements, *analyzing* their performance, *planning* and *executing* a set of actions in response to the characteristics of the system context. The self-organizing approach prompted by autonomic computing soon gained popularity in diverse research areas of computer science pushed by the motivation for transferring the robustness and flexibility of biological entities into software systems. Such endeavor led to conduct novel approaches benefited from self-organizing behaviors in the development of middleware architectures, information systems and management, security, robotics and network management [16]. The latter being accurately applied in mobile ad-hoc and sensor networks. Hence, and in a broader sense, the term autonomic communications was used to refer for developing self-managing and self-regulating network and communications infrastructures, with the final vision of a networked world in which the network architectures and their associated services can operate totally unsupervised while performing the self-CHOP properties, or other self-* characteristics at some extent [14,15]. More recently, the 5G-PPP Working Group on Network Management and QoS studied the implications of advanced autonomic models to lay the management principles of 5G networks towards a cognitive model [17]. The most remarkable trait is the

exploitation of "the selves" for accomplishing the goals of a more demanding network architecture, while tackling with scalability and flexibility. Such cognition is enabled by the application of machine learning to develop self-aware, self-configuring, self-optimization, self-healing and self-protecting systems. This landscape raises emergent challenges for autonomic network management, thus leveraging the contributions of recent advancements and trends of novel technologies such as softwarization or network slicing. In addition, 5G-PPP outlines three autonomic principles that drive the evolution of autonomic management in 5G: autonomic software-defined networks, autonomic diagnosis/anticipation, and autonomic adaptation. Having those principles in mind, the development of network solutions should target their attainment.

2.2. Self-organizing networks

The term Self-organized Networks (SON) was introduced by the Next Generation Mobile Networks (NGMN) at 2007. It has been widely adopted to describe the application of the self-organizing principles into the specific domain of mobile network architectures [18]. Its main goal was the simplification of operations and maintenance of mobile architectures, hence leading to the reduction of capital and operational expenditures whilst protecting revenue as well. The SON concept has been adopted by the 3GPP in the 32.500 Technical Specification series [7] as a technological driver automating network planning, configuration and optimization processes for helping the network operator to reduce OPEX by reducing manual intervention. It is to note that, in contrast with the previously introduced autonomic computing, the 3GPP SON definition does not advocates for a solely automated management but holds awareness on the role of the human operator. This nuance has also been stated as leaving the human mostly out of the loop in a self-organizing system [19]. From a functional perspective, SON networks includes three outstanding capabilities: self-healing, self-optimization and self-configuration [18]. Self-configuration involves the processes to automatically configure new network nodes deployed in the network by downloading and parameterizing the required software [20]. Self-optimization addresses optimal performance conditions by comparing the current network status with the target parametrization, which might trigger corrective actions when required [6]. In addition, self-healing entails the detection of network failures, the diagnosis of the situation, and the enforcement of recovery actions to restore the affected network function [21]. Even though self-protection is not explicitly documented in the technical specification, it is a key capability intrinsic to self-organizing systems which has addressed different use cases as evidenced by the research literature [15,22].

The original SON architecture [18] comprises an Operation, Administration and Maintenance (OAM) subsystem deployed either as a centralized, distributed or hybrid architectural approach. The simplest closed-loop network OAM architecture [23] entails the presence of sensors deployed along the network to monitor crucial network elements for Performance Management (PM). The monitored information is analyzed according with the network policies defined to automate the Configuration Management (CM) processes. Such policies are planned and enforced through different network actuators adapted to specific domain managements [14]. Fig. 1 illustrates a simplified view of a closed-loop architecture.

On the other hand, cognitive networks are autonomous networks that include learning capabilities in the management lifecycle; ranging from monitoring, decision-making and action enforcement processes in order to acquire knowledge about the context [24]. Thereby, SON-cognitive networks have represented an ongoing trend for the management of modern networks [25], in this way adapting the processes related with sensing the network, planning, decision-making and actuation according to the operational context. Even though the original SON definition remains suitable, it has adopted new characteristics adapted to the evolving mobile network technologies such as LTE, 4G and 5G. In this context, SON networks that include cognitive capabilities cope more efficiently with the management challenges to meet the cognition cycle [21], thus having the ability to perform complex network analysis and to deploy smart proactive and reactive actions to mitigate, correct or optimize network services. This cognitive approximation is in line with the principles introduced by the 5GPPP [17], previously introduced in this section. SON networks are key enablers of 5G networks, taking advantage of the application of advanced machine learning techniques to adapt the network for complex self-healing, self-optimization, self-configuration, self-protection and other use case scenarios. 5G provides suitable capabilities to evolve powerful SON architectures relying on the decoupling of network and data planes promoted by SDN, the advanced on-demand provision fostered by cloud platforms, the immersion of NFV management and orchestration architectures, and the evolution towards next generation self-management architectures. Finally, it is worth stressing the object of study of this research has been framed into a SON-centralized architecture, as defined by the 3GPP [18] (v. 15.0.0, 2018) on which SON algorithms are executed at the Network Management level.

2.3. Economic denial of sustainability in cloud computing

At November 2008, Hoff [26,27] firstly hypothesized about the presence of a novel strain of the denial of service threat, which was termed Economic Denial of Sustainability, abbreviated EDoS. It described a specific family of attacks against the different cloud computing platforms, where the intruder aimed on increasing the economic costs derived from both maintenance and provision of the services offered, hence making their support less viable, even achieving denial. Interested in this publication, R. Cohen [28] extended the EDoS definition by highlighting the important role played by exploiting the self-scaling mechanisms considered by each provider. That is, if the attacker succeeds in forcing the users to hire additional computing resources by exploiting the self-scaling policies of the supplier, clients will have to pay more money, which may lead them to change to a more competitive supplier. This implies that the service offered ceases to be profitable, and that hence the EDoS attack achieved its main purpose.

Although it is a novel concept, inherent to the emergent technologies, EDoS rapidly drawn the attention of the research community and organizations for information security, according to them becoming a DDoS variant framed in the categories Reduction of Quality (RoQ) [29] and Fraudulent Resource Consumption (FRC) threats [30]. They stated that EDoS typically attempt to exploit the "pay-as-you-go" service model offered by most of the cloud computing providers [31,32], which leads to its adaptation to different metrics, and self-scaling policies or mechanisms [29]. The following describes its main characteristics, impact and the defensive strategies raised by the research community.

2.3.1. Characteristics and impact

In general terms, EDoS attacks pose similarities with conventional DDoS threats, especially those based on flooding [33]. However, EDoS raise a significantly different problem that requires a separate solution: assuming the original definition of Hoff [26,27], EDoS focuses on forcing the increase of the economic cost of a cloud computing service instead of directly preventing its provision, as occurs in their predecessors. Therefore, for the sake of effectiveness the number of connections and requests involved in their execution may resemble the legitimate activities. Hence network metrics traditionally studied when detecting flooding-based threats (e.g. number of requests, number of sessions, total amount of payload, bandwidth consumption, etc.) display distributions similar to those gathered at normal traffic. The attacker usually exploits vulnerabilities at Application layer with the purpose of extend the computational cost of resolving the received requests [30]. In this last characteristic precisely lies the greatest difference between EDoS attacks and massive accumulations of legitimate

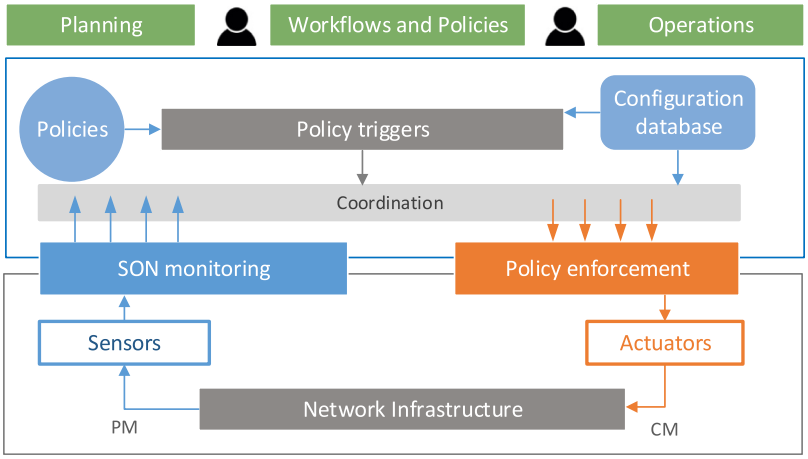


Fig. 1. Closed-loop Self-Organizing Network.

requests, the second commonly referred as flash crowds [34], in which the processing cost is similar to those of the requests monitored before the agglomeration.

The computational cost of attending the received requests can be exploited to induce EDoS in several ways, for example by requesting large files or costly queries [35], HTTP-requests on XML files [36], or taking advantage of alternative Application layer vulnerabilities related with web services [37–39]. G. Sonami et al. [30] studied in-depth the impact of these threats, concluding that it varies depending on the affected party. In the case of the customers, EDoS directly involves economic losses, which can lead them to the election of more economical suppliers. This indirectly causes the providers to lose customers, and therefore their profit is reduced. As discussed in [39], the increase in the computation requirements for addressing the malicious requests also implies a decrease in the quality of service offered, which is a consequence of the need for among others, more infrastructure, deployment of additional virtualized network functions (VNFs), multi-tenancy capabilities, etc. It was defined by Bremler-Barr et al. [29] as collateral damage, publication in which the authors demonstrated that by launching an easy low-rate EDoS attack, it was possible to prompt additional causalities directly related with enforce auto-scaling and the waste of resources it involves.

3. Related work

Despite relevance, the bibliography does not include a large collection of publications focused on the defense against EDoS threats. The studies that address this problem usually assume metrics at network-level, usually confusing features for EDoS identification with those that typically detect flooding-based DDoS attacks. The nature of EDoS threats poses instead resemblance with normal network traffic behaviors, hence requiring different defensive approaches to find particular discordances both at network and application level.

In [11,35,40] some of the most relevant proposals are collected and discussed. With the purpose of facilitate their understanding, they are classified according to their scope, as traditionally organized in the research related with the defense against DDoS [41]: detection, prevention/mitigation, and identification of sources.

3.1. Detection

Detection approaches share the main purpose of identifying the incoming threats. Therefore, they are often the triggering situation prior to the deployment of mitigation capabilities, as well as discovering the sources of intrusion attempts. Assuming as classification criteria the scope of the metrics to be analyzed, in the bibliography there are two types of proposals: those extracted at local or network monitoring environments. On the one hand, publications focused on local metrics that traditionally modeled resource consumption and studied self-scaling processes [42]. For example, in [43] a modeling approach for classifying website users potentially involved in EDoS attacks is proposed. Accordingly, legitimate behaviors were identified based on analyzing website activities tracked from log files. Then fuzzy entropy is applied for feature selection in order to build classification models with good discrimination ability and enhanced detection accuracy. Similarly, in [44] the application of anomaly-based detection methods and auto-regressive forecasting for detecting fraudulent auto-scaling requests associated to EDoS behavioral patterns is studied. The modeling stage is conducted by analyzing historical datasets of elastic microservices when handling legitimate workload, from which normal activities generated by the running applications led to discriminate malicious scaling operations. Furthermore, network-based detection approaches analyze information provided by packet headers [41,45] browsing habits of normal clients [42,46]. Although few investigations have focused on local traits, they demonstrated greater efficacy, since they directly assumed the definition of EDoS attacks originally posted by Hoff [26,27]. However, methods based on network traits analysis took advantage of the state of the art about flooding-based DDoS detection, which in many cases has led to confusion between attacks types.

3.2. Mitigation and prevention

Once the threat has been successfully identified, the mitigation capabilities act. They mainly focus on increasing the protected system restriction level through the deployment of more complex access control techniques, usually Turing tests based on image recognition [47,48] and resolution of cryptographic puzzles [47,49,50]. It is important to highlight that most of the publications gathered in the state of the art addressed the mitigation problem by the aforementioned classical solution for cloud computing security incidents. For example, in [51] a mitigation approach that combines network-level metrics and Turing tests for uniquely identifying client nodes involved in EDoS attacks was introduced. At detection phase normal traffic, flash crowds and attacks are analyzed by modeling the CPU utilization of the virtualized nodes. This leads to prevent access to unique sources of malicious request once the threat is detected, even when they are placed behind a NAT server. In [52], the mitigation of EDoS attacks by developing a framework with online anomaly-based detection capabilities suited for a multi-tier IoT architecture is detailed. It took advantage of two separate detection models (regarding network and computer resources) built at micro-service level, from which anomaly likelihood scores are assessed to prevent or allowing autoscaling processes. Similarly, in [53] a cost-effective model grounded on

fog computing was proposed. This aimed on tackling EDoS threats alongside with flooding DDoS mitigation, where an intermediate fog defender analyzed all traffic traversing the cloud server by considering packet size, protocol, request ratio and attack duration as threat indicator criteria. On the other hand, publications towards preventing EDoS threats aimed on modeling and optimizing costs related with processing malicious requests [54]. Unlike mitigation approaches, they do not require the prior detection of the threat. For instance, Xue et al. [55] designed an access control mechanism for preventing EDoS attacks performed on cloud storage systems. Ciphertext-policy Attribute-based Encryption (CP-ABE) schemes were considered to raise both fine-grained access and confidentiality when downloading encrypted files. Consequently, users were requested to authenticate before their download, hence preventing resource-exhaustion activities from malicious/untrusted parties. Notwithstanding, most of the proposals categorized in mitigation could be deployed as prevention measures.

3.3. Identification of sources

Finally, the research that aims on identifying the origin of the attacks attempt to discover the attacker itself. Given the difficulty that this entails, and the fact that it is often not possible due to the restrictions of Internet providers, as well as intermediate elements of the backbone, privacy and data protection policies, etc. from the practical point of view, identifying the origin is often simplified at reaching as close as possible to the attacker. Most of the classic techniques in the state of the art of the defense against conventional DDoS serve for this purpose [56], among them those based on the analysis of error messages [57], deployment of honeypots [58] or packet marking [59], in all of them playing the network topology an essential role [60]. More recently, it is demonstrated how two markov-based anomaly detection models are built in parallel for identifying sources participating on EDoS attacks [61]. There, legitimate user profiling relies on patterns extracted from web logs, which focus the analysis on either resource-intensive activities or request-interval footprints. Those, are evaluated on the request sequences generated by individual clients, potentially labeled as malicious.

4. Edos in Self-Organizing Networks

In this section, the problem of the economic denial of sustainability in Self-Organizing Networks is discussed. To this end, the characteristics of these threats are identified and formalized. In addition, their causes and effect are described in the context of the Self-Organizing Networks.

4.1. EDoS and CRoWN indicators

The principal disparities between EDoS attacks and the rest of similar network circumstances are easily understood by considering four essential elements involved at the information communication processes, under the prior assumption that the relationships between the sources of the requests and the provision that must to solve them act as a client–server model. These traits are the clients (C), requests (R), workload that entails their resolution (W) and the network functions necessary for their processing (NF), which are grouped in the set of characteristics $CRoWN : \{C, R, W, NF\}$. From the analytical point of view, and as deduced from the discussions reviewed in the bibliography, the quantitative study of their behavior has attracted the interest at most of the researchers. Therefore, the following CRoWN indicators are considered in network event definitions (see Table 1): with regard to C, the total number of monitored clients that made requests nC and its distribution over time $nC(t)$; with respect to R, the average number of requests per client nR and its distribution over time $nR(t)$; for W is considered the average effort towards process the requests nW (bandwidth, computational cost, memory, etc. depending the use case) and its distribution over time $nW(t)$; and finally, regarding NF instantiated network function components nNF, its distribution over time $nNF(t)$ and the productivity of each of them $P(X)_{i=0}^{nNF}$, $0 \leq i \leq nNF$. Note that the method for calculating the last indicator depends directly on the functionality of the instantiated network functions. For example, if they act as Network-based Intrusion Detection Systems (NIDS), a possible productivity indicator is the number of alerts that each of them report. On the other hand, if they deploy bandwidth optimization capabilities, productivity may be the improvement they achieve. It is also important to highlight that the aforementioned indicators can be further extended in order to define variations of the threats described throughout this research, introducing alternative network situations, or enhancing the proposed solution. But in-depth addressing these issues is out of the scope of this paper, hence focusing on the features that most comprehensively recap the problems to be faced. With this purpose, the CRoWN indicators associated to a monitoring task m are summarized in the following abstract data organization:

$$CRoWN(m) = [nC, nC(t), nR, nR(t), nW, nW(t), nNF, nNF(t), P(X)_{i=0}^{nNF}] \tag{1}$$

From them five categories of network situations related with EDoS at Self-Organizing Networks are described: normal traffic [62], flash crowds [63], flooding-based Denial of Service [64], flooding-based Distributed Denial of Service [30], Workload-based EDoS and Instantiation-based EDoS (see Table 2). Two of them are legitimate (normal traffic and flash crowds) and the rest are malicious. Unlike Denial of Service, and Distributed Denial of Service, the EDoS threats satisfy the Network-based similarity condition. Because of this, the paper focuses on indicators related with the workload and productivity traits. Note that they are variants of the classical complexity-based DoS attacks [65] that target the implementation of algorithms at software-level, but adapted to exploit Self-Organizing Network features. On this basis, let the network monitorizations A and B expressed below and the following situations:

$$\begin{aligned} &CRoWN(A) \\ &= [nC_A, nC(t)_A, nR_A, nR(t)_A, nW_A, nW(t)_A, nNF_A, nNF(t)_A, P(X)_{i=0}^{nNF_A}] \end{aligned} \tag{2}$$

$$\begin{aligned} &CRoWN(B) \\ &= [nC_B, nC(t)_B, nR_B, nR(t)_B, nW_B, nW(t)_B, nNF_B, nNF(t)_B, P(X)_{i=0}^{nNF_B}] \end{aligned} \tag{3}$$

Definition 1 (*Flash Crowd Characterization*)**.** Assuming that A is the monitorization of the habitual and legitimate network behavior, B is defined as *flash crowd* when $nC_A \ll nC_B$ and the rest of indicators display similar values. Hence let the $\mathcal{U}$ infinite set of possible network monitorizations according with CRoWN, and the Fc subset of flash crowd events, they are formalized as:

$$Fc_B \leftrightarrow \{A, B \in \mathcal{U} : nC_A \ll nC_B, \sim \text{when other indicators}\} \tag{4}$$

Definition 2 (*Flooding-based DoS Characterization*)**.** Assuming that A is the monitorization of the habitual and legitimate network behavior, B is defined as *flooding-based Denial of Service (simple)* or DoS when $nC_A \sim nC_B$, $nC_A(t) \sim nC_B(t)$, $nR_A \ll nR_B$ and $nR_A(t) \nsim nR_B(t)$, i.e. when the number of clients is similar to the normal behavior, but a significant increase in the average number of requests is observed. Let the $\mathcal{U}$ infinite set of possible network monitorizations according with CRoWN, and the DoS subset of Denial of Service events, they are formalized as follows:

$$\begin{aligned} &DoS_B \\ &\leftrightarrow \{A, B \in \mathcal{U} : nC_A \sim nC_B, nC_A(t) \sim nC_B(t), nR_A \ll nR_B, nR_A(t) \nsim nR_B(t)\} \end{aligned} \tag{5}$$

Table 1
Summary of CRoWN indicators.

Trait	Indicator	Description
C	nC	Total number of clients.
	$nC(t)$	Distribution of clients over time.
R	nR	Average number of requests per client.
	$nR(t)$	Distribution of average number of requests per client over time.
W	nW	Total workload of processing requests.
	$nW(t)$	Distribution of workload of processing requests over time.
NF	nNF	Total number of instantiated network function components (VNFC).
	$nNF(t)$	Distribution of number of instantiated VNFCs over time.
	$P(X)_{i=0}^{nNF}$	Productivity of VNFC instance i.

Table 2
Network situations similar to EDoS attacks in SON.

Nature	Family	Category	Description
Legitimate		Normal	Habitual traffic.
		Flash Crowd	Multitudinous agglomeration of legitimate requests.
Malicious	Flooding-based	DoS	Basic DoS attack from a simple source.
		DDoS	DoS attack from multiple sources.
	EDoS	Workload-based	Enforcing auto-scaling by costly requests.
		Instantiation-based	Enforcing the instantiation of network functions.

Definition 3 (*Flooding-based DDoS Characterization*)**.** Assuming that A is the monitorization of the habitual and legitimate network behavior, B is defined as *flooding-based Distributed Denial of Service* or DDoS when $nC_A \ll nC_B$, $nC_A(t) \nsim nC_B(t)$, $nR_A \ll nR_B$ and $nR_A(t) \nsim nR_B(t)$, i.e. when the number of clients and requests significantly increase. Let the $\mathcal{U}$ infinite set of possible network monitorizations according with CRoWN, and the DDoS subset of Distributed Denial of Service events, they are formalized as follows:

$$DDoS_B \leftrightarrow \{A, B \in \mathcal{U} : nC_A \ll nC_B, nC_A(t) \nsim nC_B(t), nR_A \ll nR_B, nR_A(t) \nsim nR_B(t)\} \tag{6}$$

Definition 4 (*Network-based Similarity*)**.** As stated by Hoff [26,27], EDoS attacks pose great resemblance to the legitimate traffic. Henceforth, it is reasonable to assume that in this context, the number and distribution of clients and requests remain similar. So, let the A monitorization of the habitual and legitimate network behavior, B can only be categorized as Economic Denial of Sustainability attack if $nC_A \sim nC_B$, $nC_A(t) \sim nC_B(t)$, $nR_A \sim nR_B$ and $nR_A(t) \sim nR_B(t)$. Hereinafter this relationship is referred as *network-based similarity* (abbreviated as NB), so if it is satisfied, it is possible to state that A and B are network-based similar. It is expressed as follows:

$$NB(A, B) \leftrightarrow \{A, B \in \mathcal{U} : nC_A \sim nC_B, nC_A(t) \sim nC_B(t), nR_A \sim nR_B \text{ and } nR_A(t) \sim nR_B(t)\} \tag{7}$$

Note that the variations on W and NF traits reveal if B pose a threat, hence distinguishing it from the legitimate observations. From them, EDoS attacks on Self-Organizing Networks are classified in *Workload-based EDoS* attempts and *Instantiation-based EDoS* attempts. They are defined below.

Definition 5 (*Workload-based EDoS Characterization*)**.** Assuming that A is the monitorization of the habitual and legitimate network behavior, B is defined as *Workload-based EDoS attack* (abbreviated as W-EDoS) when A and B are *network-based similar*, $nW_A \ll nW_B$ and $nW_A(t) \nsim nW_B(t)$; i.e. when they pose strong resemblance at network-level, but the average workload per request significantly increased in B. In this way, the attacker exploits the auto-scaling capabilities of the protected system. Hence let the $\mathcal{U}$ infinite set of possible network monitorizations according with CRoWN, and the W-EDoS subset of Workload-based EDoS threats, the second is formalized as follows:

$$W\text{-}EDoS \leftrightarrow \{A, B \in \mathcal{U} : NB(A, B), nW_A \ll nW_B, nW_A(t) \nsim nW_B(t)\} \tag{8}$$

Definition 6 (*Instantiation-based EDoS Characterization*)**.** Assuming that A is the monitorization of the habitual and legitimate network behavior, B is defined as *Instantiation-based EDoS attack* (abbreviated as I-EDoS) when A and B are *network-based similar*, $nNF_A \ll nNF_B$, $nNF_A(t) \nsim nNF_B(t)$ and $P_A(t) \gg P_B(t)$; i.e. when they pose strong resemblance at network-level, there is a significant increase of the virtual instances belonging to a network function in B, but their overall productivity decrease. Hence, the attacker takes advantage of the actuation capabilities of the Self-Organizing Networks by deploying useless functionalities. Let the $\mathcal{U}$ infinite set of possible network monitorizations according with CRoWN, and the I-EDoS subset of Instantiation-based EDoS threats, they are formalized as follows:

$$I\text{-}EDoS \leftrightarrow \{A, B \in \mathcal{U} : NB(A, B), nNF_A \ll nNF_B, nNF_A(t) \nsim nNF_B(t), P_A(t) \gg P_B(t)\} \tag{9}$$

4.2. Impact of workload-based EDoS

Following the formalization stated in Definition 5, a W-EDoS attack is characterized by the execution of costly operations on servers hosted in a cloud provider. Server-side operations are produced by client requests that resemble legitimate traffic in terms of network indicators considering not only the number of clients, requests, workload or deployed network function components, but also their distribution over time. The effect of W-EDoS attacks is the need to scale-up or scale-out the deployed virtual instances (typically VNFCs [66]), by adding additional computational resources when the existing ones prove insufficient to ensure the desired quality of service, in conformance with the configured scaling policies. The major drawback of these auto-scaling enforcements is the associated monetary cost derived from the allocation of the new resources. Fig. 2 illustrates the scaling process of running an additional VNF triggered by a W-EDoS attack. In the depicted scenario, the VNF has insufficient nodes to process the workload (W), leading to a scale-up or scale-out strategy which accommodates a proper task processing, balanced in various instances of the same VNFC.

The idea behind the workload-based EDoS threats (W-EDoS) is to consume as many resources as necessary to trigger scaling operations

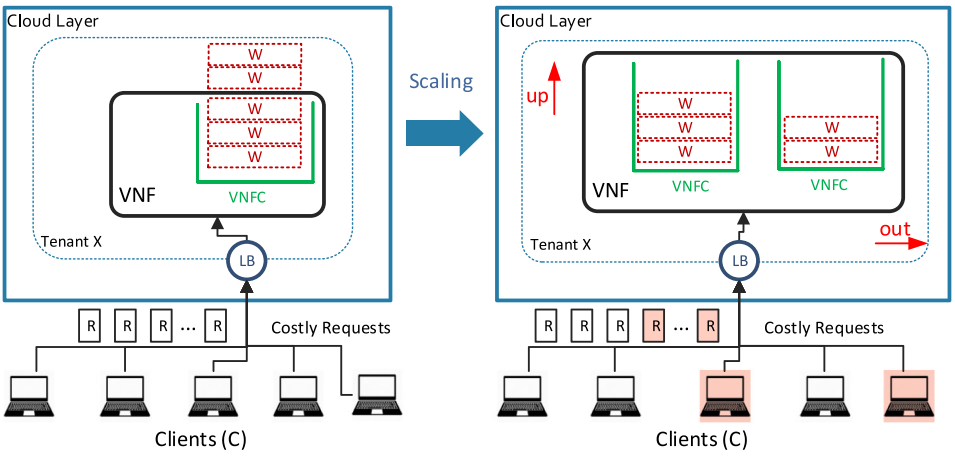


Fig. 2. Auto-scaling triggered by a W-EDoS.

managed on the cloud layer, but without causing the unavailability (or depletion) of the compromised nodes. Such workload behavior is exactly what distinguishes W-EDoS threats from conventional DDoS/DoS attacks, which can be inferred from the categorization of network situations previously introduced. Nonetheless, it is worth remarking that scaling operations originated from W-EDoS threats are triggered only when enough workload exists in the virtual instances, being it originated by legitimate or fraudulent processing requests

4.3. Impact of instantiation-based EDoS

Assuming the ability of the attacker to exploit a vulnerability that triggers the automatic deployment of virtual nodes belonging to the same VNF, an I-EDoS attack occurs when the cloud management software instantiates more VNF components than needed as a response to one or more requests ostensibly legitimate, thus increasing the cost of the hired cloud resources due to the rise on the number of instances, with a decrement on the average service productivity. This scenario fits with the principles described in Definition 6. Fig. 3 illustrates an I-EDoS attack where the cloud platform is assumed to have an auto-scaling vulnerability exposed to the attackers, serving it as an entry point for malicious requests. They generate in consequence the automatic allocation of additional VNFC instances in the network, but rather than being efficient they show scarce productivity levels. Lazy VNFC instances are thereby considered unnecessary by the network operator since they only raise the operational expenses, this way provoking a negative impact on the cost model of the service provider.

In contrast to W-EDoS, I-EDoS leaves aside the assumption that scaling requests are produced by real workload or resource consumption reported by the monitoring tool. For this reason, its detection focuses on the productivity measured at each running instance to find out if the hired resources are reasonably used. In normal conditions, consuming all the allocated resources per virtual instance is not an indicator of optimal productivity since it might compromise the quality of service, but the lowest productivity might lead to overspend the operational cost caused by the presence of idle resources. Thereby, finding the right productivity indicator is not straightforward and should be adapted to the particularities of the deployed VNFs. Under such premises, it can be noted that virtual instances that exhibit low productivity and are originated by fraudulent scaling requests, are labeled as malicious. Rather than depleting the available resources their only goal remains on incrementing of the operational cost.

5. EDiSON: EDoS detection in self-organizing networks

The method described in this section addresses the problem of distinguishing legitimate monitored situations from those related to attempts of Economical Denial of Sustainability in Self-Organizing Networks; because of this, hereafter it is refereed as EDiSON (acronym of EDoS Detection in Self-Organizing Networks). In order to define the scope of the tasks involved at the EDiSON design and development stages, the following design principles have been assumed on this proposal:

- The description of the network situations summarized in Table 2 and their descriptions assuming CRoWN indicators are considered. On this basis, the proposed method must be capable of successfully identifying W-EDoS and I-EDoS attacks, as well as distinguishing them from legitimate activities.
- A modularized design encompassing well-differentiated processing blocks is envisaged to accommodate an extensible architecture.
- During the development of the detection methods, the importance of the adaptation to changes in the monitored data has been considered. The analytical approach should accommodate to the fluctuations of metrics sampled at regular time intervals, without any assumption regarding its seasonality (i.e. when facing flash crowds) but considering its trend [67] for forecasting purposes.
- Self-Organizing Networks are complex monitoring scenarios where a large number of sensors collect information about the state of the network in real time. This information must be aggregated into observations that can be handled by high-level analytic tools. Although in the experimentation the impact of the data extraction granularity is briefly reviewed, the introduction of methods for its calibration is postponed to future investigations.
- To the best of the author's knowledge, there are no existing datasets of labeled EDoS threats available for model training. This issue leads to conduct an unsupervised learning approach for distinguishing discordant groups of observations extracted from the monitored random variables.
- The acquired knowledge must be notified with all the metadata required for their post-processing.

In the meantime, the architecture is constrained by the following limitations that are not currently addressed in this research:

- Adversarial methods based on imitation or identity theft [68,69] with the purpose of evade the proposed detection strategy are not considered.
- Even though the detection methods of conventional flooding-based DoS and DDoS attacks contribute with the EDoS threat mitigation, the detection of those attacks is out of the scope of

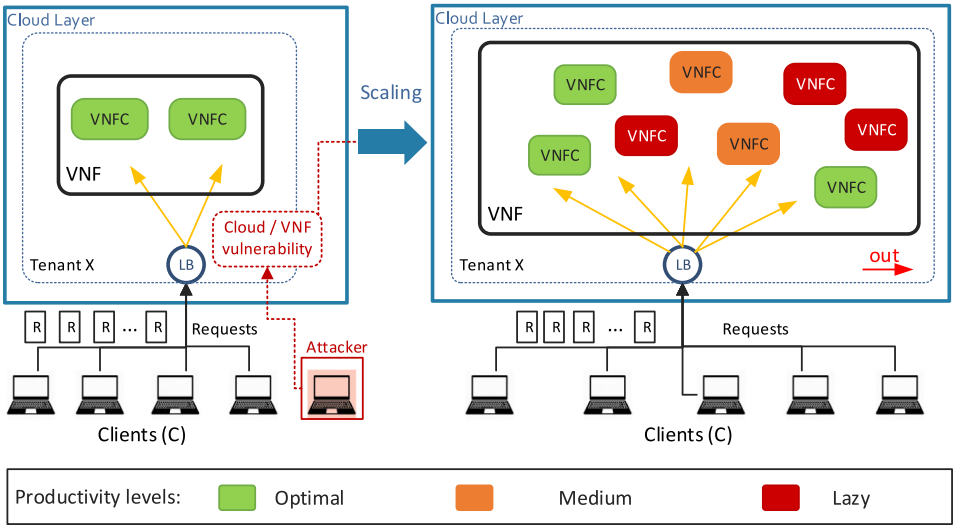


Fig. 3. Auto-scaling triggered by an I-EDoS.

this publication, but there is a vast bibliography that facilitates its recognition [33,70].

- The correlation and management of the alerts discovered [71,72] is out of the scope of this publication.
- The introduced approach does not deal with non-stationary monitoring environments [73], being its suitability widely supported by an important part of the research community and justified in publications like [74]. It plays an essential role towards avoiding situations related with the *concept drift* [75], which is postponed for future work.

5.1. Architecture

This section describes the EDiSON architecture, being illustrated in Fig. 4. It has been designed in conformance with the ETSI-NFV [12,76] and 5G architectures [10], where the decoupling of data and management planes makes it possible to distinguish the division between functional layers. The Physical sublayer is typically composed by Commercial-Off-The-Shelf (COTS) hardware on which the Virtualization Sublayer acts as a core component for the creation of virtual resources such as hosts, network links, storage elements and so on. On top of that, a Cloud Layer manages the automatic instantiation of Virtual Network Functions (VNFs) by interfacing with the Virtualization Sublayer, thus allowing the dynamic provision of resources needed to fulfill the agreed service levels. The deployed cloud environment interconnects VNFs through the underlying virtual network, composing in turn a forwarding graph where VNFs can also be chained with Physical Network Functions (PNFs), thereby creating one or more Network Services (NS) offered to the users. Cloud deployments are also isolated between customers to allow the coexistence of different network operators sharing the same physical resources, which is known as multi-tenancy support. The Cloud Layer gathers also an important number of metrics (e.g. usage or performance) measured on the instantiated resources allocated into the existing tenants. Besides that, sensors represent key components in the process of monitoring SON networks since they target to monitor customized network metrics, mainly at application-level. In this way, primary information collected by sensors and the cloud platform itself lies the basis to perform complex analysis at VNF level in the SON-Autonomic Layer, whose subcomponents are described in the forthcoming sections.

5.1.1. Data collection

In this module, the monitorization of the protected environment is carried out to extract raw metrics regarding the operational conditions of the virtual cloud. Notwithstanding the vast number of counters provided by most cloud platforms (i.e. OpenStack [77], AWS [78], Azure [79]), they might be insufficient when more complex analysis tasks are required. Therefore, a SON-driven approach can facilitate the extraction of more specific information gathered by sensors, hardly feasible at cloud-management-level, to enhance the analysis not only for EDoS attacks, but also for several other scenarios. The Data Collection module comprises the following submodules.

- *Application-level monitoring.* VNFs are monitored by sensors at operating system or application-level to extract specific metrics targeted to enable further analytical processes in the SON-autonomic layer. For instance; service response time, memory consumption per process, number of active connections, among other counters.

- *Virtual Infrastructure Monitoring.* Built-in cloud-monitoring services, such as OpenStack Telemetry (Ceilometer) [80], produce several metrics related to the virtual infrastructure (i.e. CPU, memory or network usage). Some of them are obtained periodically while others are generated when certain events are triggered (i.e. creation of a new virtual network).

5.1.2. Data aggregation

The monitorization of cloud deployments generate huge volumes of data, including counters, events, alerts, among others. To facilitate their analysis, data aggregation operations are applied to produce a single metric that summarizes a collection of sampled observations, thus reducing the existing data volume. Different data granularity levels should be also possible to configure, in accordance with the intended analysis. The Data Aggregation module comprises the following submodules:

- *Feature extraction.* A batch of raw observations generated by the sensors is grouped on regular time intervals, when the configured aggregation operation is applied on those values. This leads to the generation of a time-series of aggregated metrics; for example, the entropy degree in the W-EDoS analysis.
- *Virtual Resource Aggregation.* The virtual infrastructure metrics collected are also summarized with aggregation operations. Most of the polled metrics might be easily expressed as time series since they can be queried by the associated timestamp registered on the monitoring database. For instance, average CPU consumption in Ceilometer.

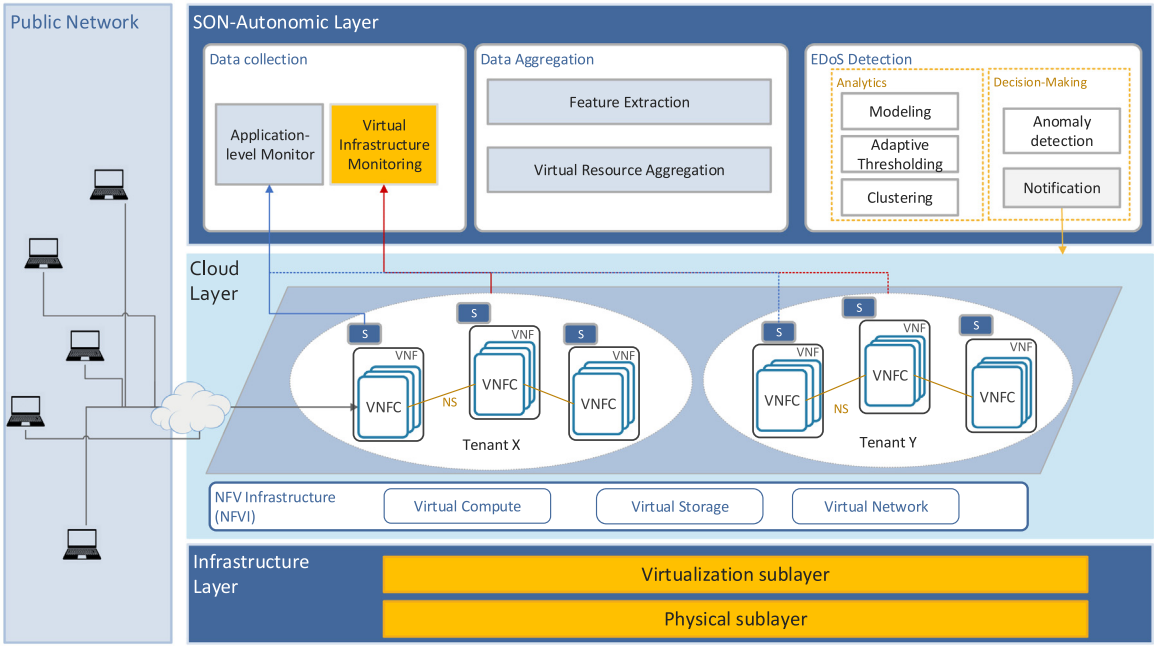


Fig. 4. EDiSON architecture.

5.1.3. EDoS detection

This module performs the essential analysis tasks for the detection of EDoS attacks, and it is composed by the following submodules:

- *Modeling.* With the time series of aggregated metrics, a forecasting model is created to estimate the next m values of the time series by the application of well-known prediction algorithms.
- *Adaptive Thresholding.* When compared with real observations, a prediction interval (PI) is generated based on the forecasting error measured at each observation. Unexpected behaviors are inferred when the value of an analyzed metric runs outside the boundaries of the upper or lower prediction thresholds.
- *Clustering.* Clustering methods are intended to separate groups of virtual instances by analyzing similarities on their productivity indicators. It leads to the distinction of a suspicious group of instances whose creation does not resemble a legitimate pattern.
- *Anomaly detection.* Discordant behaviors exposed by the forecasting and clustering processes are analyzed by a rule-based system to infer conclusions about possible anomalies. When rule matches are found, they are labeled either as W-EDoS or I-EDoS attacks depending on the analyzed scenario.
- *Notification.* The inference of anomalies is reported to the cloud-auto-scaling engine. It aims to prevent the creation of VNF instances that are targeted to overspend the use of virtual resources, and the operational cost in consequence.

5.2. Workload-based EDoS recognition

The following describes in detail the metrics required for W-EDoS detection, and how they are analyzed for identifying unexpected behaviors and deciding if they should be tagged as potential threats.

5.2.1. W-EDOS metrics

As stated in Section 4, the W-EDoS threat satisfies the *network-based similarity*, but unleash a significant variation in terms of workload (at both W and $W(t)$); in particular, concerning those metrics directly related with the auto-scaling capabilities of the hosts that support the task involved in solving requests [29]. Because of this, the proposed approach considers as W-EDoS detection metrics, the CPU consumption (X_{cpu}) and response-time at application-level (X_{app}). The first one measures the CPU usage at the operating system-level, and the second focuses specifically on each service request consumption. Although a direct relationship between X_{cpu} and X_{app} is expected, this is not always the case. The nature of the requests can lead to a different workload in each of these levels, in this way being able to force self-scaling [30]. Because of the proposed method lies on detecting unexpected behaviors by observing the aforementioned metrics, the first step is to discover the response-time variations. This is carried out by studying the disorder degree on the application-level metrics reported once the different tasks are fulfilled. As is commonly approached in the bibliography, this is addressed on the basis of [45,68], where the gathered information is correlated in terms of entropy. As indicated by Bhuyan et al. [70], among the various proposals, that defined by Rényi provides a general-purpose solution, so it was implemented for W-EDoS detection. Bearing in mind the reasons advocated by Bhuyan et al. [70], the performed research focused on information-based metrics, since it was proven that they: (a) were able to differentiate legitimate from malicious traffic using few attributes, (b) entail low computational cost, and (c) are applicable at different granularity observations. The Rényi entropy provides a generalization that allows to configure different information metrics in a simple way, just modifying a unique adjustment parameter (α). Hence $H_\alpha(X_{app})$ defines the Rényi entropy and α is the Rényi entropy order, $\alpha \geq 0$, $\alpha \neq 1$. The result is expressed as follows:

$$H_\alpha(X_{app}) = \frac{1}{1-\alpha} \log \sum_{i=1}^{n} P_i^\alpha \tag{10}$$

the limit for $\alpha \to 1$ particularizes the Rényi entropy as Shannon entropy:

$$H_\alpha(X_{app}) = \frac{-\sum_{i=1}^{n} P_i \log_a p_i}{\log_a n} \tag{11}$$

Note that given that Rényi entropies are in range $[0, \log n]$, they are normalized as $H_\alpha(X_{app})/\log n$. The implemented method considers the normalized version.

The study of the impact of X_{cpu} and X_{app} can be approached as a classical Statistical Process Control (SPC), where the metrics monitored over time are modeled as time series. In this context they are univariate, since it was previously assumed that it is not possible to

state that $X_{cpu} \perp\!\!\!\perp X_{app}$. Hence, the Rényi entropies calculated through the performed application-level observations relate with each other as $H_\alpha(X_{app})_{t=0}^{n}$ sequences, where:

$$H_\alpha(x_{app})_{t=0}, H_\alpha(x_{app})_{t=1}, \dots, H_\alpha(x_{app})_{t=n} \tag{12}$$

and the CPU consumption is represented as the following time series:

$$(x_{cpu})_{t=0}, (x_{cpu})_{t=1}, \dots, (x_{cpu})_{t=n} \tag{13}$$

Note that for simplicity and taking into account that the rest of the analytics for W-EDoS are similar for both time series, X_{cpu} and X_{app} are refereed as the random variable X, hence not assuming distinctions.

5.3. Workload-based unexpected behaviors

The W-EDoS detection method lies on deciding if the estimation $\hat{X}_{t=m}$ at m horizon, $H > n$, significantly differs from $X_{t=m}$. Hence it is required to forecast the time series of the current $X_{t=0}^{n}$ observations until a predefined horizon is reached, then being able to complete the comparison (see Fig. 5). Among the several approaches to this problem [81], the W-EDoS detection implements the Double Exponential Smoothing (DES) [82]. This decision has taken into account three conditions: firstly, non-seasonality is assumed as design principle, hence leaving aside more complex models like Holt-Winters [83], which are capable of adapting to these variations. On the other hand, the system must be efficient and requires short time series for warmup and modeling, which leaves aside autoregressive approaches such as ARIMA [84].

By definition, the Simple Exponential Smoothing (SES) algorithm may not operate effectively when there is a trend in the analyzed time series [85,86]. In addition to the smoothing constant α inherited from the SES model (in order to disambiguate this value with respect to the Rényi entropy degree, henceforth refereed as A, DES considers the constant γ related with the trend smoothing factor, so $0 < A), \gamma < 1$. It is calculated by the following recursive equations:

$$S_t = x_t + (1 - A)(S_{t-1} + b_{t-1}) \tag{14}$$

$$b_t = \gamma(S_t - S_{t-1}) + (1 - \gamma)b_{t-1} \tag{15}$$

where S_t is the level of the time series at t, and b_t is the trend. Note that when $A = 0$, DES becomes a naïve forecasting approach, and if A, $\gamma = 1$ it acts as SES. As is frequent in the bibliography, the base cases are the initializations $S_1 = \gamma_1$ and b_1, so:

$$b_1 = \frac{x_{t=n} - x_{t=1}}{n - 1} \tag{16}$$

and the forecasts are calculated as follows:

$$\hat{X}_{t=n+1} = S_t - b_t \tag{17}$$

$$\hat{X}_{t=n+m} = S_t - mb_t \tag{18}$$

The prediction intervals define the adaptive thresholds considering [87], and as suggested in [88]. They are expressed based on the ϵ_t prediction error based on the Mahalanobis distance at t, in particular when $t = m$.

$$\epsilon_t = \sqrt{(x_m - \hat{x}_m)^2} \tag{19}$$

In this way the following adaptive threshold is built:

$$PI = x_{t=n} \pm \eta\sqrt{\sigma^2(\epsilon_t)} \tag{20}$$

where the parameter η, $\eta \geq 0$ calibrates the restrictiveness of the prediction interval and m is the forecast horizon. In the context of the W-EDoS detector, the significance of the forecasting errors can be assessed according to Definition 7.

Definition 7 (*Workload-based Unexpected Behaviors*)**.** Let $X_{t=0}^{n}$ and its forecast $\hat{X}_{t=n+m}$ at horizon m, an observation $X_{t=n+m}$ is *workload-based unexpected* when $\epsilon_t \notin PI$ is satisfied, i.e. when $\hat{x}_{t=n+m}$ and $x_{t=n+m}$ differ significantly. The sequences of unexpected observations are referred as *workload-based unexpected behaviors*.

The persistence of observations tagged as *workload-based unexpected behaviors* establishes the duration of the possible threats. An example of this situation is illustrated in Fig. 6, where $X_{t=0}^{n}$ is analyzed and the prediction intervals are built. At the range of observations ($x_{t=41}$, $x_{t=44}$) they are overtaken, in this way displaying four *workload-based unexpected* observations ($x_{t=41}$, $x_{t=42}$, $x_{t=43}$ and $x_{t=44}$) and the *workload-based unexpected behavior* $X_{t=41}^{44}$.

Alerts related with W-EDoS are reported when at the same observation t, both detection metrics (X_{cpu} and X_{app}) are *workload-based unexpected*. Note that given the not $X_{cpu} \perp\!\!\!\perp X_{app}$ relationship, each of them may deduce a suspicious outlier, being $Symp(X_{cpu})$ related with abusive CPU workloads, and $Sympt(X_{app})$ being related with anomalous time-responses at application-level. The decision of pooling both symptoms lies on filtering the rebound effect of the adaptive thresholding method, where when a representative change occurs in the time series, the threshold takes a while to re-calibrate, which may lead to the emission of false positives as replicas. This is clearly illustrated in Fig. 7, where after discovering an outlier ($t = 47$), the prediction interval is erroneously calculated at the next monitorizations ($t = 48$ to $t = 65$). On the other hand, a $Symp(X_{cpu})$ may be triggered by anomalous decrements on the workload of the monitoring systems, which leads to an effect contrary to that caused by the W-EDoS. This is illustrated in Fig. 8(a) where $H_\alpha(X_{app})$ displays a discordant behavior because of changes at the disorder degree on the application-level; in particular the entropy decreased because most of the host supporting certain service are removed driven by a decrease of the demanded workload Fig. 8(b).

5.4. Instantiation-based EDoS recognition

The subsections below detail the metrics studied by the I-EDoS detection components, how they are analyzed and the decision-making task.

5.4.1. I-EDOS metrics

As indicated in Definition 4, I-EDoS situations are *network-based similar* with the normal and legitimate traffic, but entail outliers in terms of nNF, $nNF(t)$ and $P(X)_{t=0}^{nNF}$. It was also stated that the classical instantiation-based attacks are characterized by the emergence of a significant number of VNFC instances behaving in low-productivity. Therefore, before labeling a situation as possible threat, there must be observed a direct relationship between the new instances and their low productivity. This leads to consider a pair of I-EDoS detection metrics: the number of VNFC instances per observation Y, and their productivity Z; where Z is the set $Z = \{z_1 \cdots z_Y, Y \geq 0\}$ that defines the productivity of the different running instances of the same VNFC type $\{z_1 \cdots z_Y\}$ at certain t observation. In analogy to the W-EDoS, they are monitored over time and collected in the following times series:

$$Y_{t=0}, Y_{t=1}, \dots, Y_{t=n}; (Y_{t=0}^{n}) \tag{21}$$

$$Z_{t=0}, Z_{t=1}, \dots, Z_{t=n}; (Z_{t=0}^{n}) \tag{22}$$

On this basis, an observation at t, $0 \leq t \leq n$, is potentially malicious when Y_t has grown discordantly and $Z_t = \{z_1, \dots, z_{Y(t)}\}$ displays a group of instances that clearly behave at low productivity; in particular, the low-productive functions must be those that triggered the increase of Y_t, in this way satisfying $Y \perp\!\!\!\perp Z$. These situations are defined below.

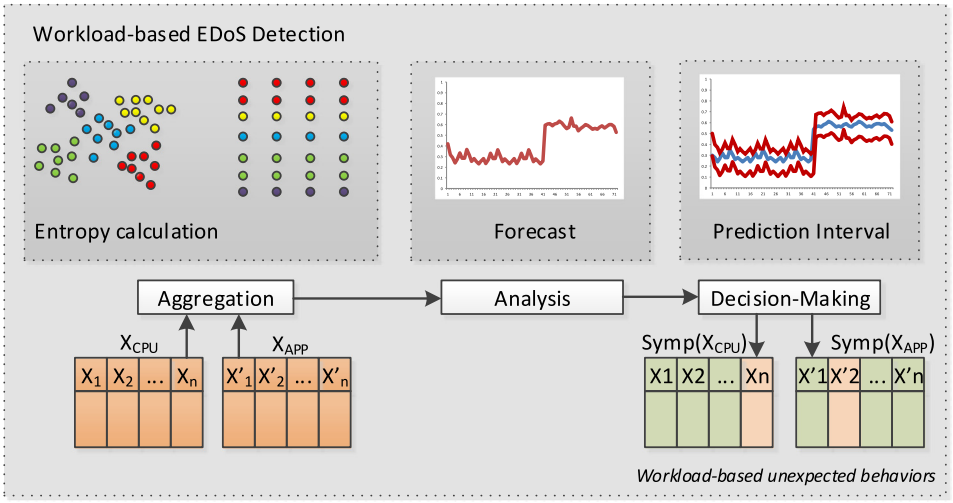


Fig. 5. Workload-based EDoS detection process.

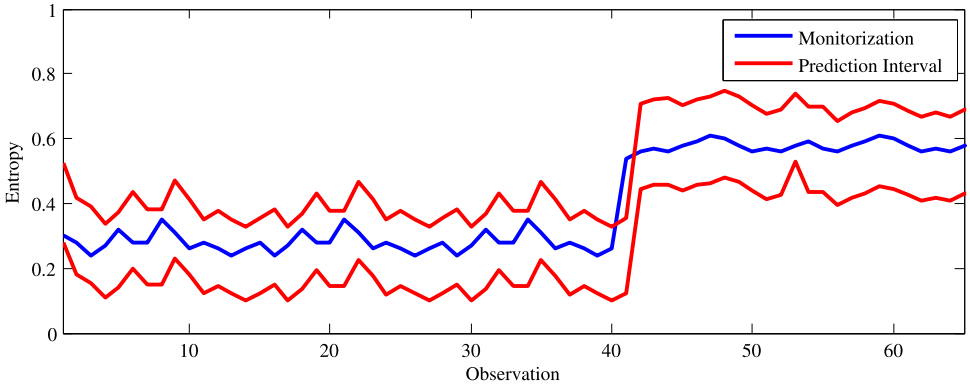


Fig. 6. Example of workload-based unexpected behavior.

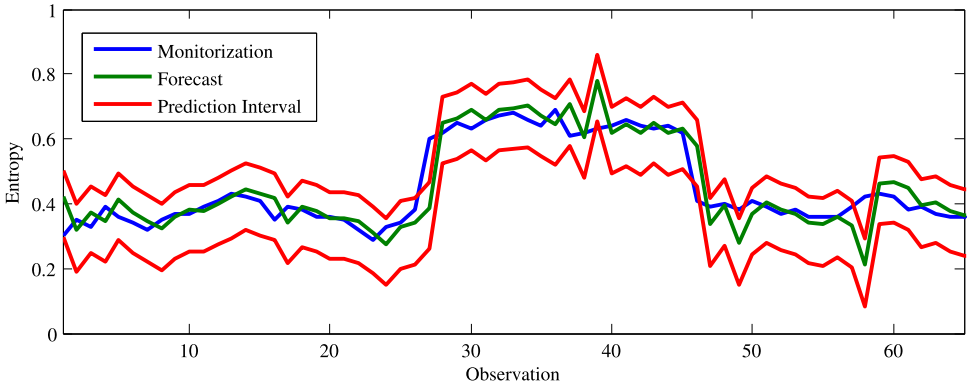


Fig. 7. Example of rebound effect.

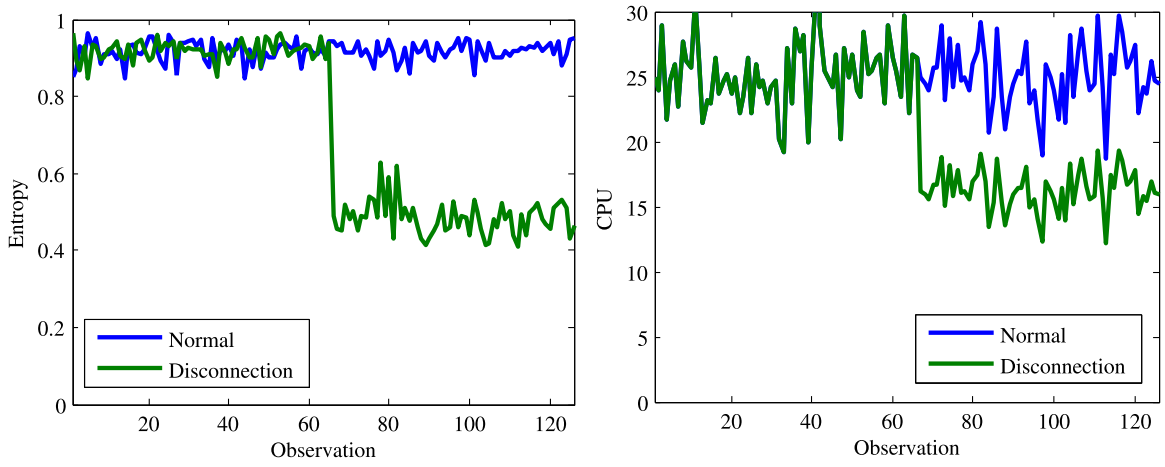
5.4.2. Novelty detection

The anomalous growth of the number of instances are studied similarly to the *workload-based unexpected behaviors*, i.e. by observing if their projection at m differ significantly from the observations at m. With this purpose, and in the grounds established at Section 5.3, the Double Exponential Smoothing (DES) [82] is implemented, and adaptive thresholds are built according with [88]. Consequently, a novelty in terms of number of instances Y at t is considered as potential trait of I-EDoS when a significant growth is observed in Y_t, which is formalized according to Definition 8.

Definition 8 (*Significant Growth*)**.** Let $Y^{n}_{t=0}$ and its forecast $\hat{Y}_{t=n+m}$ at horizon m, an observation $Y_{t=n+m}$ implies a significant growth when $Y_{t=n+m} \notin PI$ is satisfied, i.e. when $Y_{t=n+m}$ and $\hat{Y}_{t=n+m}$ differ significantly, and $Y_{t=n} < Y_{t=m}$.

5.5. Identification of suspicious network function instances

Once a self-organizing action at t has resulted in the instantiation of $Z_t = \{z_1, \ldots, z_{Y_t}\}$ new network capabilities, it is possible to detect if they are involved in an I-EDoS attack. According to the threat definitions discussed in Section 4, the typical behavior of the malicious



(a) Entropy decrease (b) Workload decrease

Fig. 8. Example of discordant entropy not related with W-EDoS.

instances is low-productivity. But low-productivity may be the pattern observed in most of the network functions deployed; therefore, it is assumed that the malicious instances should register significantly low-productivity with respect to the rest of the instances that operate on the monitoring environment at t. How to detect this situation is not a simple question. But despite detecting this situation is not trivial in many of the network scenarios, it can be addressed from different perspectives. Among them, the performed research has focused on those based on clustering, thus leaving other possible approaches for future investigation. At the experimentation, a density-based approach was implemented; in particular, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [89].

DBSCAN considers groups of observations based on the density of the K-nearest neighbors. This process distinguishes three kinds of o objects in the space: core objects, density-reachable and outliers. Let the object p, it is a core object if the ϵ-neighborhood of the object contains at least $minPts$ objects; where the ϵ-neighborhood of p is the space within a radius $epsilon$ centered at p. Because of this they are considered the pillars of dense regions. The objects within ϵ are considered directly density-reachable by p. On the other hand, a q object is reachable by p if there is a chain of objects $p_1, \dots, p_n$, where $p_1 = q$, $p_n = q$; and each p_{i+1} is directly density-reachable with respect to ϵ and $minPts$, $0 \leq i \leq n$, $p_i \in D$. Each cluster has at least a core point, being the rest of the points their periphery. Consequently, the objects within a cluster are interconnected, and if p is density-reachable by other object q, the second also is part of the same group. Observations non-reachable by objects in clusters are considered outliers [62]. DBSCAN has several advantages, which played an essential role in its selection as support for deploying an effective defense against I-EDoS, among them high efficiency, not requirement of previous estimations of the number of clusters to be defined, and noise tolerance. However, among other drawbacks it is worth mentioning its efficiency, which highly depends on the distances and similarity measures adopted. With the purpose of discover I-EDoS threats, DBSCAN is performed on each $Z_t = \{z_1, \dots, z_{Y_t}\}$ set, where each productivity z_i is an object p_i, so $Y_t = n$. The following ϵ distance based on the Mahalanobis divergence was implemented:

$$\epsilon = \sqrt{(z_i - z_j)^2} \tag{23}$$

where $0 \leq i, j \leq n$ and $i \neq j$. The resultant set of K clusters is $C_t = \{c_1, \dots, c_k\}$, $K \geq 0$ were $c_i = \{z_r, \dots, z_s\}$, $0 \leq r, j \leq n$, i.e. $z_r, \dots, z_s$ are network functions with similar productivities. The members of C_t can be increasingly ordered according to the average productivity of their members, being each $\bar{c_i}$ the mean of $z_r, \dots, z_s$. The arranged list

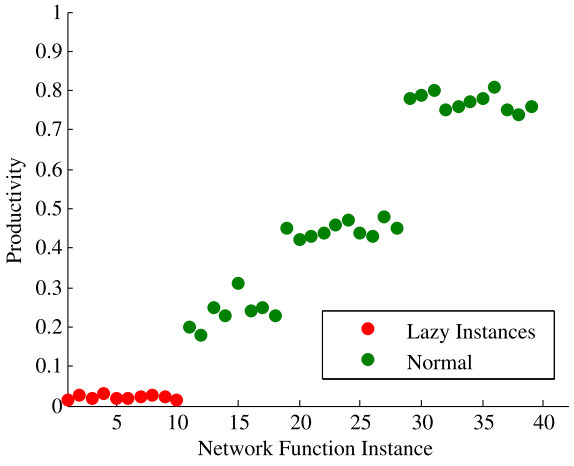


Fig. 9. Example of lazy group of instances.

is expressed as $s(C_t) = [c_1, \dots, c_K]$, $\forall c_i : c_j \; 0 \leq i \leq j \leq K$ is satisfied $\bar{c_i} \leq \bar{c_j}$. The clusters relevant for the I-EDoS detection are tagged as *lazy groups* and their members are termed *lazy instances*, which are defined in Definition 9. An example of them is illustrated in Fig. 9, where it is possible to observe a subset of instances which productivity is significantly low.

Definition 9 (*Lazy Instances and Groups*). Let $Z^n_{t=0}$, for each $Z_t = \{z_1, \dots, z_{Y_t}\}$ clustered into $C_t = \{c_1, \dots, c_k\}$, $K \geq 0$, and sorted resulting $s(C_t) = [c_1, \dots, c_K]$; c_1 is defined as the *lazy group* of network function instances, hence $\forall c_j : c_j \in s(C_t), 1 < j \leq K$ and the inequation $\bar{c_1} \leq \bar{c_j}$ is satisfied. Each member of c_1 is considered a *lazy instance*, and will play a relevant role when deciding if the protected environment is suffering an I-EDoS attack.

5.6. Instantiation-based unexpected behaviors

Ideally, it is expected that when an I-EDoS attack occurs a discordant increase in the number of instantiated network functions is observed. Consequently, and assuming that the attack is launched at the observation registered at t, a *significant growth* (see Definition 8) in the number of instances must be detected on Y_t. On the other hand, the network functions triggered by the attacker must behave with low

productivity. Thus, most of them must be part of the lazy group of $Z_t = \{z_1, \ldots, z_{Y_t}\}$, in this way being lazy instances. When both conditions are satisfied, a potential I-EDoS threat was recognized at t.

But it is important to bear in mind that in real circumstances, these observations may occur asynchronously. This is because each type of network function taken into account in Z_t may require a different adaptation period, which is usually referred as warm-up time. Therefore, they are not expected to be productive until the warm-up time expires. A clear example of this situation can be comprehensively illustrated with the following example: let a SON configured to deploy intelligent agents for load balancing purposes similar to those introduced in [90]. These actuators are triggered once the number of clients connected to certain service overtake a previously defined static threshold. The productivity metric considered for I-EDoS detection is the Key Performance Indicator (KPI) that aggregates the server throughput in terms of GB/s, i.e. the amount of data transferred to and from the load balancer. These intelligent actuators require a warm-up time to collect reference data and build prediction models. Now suppose that the intruder knows the threshold that triggers the generation of new agents, and that by registering malicious clients, it is able to cause an I-EDoS attack. In the observations where this occurs, a *significant growth* of the number of instances is registered, but they are not able to report productivity, since they are initializing their internal network usage models. Because of this, the definition of the *instantiation-based unexpected* behaviors related with I-EDoS threats must take into account the creation date of the instances and from when they are expected to begin to be productive. This leads to Definition 10, which establishes the condition that must be satisfied prior to report possible I-EDoS incidents (see Fig. 10).

Definition 10 (*Instantiation-based Unexpected Behaviors*)**.** Let the set of instances $Z_t = \{z_1, \ldots, z_{Y_t}\}$ observed at t, clustered into $C_t = \{c_1, \ldots, c_k\}$, $K \geq 0$, and sorted resulting $s(C_t) = [c_1, \ldots, c_K]$. They are *instantiation-based unexpected* when there was registered a *significant growth* at the creation date of most of the members of their *lazy group* c_1. The set of instantiation-based unexpected observations over a monitorization process is referred as instantiation-based unexpected behavior.

Therefore, when some observation in terms of the detection metrics (Y, Z) is tagged as *instantiation-based unexpected*, the proposed systems report a possible I-EDoS situation $Symp(Y, Z)$. The persistence of events tagged as *instantiation-based unexpected behaviors* establishes the duration of the possible threat.

6. Experiments

This section describes the SON scenario where the Cloud layer is complemented with the SON Autonomic components, thus extending the analytical capabilities of Cloud-platforms with more advanced features targeted to mitigate EDoS attacks. In addition, the performed tests carried out to detect W-EDoS and I-EDoS attacks are described in detail.

6.1. TestBed

The experimentation testbed was deployed to match the architectural layers defined in Section 5.1. This SON-oriented scheme is illustrated in Fig. 11. The Cloud Layer has been implemented with OpenStack [77], a widely used opensource platform to manage the lifecycle of small and large-scale cloud environments. OpenStack has been deployed on two nodes: controller and compute (Nova). Each of them runs core OpenStack functionalities in conformance with the NFV infrastructure layer. Moreover, additional features oriented to include auto-scaling mechanisms have been included to showcase EDoS attacks. The controller node hosts also the networking service (Neutron), Nova essential features, and RabbitMQ [91], the message broker software that allows the inter-process communication between the OpenStack services. The compute node hosts also the Clustering (Senlin) [92] and Orchestration (Heat) services required to configure scaling policies. In addition, the compute node hosts the Telemetry service, intended to measure cloud-platform statistics with monitoring, scaling and billing purposes. All the OpenStack services are interconnected through a private management network, in this way making possible to manage the on-demand instantiation of virtual resources to conduct the experimental EDoS scenarios. The SON-Autonomic Layer is implemented by a combination of custom and well-known opensource tools matching the modular design.

Data Collection is carried out at application-level by a Python client–server application composed by the Raw Data Pollster Agent (RDPA) running on each instance (client-side), and the Central Collector Node (CCN) on the server-side. They are communicated by means of HTTP REST messages pushed by the RDPA on regular time-intervals. Apart from that, the virtual infrastructure monitoring is natively supported by OpenStack Ceilometer which gathers an extensive set of meters [93] related to the cloud deployment. Depending on the type of measurement, notification or polling methods are implemented.

Data Aggregation is performed by two Python modules. On the one hand, the Feature Extraction (FE) module fetches from the CCN a batch of client-request processing times observed on one-second intervals. Then, the FE computes the Rényi entropy of the sampled observations, thus creating a time series of measurements. On the other hand, the Virtual Resource Aggregator (VRA) queries the Ceilometer REST API and extracts CPU usage counters for each analyzed virtual instance on one-second intervals, thus leading also to the creation of a time series.

EDoS detection is implemented on two major functional blocks: Analytics and Decision-Making (DM). The EDoS Analysis (EA) is a Java module implementing the Double Exponential Smoothing algorithm which results are stored in data structures, and the forecasted metrics are taken into account to elaborate the adaptive thresholds. In addition, EA implements the Clustering capabilities of the Weka libraries [94], and the Decision Maker (DM) is implemented in Drools [95] as a rule-based expert system whose conclusions (alerts) are written in JSON files to match the Notification stage.

Finally, the client-side of the experimentation is performed by Python multi-threading modules capable to emulate a configurable number of clients. It also allows the possibility to control the number of requests and their connection rate to generate network traffic adapted to the experimental W-EDoS and I-EDoS scenarios.

6.2. Evaluation methodology

The methodology used to conduct the experimentation is explained in the forthcoming sections, which distinguishes tests for assessing the W-EDoS and I-EDoS recognition capabilities of the proposal.

6.2.1. W-EDoS detection evaluation

The evaluation of W-EDoS analyzes the Rényi entropy degrees measured on the server-side response times observed in one-second intervals, compared against the CPU consumption at operating-system-level. To this end, a client–server application has been implemented as described below.

- *Server-side software.* A RESTful HTTP application was written in Flask [96] due to the simplicity that entails this type of web service, hence exposing eight endpoints with different computational costs each in terms of processing time. When requested, each of them performs list-sorting operations with randomly generated values, varying on the list size and the number of iterations. The HTTP endpoints and their average execution times measured for 1000 executions are described in Table 3. All the HTTP methods support a GET parameter (id) used to distinguish client requests when they are generated from the same host. This application is

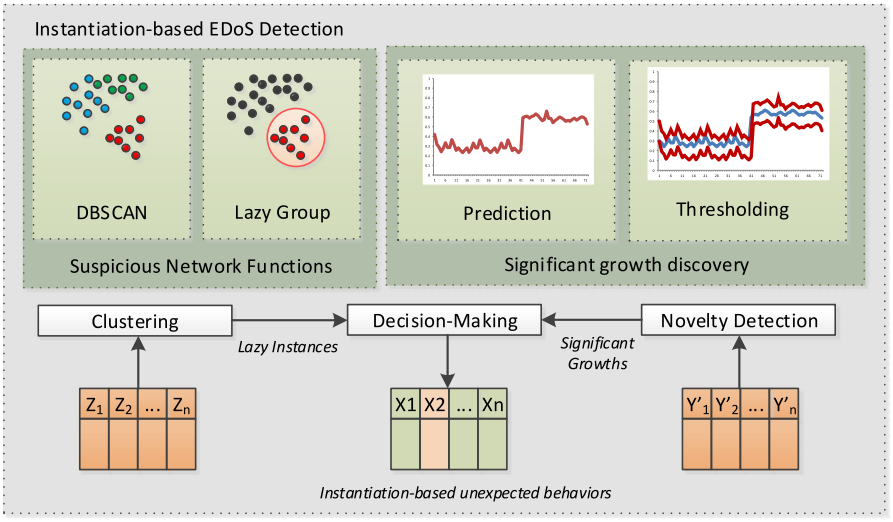


Fig. 10. Instantiation-based EDoS detection process.

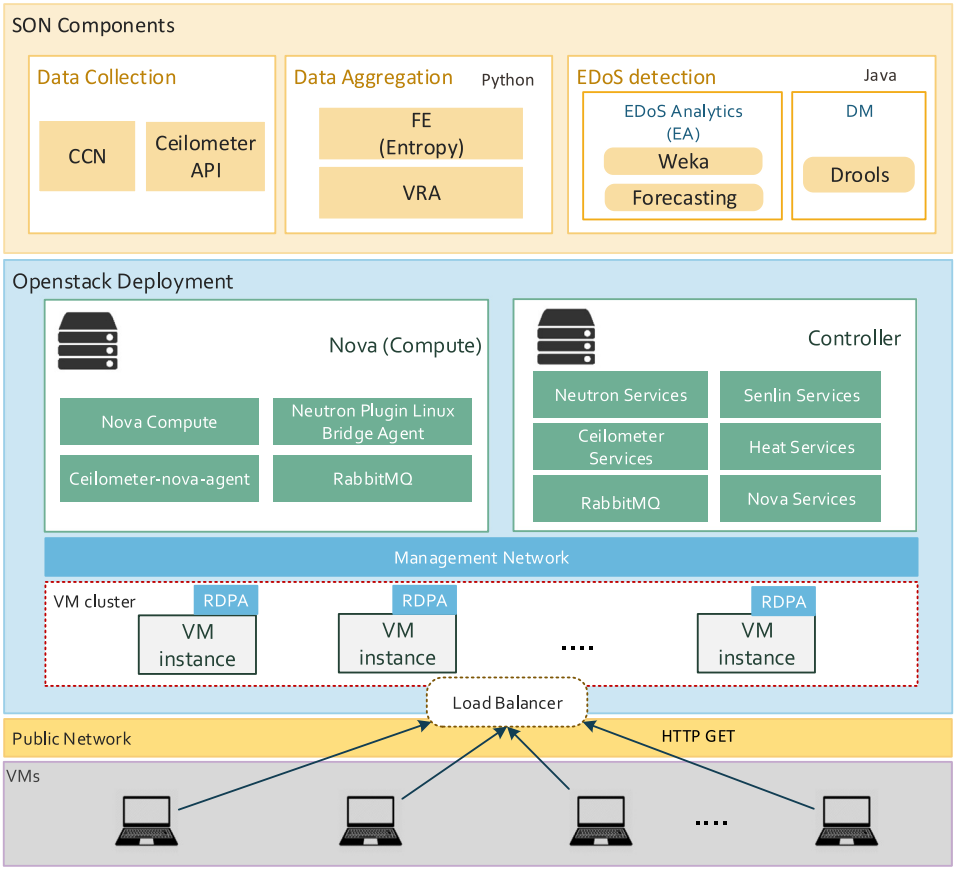


Fig. 11. Testbed.

hosted in a server deployed as an OpenStack Compute instance running on a single-tenant private cloud.

- *Client-side software.* A Python application instantiates each client as a standalone thread, which in turn send HTTP requests to the web service following a random Poisson distribution being lambda (λ) the expected number of occurrences in a given time interval. Such distribution is suitable to generate legitimate web requests as often assumed by the bibliography [74]. Thereby, normal traffic conditions and EDoS attacks can be conducted with multiple clients running in parallel. Taking advantage of the HTTP GET id parameter of each endpoint, a single node can impersonate multiple clients. For example, two HTTP requests launched from the IP 192.168.5.48 with URIs /3?id=145 and /3?id=213 would be interpreted as originated from different clients with ids 145 and 213 respectively. So that, response times measurements on the server are associated to those client ids.
- *Cloud Auto-scaling.* It has been configured an auto-scaling policy for launching a new virtual instance of the web service when

Table 3
HTTP endpoints and average CPU processing time.

URI	Avg. CPU time (ms)
/1	18.56
/2	21.58
/3	24.64
/4	27.81
/5	31.06
/6	33.72
/7	36.73
/8	226.04

Table 4
Normal traffic attributes.

Feature	S1	S2	S3	S4
Number of clients (C)	1000	1000	1000	1000
λ requests per second (px)	50	60	70	80
Number of requests (R)	9000	10800	12600	14400

Table 5
Malicious requests per scenario.

Intensity	S1	S2	S3	S4
1%	90	108	126	144
5%	450	540	630	720
10%	900	1080	1260	1440

the average CPU consumption runs above 60% in a one-minute interval.

- *W-EDoS attack scenarios.* The detection of a W-EDoS attack is carried out by examining the variation of the threat features in terms of the number of clients, request rates and percentage of malicious connections triggered from compromised nodes. Normal traffic conditions were considered when clients launched requests to endpoints 1 to 8, randomly chosen for each connection; whereas W-EDoS attack traffic is targeted to execute only endpoint 8 since it produces the costliest operations at server-side and the maximum CPU overhead in consequence. Table 4 summarizes the parameters considered to define the W-EDoS scenarios for normal network traffic, and Table 5 shows the malicious traffic volume determined by the attack intensity on each scenario (S1 to S4).
- *SON W-EDoS detection and notification.* The SON layer follows the principles of Definition 7. If a significative variation between the observed and forecasted entropy degree of the request processing times ($H_\alpha(X_{app})$) matches with an increment of the CPU consumption at operating system-level (X_{app}), the autonomic layer infers a W-EDoS attack in the analyzed web server and triggers an alert intended to prevent the auto-scaling of a new virtual instance since it does not resemble a legitimate origin.

6.2.2. I-EDoS detection evaluation

The evaluation of I-EDoS aims to evaluate inconsistencies between the number of deployed VNF instances compared against the productivity measured on them. This is implemented as follows:

- *VNF and productivity.* A single-component VNF is deployed as a cluster of VNFC instances. That VNFC is implemented as a HTTP REST web service which exposes a unique endpoint with an average response-time of 27.89 ms. The VNFC is packaged as an Ubuntu-based OpenStack Glance image, used as a template for launching additional instances in the clustered VNF when scaling is carried out. The performance of this VNFC has individually tested with Httperf [97], a well-known benchmarking open-source tool to measure web performance under different workload conditions whose results are shown in Table 6. Therefore, the ideal performance of each running instance is reached when Medium and Optimal productivity levels are observed at any time of operation.
- *Cloud environment configuration.* To showcase the evaluation scenarios, an OpenStack cluster has been deployed and auto-scaling rules have also been configured to dynamically scale-out and scale-in the cluster size. It has been considered a minimum size of 2 instances and a maximum size of 12. A Neutron load balancer has also been deployed, which implements a round-robin policy to handle the traffic request to be distributed across the active cluster nodes.
- *Cloud auto-scaling in and out policy.* To scale-out the cluster, a new VNF instance is launched when the cluster average CPU consumption runs above 80% in a one-minute interval. Likewise, when the average CPU usage runs below 40% a virtual machine instance is removed, following the "youngest first" deletion policy. In addition, a "best effort scaling" is enabled to prevent breaking the size limitations (minimum and maximum) of the cluster.
- *Normal traffic conditions.* A normal scenario has been defined by modeling a traffic profile based on a variable number of expected HTTP connection rate (λ), measured in requests per second (px), in different time slots, elapsing a total time-window of three hours. The cloud platform will then determine the right number of VNF instances based on the auto-scaling rules, ranging from 2 to 12 according to the cluster configuration. A graphical representation of the traffic workload is illustrated in Fig. 12.
- *I-EDoS attack scenarios.* Auto-scaling is configured in OpenStack Senlin, which is also integrated with Heat and Ceilometer to define cluster-related configuration templates with auto-scaling policies. Ceilometer alarms are thereby configured to trigger notification when some virtual-infrastructure-related counters run outside pre-defined thresholds, such as with CPU or memory usage. Based on the dependance on Ceilometer measurements, the I-EDoS attack is conducted by poisoning the counters gathered by Ceilometer related with CPU consumption since scaling policies rely on their values. To this end, it is assumed the ability of the attacker to gain access to the RabbitMQ message broker, either with legitimate authentication credentials, such as a valid username and password, or by exploiting a common Telemetry-related vulnerability, such as the one reported in CVE-2016-9877 [98] where access to the messaging service can be granted only with a legitimate user regardless of the provided password. Once connected to the message bus, the attacker fetches legitimate CPU usage counters obtained by Ceilometer (represented as JSON objects) and injects a manipulated version of the original message with random CPU usage counters above 90 percent. As a result, and maintaining the same normal traffic conditions described in the previous section, the Heat engine performs scaling-out decisions based on unrealistic data, creating in the meantime more instances (cluster nodes) than needed to process the traffic workload.
- *SON I-EDoS detection and notification.* The SON autonomic layer forecasts the number of virtual instances in the OpenStack cluster with the Double Exponential Smoothing (DES) algorithm. Meanwhile, the adaptive thresholds are built to look for situations where the number of existing instances is higher than the predicted values. The EDoS detection module targets these situations of unexpected growth to carry out DBSCAN density-based clustering based on the productivity measured on each virtual instance. It leads to the distinction between productive and lazy instances, which corresponds with Definition 10, triggering in consequence an instantiation-based unexpected behavior alert.

Table 6
VNF performance and metrics.

Connection rate (cons/sec)	Reply time (ms)	Reply rate	(replies/sec)	Connection errors (%)	Avg. server CPU (%)
5	31.9	5	0	22.9	Lazy
10	30.7	10	0	39.1	
15	29.5	15	0	49.3	Medium
20	27.2	20	0	57.5	
25	25.9	25	0	66.1	Optimal
30	25.3	30	0	74.3	
35	28.2	35	0	81.6	
40	133.1	38.2	4	92.1	Overload
45	736.2	8.1	89	99.6	

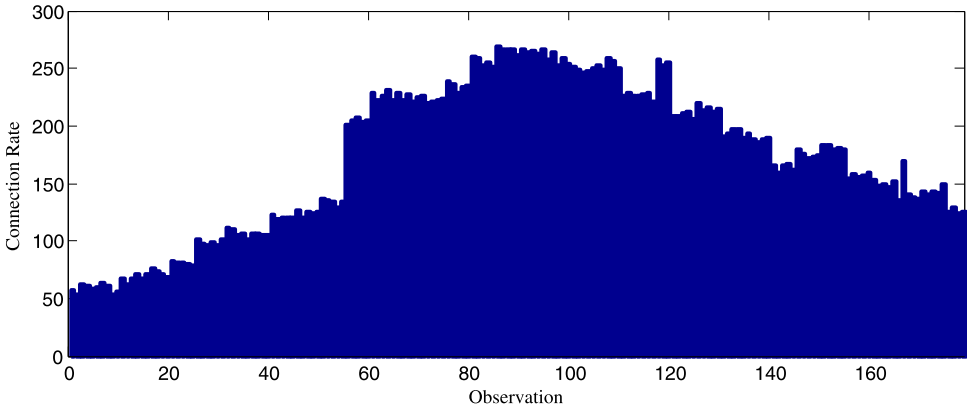


Fig. 12. Traffic workload for I-EDoS experimentation.

7. Results

The effectiveness of EDiSON when analyzing EDoS threats and legitimate situations are described and discussed below.

7.1. W-EDoS detection

As indicated in Section 6, the evaluation of the W-EDoS detector is based on studying their behavior when varying the features of the threats and the legitimate environment, the impact of the Rényi entropy degree and the improvement obtained by refining the analysis of the X_{App} evolution with the study of X_{CPU}. They are described throughout the rest of this subsection.

7.1.1. Attack distribution and requests

The results obtained when varying the request rate and the intensity of the W-EDoS threats are summarized in Fig. 13(a). With the purpose of facilitating their understanding, the intensity of attacks has been measured at the interval 1%, 5%, 10%, where the percentage indicates the distribution of malicious requests per observation. In addition, four different scenarios have been studied based on the number of average requests per second (px) launched by each client: $\{50, 60, 70, 80\}$. In Figs. 13(a)–13(d) the ROC curve obtained at each experiment is illustrated, where it is assumed the K value of the prediction interval as the main parameter that adjusts the W-EDoS detection sensitivity. The obtained effectiveness is summarized in Table 7 and Fig. 13(e). The worst results are observed when clients perform an average of 60 requests per second, as well as when the attack is present in at least 1% of the request, being the trapezoidal approximation to the Area Under the Curve (AUC) 0.901, and in the best case under this setting, the True Positive Rate (TPR) 0.816 and the False Positive Rate (FPR) 0.15. On the opposite, the W-EDoS detector best performs when the sensors submit an average of at least 80 requests per sensor and the attack poses 10% intensity. In this case the AUC is 0.995, and the best TPR is 1 and FPR is 0.01. From the plots and the summarized data, it is possible to conclude that, as the number of requests per node grows, the accuracy of the system improves. This denotes that under such circumstance, the prediction strategy has a greater capacity to model the characteristics of the monitored traffic, and therefore to act more accurately. On the other hand, it is possible to affirm that the greater attack intensity, the higher its visibility, resulting in much more obvious entropy variations and CPU overload. In general terms, the obtained accuracy demonstrates the ability of the W-EDoS detection method of operating on scenarios similar to those proposed in the testbed.

7.1.2. Entropy degree

In the performed experimentation, the Rényi entropy degree that configures the W-EDoS detectors has proven to play an essential role in the quality of the decisions made. With evaluation purposes, and in view of the results observed in Section 7.1.1, a testing scenario with average rate of 60 requests per client/second, and attacks originated from 5% of the clients to be served, has been considered. It is worth mentioning that this configuration represents an intermediate point between the circumstances that provide greater visibility of the threats, and those that hinder their identification. The results obtained are illustrated in Fig. 14, where it is possible to distinguish the impact of α in the ROC space (Fig. 14(a)), the summary of the precision obtained in terms of TPR and FPR (Fig. 14(b)), and an example of its smoothing effect in the time series to be analyzed (Fig. 14(c)). Note that at this study the range $1 \leq \alpha \leq 5$ has been studied. The minimum value supported by the Rényi entropy is $\alpha = 0$, which has not been taken into account since it coincides with the expression of Hartley (i.e. max-entropy) $H_\alpha(X) = \log_2 n$, which due that normalized values are adopted, always returns 1. From $\alpha = 5$, the observed AUC is less than 0.5, so it is inferred that the obtained values correspond to a random behavior. From Fig. 14(a) and Table 8 it is possible deduce that the best fit is $\alpha = 1$, where AUC = 0.955, TPR = 0.911 and FPR = 0.04. Hence, as the degree increases,

Table 7
Summary of results of the W-EDoS detector.

	Average Requests per Second (px)											
	50 px			60 px			70 px			80 px		
Attack	AUC	TP	FP	AUC	TP	FP	AUC	TP	FP	AUC	TP	FP
1%	0.9069	0.8183	0.08	0.9013	0.8161	0.15	0.9175	0.83654474	0.07	0.9345	0.8492	0.05
5%	0.9557	0.9113	0.04	0.9432	0.8195	0.15	0.9631	0.9347	0.05	0.9603	0.9434	0.11
10%	0.991	0.9889	0.01	0.9631	0.9843	0.02	0.9908	0.989	0.01	0.995	1	0.01

Table 8
Summary of results when varying α.

	Rényi entropy degree				
Measure	α=1	α=2	α=3	α=4	α=5
AUC	0.9557	0.8692	0.7574	0.6072	0.5138
TPR	0.9113	0.8388	0.7147	0.7041	0.5568
FPR	0.0400	0.1600	0.3200	0.6500	0.6400

Table 9
Summary of results considering X_{App} and X_{CPU}.

		Effectiveness measurements		
Setting	Px	AUC	TPR	FPR
X_{App}	50	0.6942	0.5152	0.10
	60	0.8189	0.7028	0.12
	70	0.7319	0.7021	0.32
	80	0.7766	0.7541	0.29
X_{App} and X_{CPU}	50	0.9557	0.8195	0.15
	60	0.9432	0.9113	0.04
	70	0.9633	0.9346	0.05
	80	0.9603	0.9434	0.11

the smoothing of the resulting time series decreases, which entails a greater tendency to report false positives, (see Fig. 14(c)). Because of this, the worst measured value has been observed when $\alpha = 5$, resulting in AUC=0.513, TPR=0.556 and FPR=0.64, which obviously are far from achieving a desired accuracy. Consequently, it is possible to conclude that the degree of Rényi entropy has a direct relationship with the successfulness of the performed deployment; and that in order to mitigate the false positives problem, it is advisable to select low values, thus reducing the impact of the temporary outliers inherent to the monitoring tasks on network environments.

7.1.3. Simple X_{App} and refinement by X_{CPU}

In order to evaluate the refinement of the analysis stage based solely on the X_{App} achieved by studying X_{CPU}, a new test has been performed. Firstly, it only attempted to detect W-EDoS based on X_{App}, then assuming the combination of both random variables. The obtained effectiveness has been assessed based on the number of average requests per second (px) made per client: $\{50, 60, 70, 80\}$, and by configuring 5% of the clients to be served as attackers. The results are illustrated in Fig. 15, where Fig. 15(a) indicates the precision obtained to consider only X_{App}; Fig. 15(b) displays the results when combining both random variables; and Fig. 15(c) plots its comparison, which contents are summarized in Table 9. The best results in the simple X_{App} tests were AUC=0.7766, TPR=0.5152 and FPR=0.1, when 80 px. On the opposite, the combined setting provided AUC=0.9633, TPR=0.9346 and FPR=0.05 when 70 px. Note that as discussed in Section 7.1.1, the greater numbers of average requests per second tend to facilitate the detection task. In the light of these results, it is possible to deduce that the W-EDoS detection when only considering X_{App} was not feasible, since the AUC fall behind those provided by the combined approach, as well as a significantly greater false positive rate was displayed. This is because of the situations previously described in Section 5.3, where it was explained the rebound effect of the forecasting models, and other legitimate situations that may lead to false positives.

7.2. I-EDoS detection

This section describes the results observed when evaluating the I-EDoS detection capabilities of the proposed defensive scheme. In order to facilitate the understanding of the performed research, an illustrative case of study is detailed, where the differences between normal activities and I-EDoS are displayed, and the detection process is demonstrated. In addition, the effectiveness of the approach when dealing with attacks of different intensity is discussed.

7.2.1. Case of study

The testing scenario evaluates the effectiveness of the I-EDoS detection in the experimental testbed, where a single-component VNF is deployed as a cluster of virtual machines, each representing an instance of the VNFC. The comparison between the number of VNFC instances deployed as a result of auto-scaling decisions in both normal and attack traffic are presented in Fig. 16, where it is noticeable that the deployment of a higher number VM instances has been caused by an I-EDoS attack. Under normal conditions, the cluster scaled up to a maximum of 10 instances when the highest traffic workload has been reached; but when the I-EDoS attack is launched the cluster is forced to scale up to the maximum size of 12 running elements. Then, the number of virtual instances have been compared with the adaptive thresholds estimated for normal (Fig. 17(a)) and attack conditions (Fig. 17(b)), where deviations between the forecasted and real measurements are noticed. For instance, at observations 12 or 36 (normal traffic conditions), and at observations 14 or 34 (attack). Besides the identification of a suspicious growth of the VNFC instances, the I-EDoS detection strategy relies on the productivity measurements, whose dispersion graphs are compared. Under normal conditions (Fig. 18(a)), the lower number of instances leads to a more uniform distribution of the productivity exposed by the VNFs which in general reach higher values and less dispersed measurements than its counterpart when the I-EDoS attack was launched (Fig. 18(b)). In such attack conditions, the red dots clearly denote the presence of a subset of instances whose productivity is significantly lower, thus being referred as lazy nodes. In consequence, the overall productivity is decremented since this scenario forces the workload distribution into a higher number of VNFC instances. To quantitively validate the distinction of productive and lazy instances, the results of the DBSCAN density-based clustering are presented for sampled measurements obtained at $t = 86$ under normal traffic conditions (Fig. 19(a)). Three groups of instances are created (Group A, Group B and lazy), with 80% of instances allocated to Groups A and B and remaining of 20% allocated to the group of possible lazy

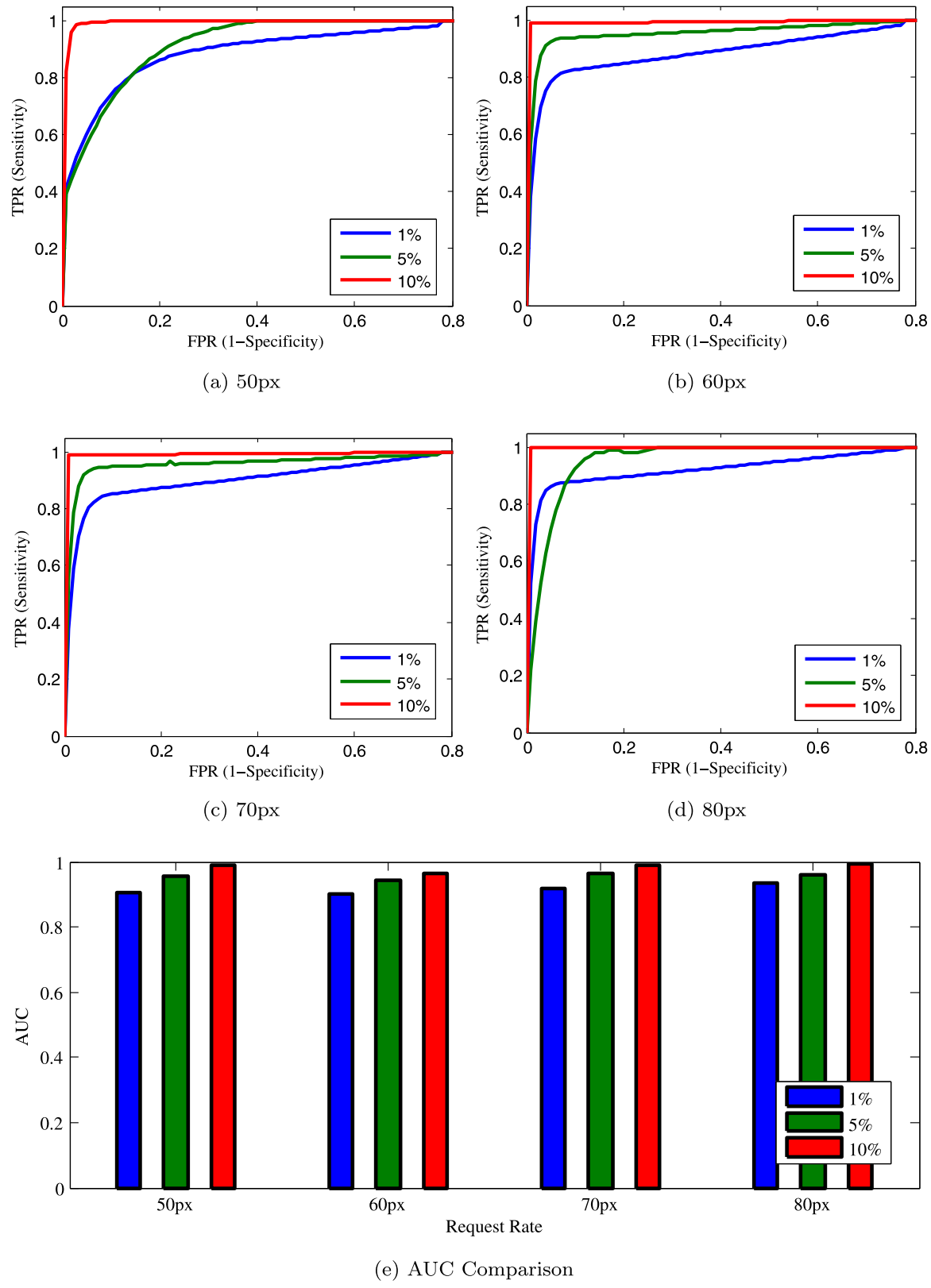
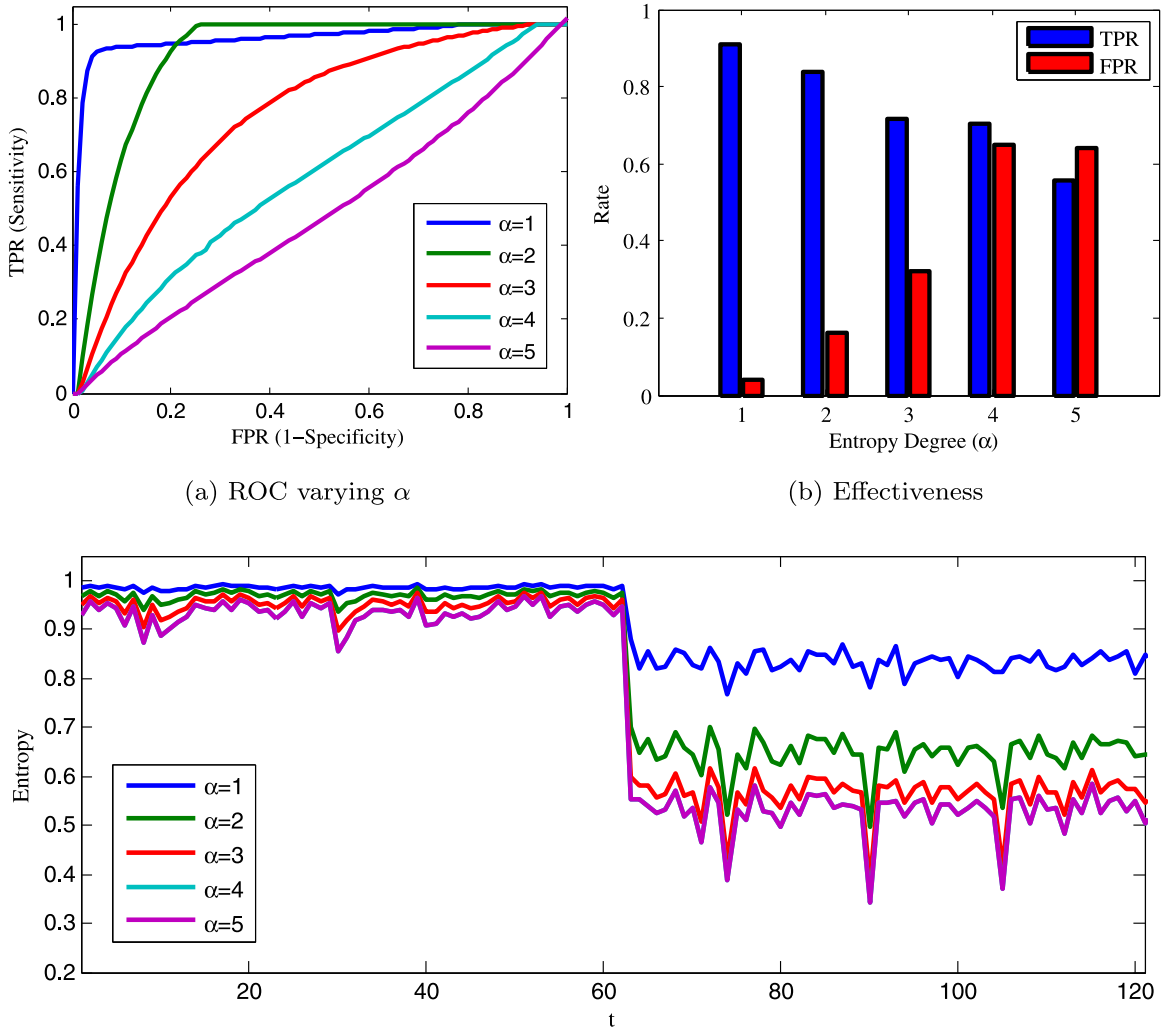


Fig. 13. Results when varying the attack rate and requests. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

instances. Likewise; taking the productivity measurements observed at $t = 19$ when the attack is performed (Fig. 19(b)), DBSCAN generates two groups. The lazy group gathers 73% of the allocated instances, while the latter poses the remaining 27% as productive instances.



(a) ROC varying α

(b) Effectiveness

(c) Variations on Entropy degree (α)

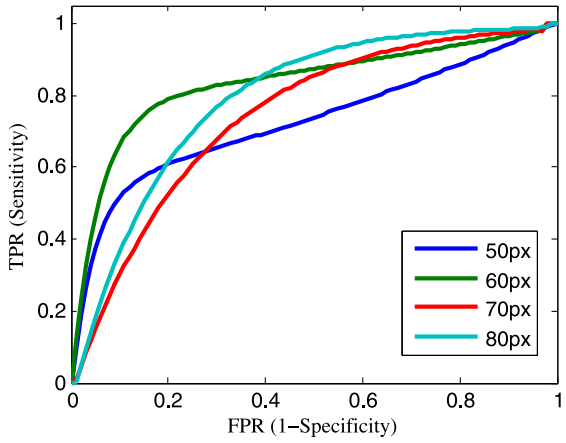
Fig. 14. Analysis of entropy degree variation.

7.2.2. Attack intensity

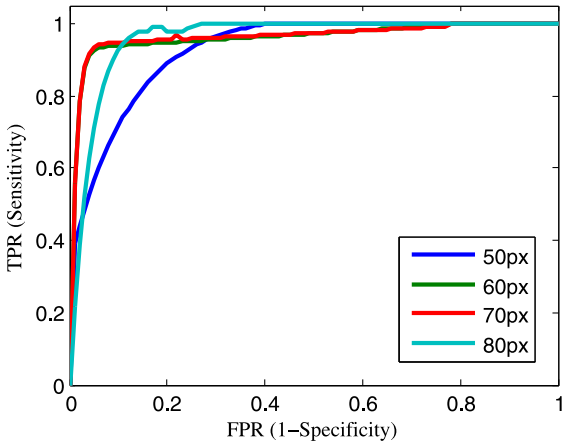
As would seem logical, the intensity of the I-EDoS attacks has directly influenced the detection capability of the approach. Fig. 20(a) and Table 10 display the effectiveness registered when analyzing threats that lead to increase 10%, 20%, 30% 40% and 50% the number of instances of the NFV considered at the previous example. In general terms, the success rate has varied little (see Fig. 20(c)); at the first four groups of attacks (10%, 20%, 30%, 40%) a distance of 0.022 (0.025%) was observed between the minimum hit rate (TPR=0.89, in 10%) and the best hit rate (TPR=0.91 in 40%). When the attacks posed higher intensity (50%) the hit rate slightly improved (TPR=0.94 in 50%). However, by taking into account the issuing of false positives the results were more significant (see Fig. 20(d)); in particular, if the attacks posed lower intensity, the detection method reached FPR=0.12; but by increasing their capacity to cause economical losses, the best observation decreased to FPR = 0.07, which represents an improvement of 58.3% over the worst registration. These situations are reflected in Fig. 20(a) and 20(b), where the AUC varies according to the attack intensity, being AUC=0.9483 in the worst case, and AUC=0.9811 in the best case. The effectiveness variations are mainly due to the clustering stage, where the instantiated NFVs are grouped based on productivity. The greater is the attack intensity, the greater is the number of instances created by the intruder that belong to the lazy group. This places a larger amount of normal instances in other groups, hence reducing the false positive rate in the best calibrations despite generally, it remains similar. In view of the results, it is possible to conclude that the proposed strategy is capable of detecting I-EDoS threats successfully. However, to maintain a setting that facilitates recognition of low intensity threats may result in the emission of a greater number of false positives, which should be considered by the security management strategy. It entails deciding the trade-off between protection and economic losses that better fit into the monitoring scenario and the adopted security policies.

Table 10
Summary of results considering different attack intensities.

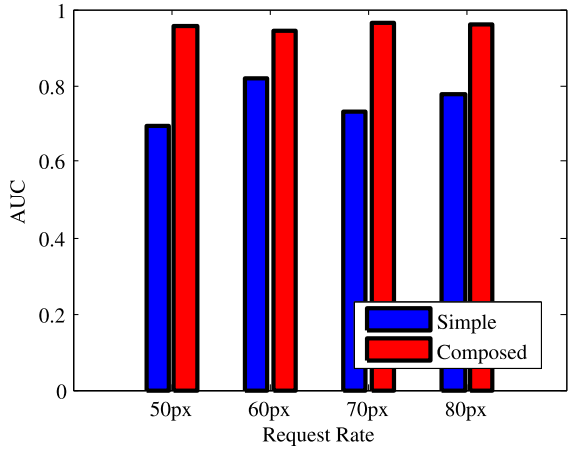
		Effectiveness measurements	
Intensity (%)	AUC	TPR	FPR
10	0.9483	0.8936	0.12
20	0.9498	0.9099	0.10
30	0.9654	0.9167	0.08
40	0.9708	0.9155	0.07
50	0.9811	0.9404	0.07



(a) Effectiveness when only analyzing X_{App}



(b) Effectiveness when analyzing X_{App} and X_{CPU}



(c) AUC Comparison

Fig. 15. Analysis when considering X_{App} and X_{CPU}.

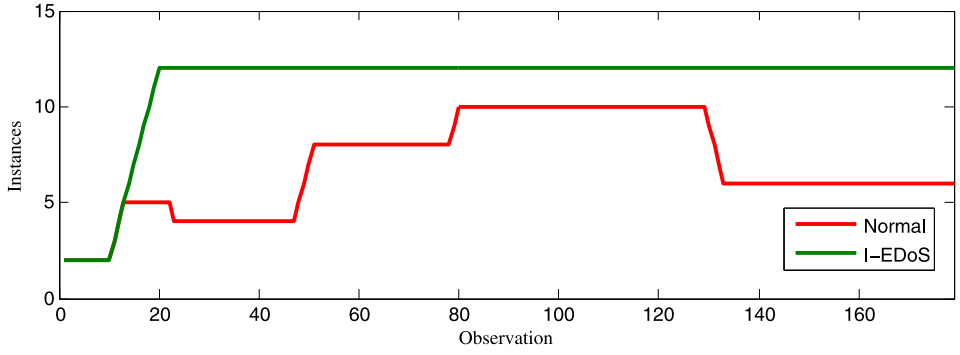
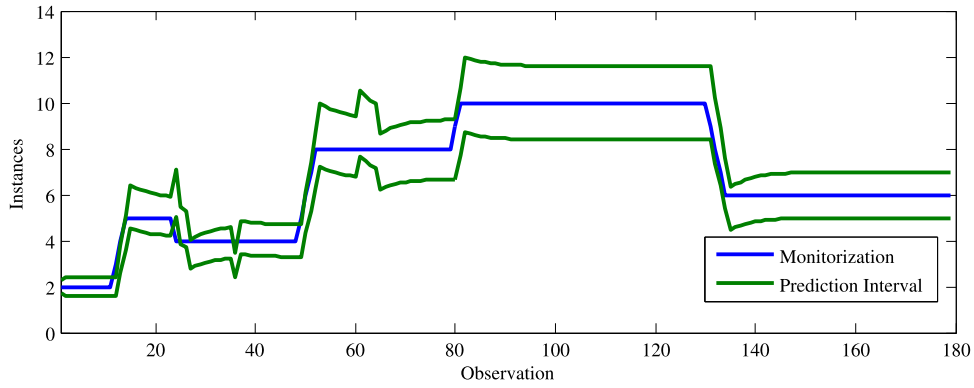


Fig. 16. Number of instances deployed in normal and attack scenarios.

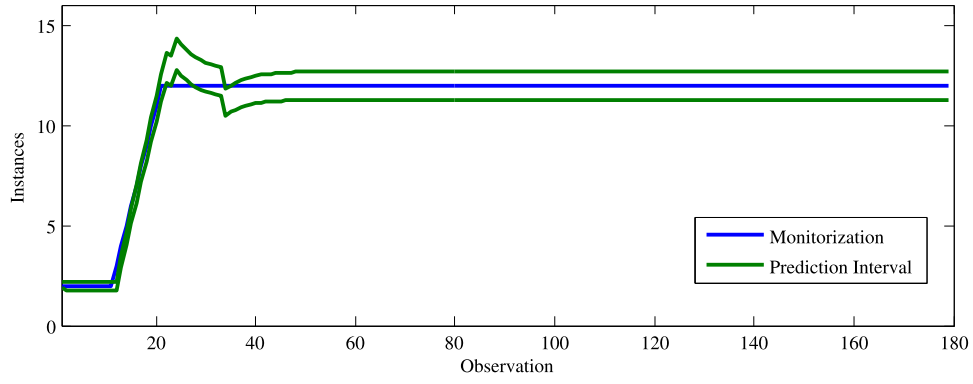
7.3. Discussion

Bearing the performed experimentation in mind, the following items are highlighted:

1. When analyzing W-EDoS, as the number of requests per node and the intensity of the attack increase, the accuracy of the system improves. The best results were observed when the sensors submit an average of at least 80 requests per sensor and the attack poses 10% intensity (AUC=0.995, TPR=1, FPR=0.01).
2. Attacks with 1% and 5% intensity, performed under the 80 px setting (Fig. 13d), expose a higher sensitivity to the K calibration, a fact explained by a less representative outperformance of the 1% intensity in a short segment of the ROC curve (i.e. small region where the blue line is above the green one). However,



(a) Normal traffic



(b) I-EDoS attack

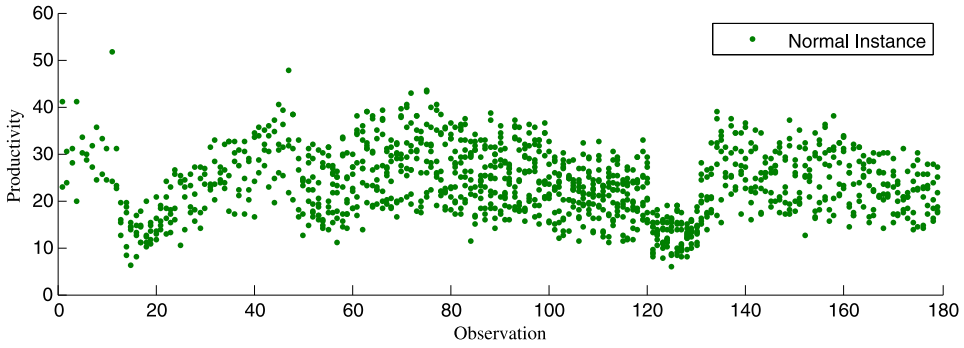
Fig. 17. Forecasting of the number of VNF instances.

a trade-off between a higher AUC of 0.9603 (5% intensity) and a lower FPR of 0.05 (reached at 1%) is inferred from Table 7, where it is also noticeable a seamlessly relationship at the highest intensity of 10%, concerning both the AUC and FPR

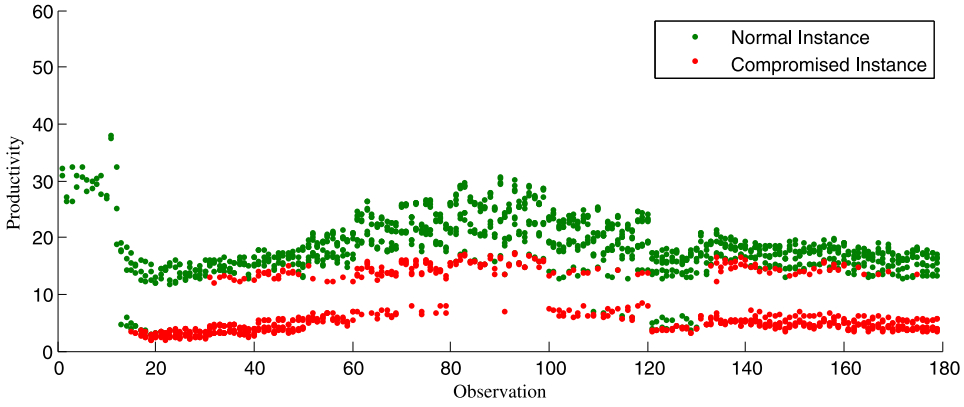
3. The W-EDoS detection approach is more effective when the Rényi entropy degree is low. The best observation was $\alpha = 1$ (AUC=0.9557, TPR=0.9113, FPR=0.0400) with a test setting that assumed an average rate of 60 requests per client/second, and attacks originated from 5% of the clients to be served
4. The W-EDoS detection when only considering X_{App} was not recommended because of the rebound effect of the forecasting and adaptive thresholding methods described in Section 5.2.1. It leads to a significant increase in the false positive rate.
5. The proposal for I-EDoS detection behaves better when analyzing attacks of greater intensity. The best results were observed when the attack intensity was 50% (AUC=0.9811, TPR=0.9404, FPR=0.07).
6. At I-EDoS detection, the greater is the attack intensity, the greater is the number of instances created by the intruded that belong to the lazy group. This has a direct impact on the false positive rate resultant of the optimal calibration. Hence if the attacks posed lower intensity, the detection method reached FPR=0.12; but by increasing their intensity, the best observation decreased to FPR = 0.07, which represents an improvement of 58.3% over the worst register

From these facts it is deduced that the main objective of the EDiSON proposal has been achieved, i.e. it was able to successfully recognize the different EDoS attacks against the self-organized network considered during the experimentation. However, the effectiveness has varied depending on different characteristics of the monitoring scenario, highlighting among them the heterogeneity of the information to be studied (1) and the intensity of the attack (1), (2), (4), (5). The first one allows more accurate modeling of the legitimate behavior by reducing the false positive rate, and the second facilitates the recognition of the intrusive activities. Finally, it is worth to highlighting the impact of the Rényi entropy degree at the W-EDoS detection approach (3), where this parameter significantly influenced the smoothing and noise reduction of the time series to be analyzed. This has resulted in the fact that $\alpha = 1$ (Shannon entropy), i.e. the most noise-tolerant setting, entailed greater accuracy, in a similar way to previous publications related with conventional DDoS recognition [68].

From an analytical point of view, a simple and modular solution has been proposed, following the extensibility by design principle, where the most complex tasks are executed in a dedicated processing layer, thus minimizing operational costs. Given that the definitions of W-EDoS and I-EDoS attacks lie on CRoWN indicators, the object of study has been clearly delimited. But certainly, the future will bring new ways of achieving EDoS, so it is not possible to guarantee the effectiveness of EDiSON in uncharted circumstances. It is also important to bear in mind that the proposed solution implements analytic tools well-known by the research community, for example the DES [82] forecast algorithm or the DBSCAN [89] clustering method. Their selection has been conveniently justified throughout the paper, but they pose advantages and disadvantages that should be considered prior to instantiation in alternative use cases. Note that if necessary, they can be replaced by similar algorithms. Finally, it should be highlighted that the proposal inherits the countermeasures related with the limitations

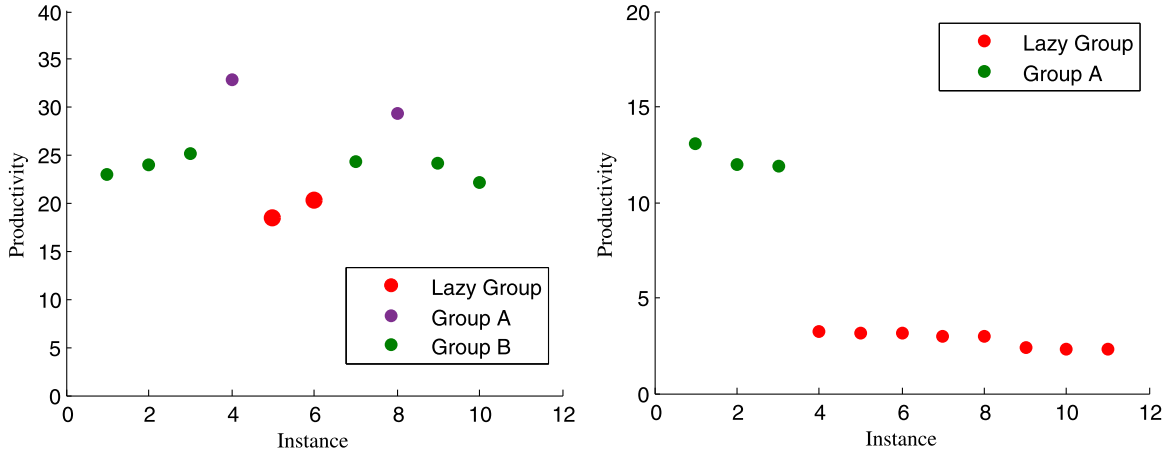


(a) Normal traffic



(b) I-EDoS attack

Fig. 18. Productivity distribution per VNF instance.



(a) Normal traffic classification at t=86 (b) Attack traffic classification at t=19

Fig. 19. DBSCAN classification by VNF productivity.

assumed at the design principles (see Section 5). Therefore, it does not incorporate adaptation techniques to non-stationary scenarios [73] nor robustness against adversarial attacks [68,69]. Neither an advanced information correlation strategy has been proposed, where data from different sensors could be pooled. These facilitate the inference of more precise diagnoses [71,72], which serve to risk assessment and to resolve the trade-off between the cost of countermeasures deployment and the estimated losses caused by the intrusion. Consequently, many interesting lines of future work have been raised.

8. Conclusions

The problem of Economical Denial of Sustainability in the Self-Organizing Network scene has been reviewed highlighting its adaptation to the two most prominent weak points. The first of them is the

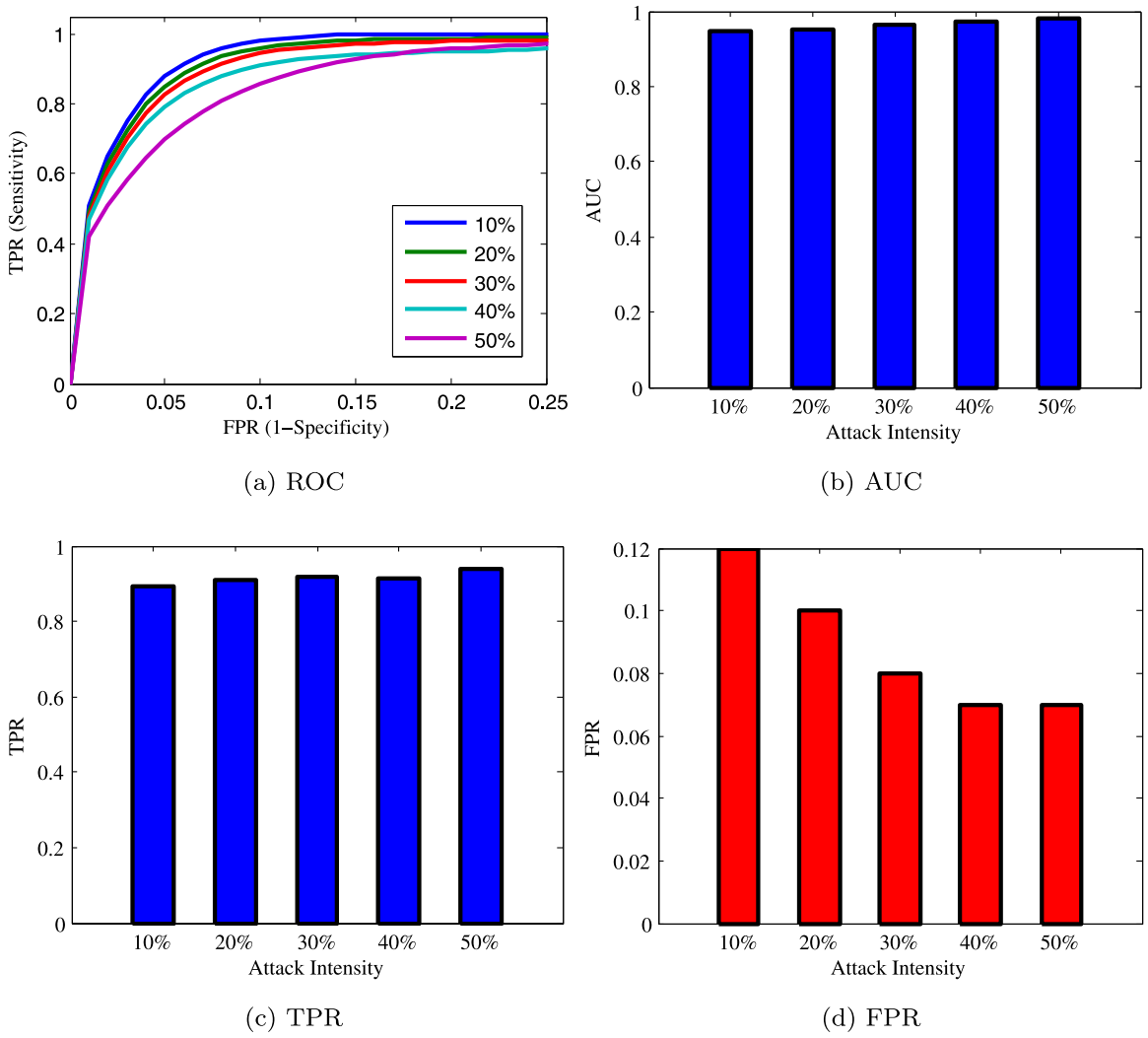


Fig. 20. Analysis of the effectiveness at different attack intensities.

W-EDoS threat, which aims on exploiting vulnerabilities by workload injection in NFV auto-scaling policies. On the other hand, the I-EDoS attacks take advantage of the NFV auto-instantiation capabilities by thwarting orchestration processes. Consequently, a pair of strategies for their mitigation have been proposed. They were based on modeling the normal behavior of the protected system and discovering discordant activities. The W-EDoS attack detection considers significant prediction errors in terms of CPU consumption (X_{app}) and response-time at application-level (X_{cpu}) of the instantiated VNFs. On the other hand, I-EDoS attack detection analyzes the relationship between the initialization of unproductive instances and the suspicious growth of the number of deployed VNF instances. Their effectiveness was proven at the performed preliminary experimentation, which considered different adjustment parameters, among others attack intensity, normal traffic features, confidence interval of adaptive thresholding and entropy degree. Therefore, it was possible to demonstrate that the proposal fulfills its main objective at the deployed testbed. However, in certain circumstances the proposal reported significant false positive rates, which rise interesting lines of future work. For example, integration of alternative analytic techniques, granularity optimization or adaptation to non-stationary processes. In addition, for the better understanding of our contributions several issues have not been addressed, which were outlined throughout the document. They included among others robustness against adversarial threats or the assumption of data protection policies, hence being relegated to forthcoming research.

Acknowledgments

This research was been framed into the SELFNET (Framework for Self-Organized Network Management in Virtualized and Software Defined Networks) project, funded by the European Commission Horizon 2020 Programme under Grant Agreement number H2020-ICT-2014-2/671672. The authors particularly acknowledge L.J.G. Villalba for his guidance and support on coordinating tasks and logistics.

References

[1] NGMN Alliance, 5G White paper, 2015. https://www.ngmn.org/5g-white-paper/5g-white-paper.html. (Accessed on 16.08.2017).
[2] NetWorld2020 ETP: European Technology Platform for Communications Networks and Services, 5G: Challenges, Research Priorities, and Recommendations, 2014, https://www.networld2020.eu/wp-content/uploads/2015/01/Joint-Whitepaper-V12-clean-after-consultation.pdf. (Accessed on 31.10.2018).
[3] L. Gavrilovska, V. Rakovic, V. Atanasovski, Visions towards 5G: Technical requirements and potential enablers, wireless personal communications, Wirel. Pers. Commun. 87 (3) (2015) 731–757.
[4] L. Barona Lopez, A. Valdivieso Caraguay, M. Sotelo Monge, L. Garcia Villalba, Key technologies in the context of future networks: Operational and management requirements, Future Internet 9 (1) (2016) 731–757.
[5] SELFNET, Self-Organized Network Management in Virtualized and Software Defined Networks, 2015, http://www.selfnet-5g.eu. (Accessed on 27.11.2017).

[6] A. Atayero, O. Adu, A. Alatishe, Self organizing networks for 3GPP LTE, in: Proceedings of the 14th International Conference on Computational Science and Its Applications, ICCSA, Guimaraes, Portugal, 2014, pp. 242–254.
[7] 3GPP TS 32.500, Self-Organising Networks (SON): Concepts and requirements, 2015. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2031. (Accessed on 19.02.2018).
[8] J.O. Kephart, D.M. Chess, The vision of autonomic computing, Computer (1) (2003) 41–50.
[9] M. Mao, J. Li, M. Humphrey, Cloud auto-scaling with deadline and budget constraints, in: Proceedings of the 11th IEEE/ACM International Conference on Grid Computing, GRID, Brussels, Belgium, 2010, pp. 41–48.
[10] G.A.W. Group, View on 5G Architecture, 2016, http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/010201_60/gs_NFV002v010201p.pdf. (Accessed on 05.10.2017).
[11] P. Bawa, S. Manickam, Critical review of economical denial of sustainability (EDoS) mitigation techniques, J. Comput. Sci. 11 (7) (2015) 855–862.
[12] ETSI, GS NFV-MAN 001 V1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration, 2014, https://www.etsi.org/deliver/etsi_gs/nfv-man/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf. (Accessed on 05.01.2019).
[13] J. Maestre Vidal, M.A. Sotelo Monge, L.J. García Villalba, Detecting Workload-based and Instantiation-based Economic Denial of Sustainability on 5G environments, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES, Hamburg, Germany, 2018, pp. 50:1–50:8.
[14] N. Samaan, A. Karmouch, Towards autonomic network management: an analysis of current and future research directions, IEEE Commun. Surv. Tutor. 11 (3) (2009).
[15] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, F. Zambonelli, A survey of autonomic communications, ACM Trans. Auton. Adapt. Syst. 1 (2) (2006) 223–259.
[16] M. Mamei, R. Menezes, R. Tolksdorf, F. Zambonelli, Case studies for self-organization in computer science, J. Syst. Archit. 52 (8) (2006) 443–460.
[17] G.W.G. QoS, Cognitive Network Management for 5G on Network Management, 2016, https://5g-ppp.eu/wp-content/uploads/2016/11/NetworkManagement_WhitePaper_1.0.pdf.
[18] S. Feng, E. Seidel, Self-Organizing Networks (SON) in 3GPP Long Term Evolution), Novel Mobile Radio Research, 2008, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2031.
[19] H. Rahman, T. Kanter, R. Rahmani, Supporting self-organization with logical-clustering towards autonomic management of internet-of-things, Int. J. Adv. Comput. Sci. Appl. 6 (2) (2015) 24–33.
[20] M. Peng, D. Liang, Y. Wei, J. Li, C. H.H., Self-configuration and self-optimization in LTE-advanced heterogeneous networks, IEEE Commun. Mag. 51 (5) (2013) 36–45.
[21] O. Aliu, A. Imran, M. Imran, B. Evans, A survey of self organisation in future cellular networks, IEEE Commun. Surv. Tutor. 51 (1) (2013) 336–361.
[22] A.H. Celdrán, M.G. Pérez, F.J.G. Clemente, G.M. Pérez, Towards the autonomous provision of self-protection capabilities in 5G networks, J. Ambient Intell. Humaniz. Comput. (2018) 1–14.
[23] S. Hämäläinen, H. Sanneck, C. Sartori, LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency, John Wiley and Sons, New York, 2012.
[24] B. Jennings, S. Van Der Meer, S. Balasubramaniam, D. Botvich, M. Foghlu, W. Donnelly, J. Strassner, Towards autonomic management of communications networks, IEEE Commun. Mag. 45 (10) (2007) 112–121.
[25] C. Fortuna, M. Mohorcic, Trends in the development of communication networks: Cognitive networks, Comput. Netw. 53 (9) (2009) 1354–1376.
[26] C. Hoff, Cloud Computing Security: From DDoS (Distributed Denial Of Service) to EDoS (Economic Denial of Sustainability), 2008, http://rationalsecurity.typepad.com/blog/2008/11/cloud-computing-security-from-ddos-distributed-denial-of-service-to-edos-economic-denial-of-sustaina.html. (Accessed on 18.09.2017).
[27] C. Hoff, A Couple of Follow-Ups On The EDoS (Economic Denial Of Sustainability) Concept…, 2009, https://rationalsecurity.typepad.com/blog/edos. (Accessed on 19.09.2017).
[28] R. Cohen, Cloud Attack: Economic Denial of Sustainability (EDoS), 2009, http://www.elasticvapor.com/2009/01/cloud-attack-economic-denial-of.html. (Accessed on 13.09.2017).
[29] A. Bremler-Barr, E. Brosh, M. Sides, DDoS attack on cloud auto-scaling mechanisms, in: Proceedings of the 2017 IEEE Conference on Computer Communications , INFOCOM 2017, Atlanta, GA, US, 2017, pp. 1–9.
[30] G. Sonami, M. Gaur, D. Sanhgi, M. Conti, DDoS Attacks in cloud computing: Collateral damage to non-targets, Comput. Netw. 109 (2016) 157–171.
[31] P. Singh, S. Manickam, S. Rehman, A survey of mitigation techniques against Economic Denial of Sustainability (EDoS) ttack on cloud computing architecture, in: Proceedings of the 3rd International Conference on Reliability, Infocom Technologies and Optimization, ICRITO, Noida, India, 2014.
[32] G. Sonami, M. Gaur, D. Sanhgi, M. Conti, R. Buyya, DDoS Attacks in cloud computing: Issues, taxonomy, and future directions, Comput. Commun. 107 (2017) 30–48.
[33] S. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surv. Tutor. 15 (4) (2013) 2046–2069.
[34] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, F. Tangm, Discriminating DDoS attacks from flash crowds using flow correlation coefficient, IEEE Trans. Parallel Distrib. Syst. 23 (6) (2012) 1073–1080.
[35] A. Bhingarkar, B. Shah, A survey: Securing cloud infrastructure against edos attack, in: Proceedings of the 2015 of the International Conference on Grid Computing and Applications, GCA, Athens, Greece, 2015, pp. 16–22.
[36] S. Vivinsandar, S. Shenai, Economic denial of sustainability (EDoS) in cloud services using HTTP and XML based DDoS attacks, Int. J. Comput. Appl. 41 (20) (2012) 11–16.
[37] W. Zhou, W. Jia, S. Wen, Y. Xiang, W. Zhou, Detection and defense of application-layer DDoS attacks in backbone web traffic, Future Gener. Comput. Syst. 38 (2014) 36–46.
[38] K. Singh, P. Singh, K. Kumar, Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges, Comput. Secur. 65 (2017) 344–372.
[39] K. Singh, K. Chatterjee, Cloud security numbers and challenges: A survey, J. Netw. Comput. Appl. 79 (2017) 88–115.
[40] Z. Baig, S. Sait, F. Binbeshr, Controlled access to cloud resources for mitigating economic denial of sustainability (EDoS) attacks, Comput. Netw. 97 (2016) 31–47.
[41] F. Al-Haidari, M. Sqalli, K. Salah, Enhanced EDoS-shield for mitigating EDoS attacks originating from spoofed IP addresses, in: Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom, Liverpool, UK, 2012, pp. 16–22.
[42] J. Idziorek, M. Tannian, Exploiting cloud utility models for profit and ruin, in: Proceedings of the 2011 IEEE International Conference on Cloud Computing, CLOUD, Washington, DC, US, 2011, pp. 16–22.
[43] S. Bhingarkar, D. Shah, Fuzzy entropy based feature selection for website user classification in EDoS defense, in: International Conference on Next Generation Computing Technologies, Springer, Singapore, Singapore, 2017, pp. 440–449.
[44] R. Ravichandiran, H. Bannazadeh, A. Leon-Garcia, Anomaly detection using resource behaviour analysis for autoscaling systems, in: Proceedings of the 4th IEEE Conference on Network Softwarization and Workshops, NetSoft, IEEE, Montreal, QC, Canada, 2018, pp. 192–196.
[45] K. Singh, K. Thongam, T. De, Entropy-based application layer DDoS attack detection using artificial neural networks, Entropy 18 (10) (2016).
[46] A. Koduru, T. Neelakantam, S. Bhanu, Detection of economic denial of sustainability using time spent on a web page in cloud, in: Proceedings of the 2013 IEEE International Conference on Cloud Computing in Emerging Markets, CCEM, Bangalore, India, 2013, pp. 16–22.
[47] M. Kumar, P. Sujatha, V. Kalva, R. Nagori, A. Katukojwala, M. Kumar, Mitigating economic denial of sustainability (EDoS) in cloud computing using in-cloud scrubber service, in: Proceedings of the 4th International Conference on Computational Intelligence and Communication Networks, CICN, Mathura, India, 2012.
[48] W. Alosaimi, K. Al-Begain, A new method to mitigate the impacts of the economical denial of sustainability attacks against the cloud, in: Proceedings of the 14th Annual Graduates Symposium on the Convergence of Telecommunication, Networking and Broadcasting, PGNet, Liverpool, UK, 2013.
[49] M. Masood, Z. Anwar, S. Raza, M. Hur, EDoS Armor: A cost effective economic denial of sustainability attack mitigation framework for e-commerce applications in cloud environments, in: Proceedings of the 16th International Multi Topic Conference (INMIC), Lahore, Pakistan, 2013.
[50] H. Khor, A. Nakao, sPoW: On-demand cloud-based eDDoS mitigation mechanism, in: Proceedings of the IEEE/IFIP International Conference on Dependable Systems & Networks (DSN), Lisbon, Portugal, 2009.
[51] A. Shawahna, M. Abu-Amara, A. Mahmoud, Y.E. Osais, EDoS-ADS: An enhanced mitigation technique against economic denial of sustainability (EDoS) attacks, IEEE Trans. Cloud Comput. (2018) http://dx.doi.org/10.1109/TCC.2018.2805907.
[52] R. Ravichandiran, H. Bannazadeh, A. Leon-Garcia, Edos mitigation for autonomic management on multi-tier IoT, in: Proceedings of the 14th International Conference on Network and Service Management, CNSM, IEEE, Rome, Italy, 2018, pp. 285–289.
[53] K. Bhushan, et al., DDoS attack mitigation and resource provisioning in cloud using fog computing, in: Proceedings of the International Conference on Smart Technologies for Smart Nation, SmartTechCon, IEEE, Bangalore, India, 2017, pp. 308–313.
[54] S. Yu, Y. Tian, S. Guo, D. Wu, Can we beat DDoS attacks in clouds? IEEE Trans. Parallel Distrib. Syst. 25 (9) (2014) 2245–2254.
[55] K. Xue, W. Chen, W. Li, J. Hong, P. Hong, Combining data owner-side and cloud-side access control for encrypted cloud storage, IEEE Trans. Inf. Forensics Secur. 13 (8) (2018) 2062–2074.
[56] J. Idziorek, M. Tannian, D. Jacobson, Attribution of fraudulent resource consumption in the cloud, in: Proceedings of the 5th IEEE International Conference on Cloud Computing, Honolulu, HI, US, 2012.
[57] N. Alenezi, M. Reed, Uniform DoS traceback, Comput. Secur. 45 (1) (2014) 17–26.

[58] K. Wang, M. Du, S. Maharjan, Y. Sun, Strategic honeypot game model for distributed denial of service attacks in the smart grid, IEEE Trans. Smart Grid 8 (5) (2017) 2474–2482.
[59] G. Yao, J. Bi, A. Vasilakos, Passive IP traceback: Disclosing the locations of IP spoofers from path backscatter, IEEE Trans. Inf. Forensics Secur. 10 (3) (2015) 471–484.
[60] E. Jeong, B. Lee, An IP traceback protocol using a compressed hash table, a sinkhole router and data mining based on network forensics against network attacks, Future Gener. Comput. Syst. 33 (1) (2014) 42–52.
[61] M. Karami, S. Chen, Attribution of economic denial of sustainability attacks in public clouds, in: International Conference on Security and Privacy in Communication Systems, Springer, Cham, Germany, 2017, pp. 373–391.
[62] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009).
[63] S. Bhatia, D. Schmidt, G. Mohay, A framework for generating realistic traffic for distributed denial-of-service attacks and flash events, Comput. Secur. 40 (1) (2014) 95–107.
[64] P. Zhang, H. Wang, C. Hu, C. Lin, On denial of service attacks in software defined networks, IEEE Netw. 30 (6) (2016) 28–33.
[65] Y. Afek, A. Bremler-Barr, Y. Harchol, D. Hay, Y. Korai, Making DPI engines resilient to algorithmic complexity attacks, IEEE/ACM Trans. Netw. 24 (6) (2016) 3262–3275.
[66] ETSI, GS NFV 003 V1.2.1: Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV, 2014, https://www.etsi.org/deliver/etsi_gs/nfv/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf. (Accessed on 11.01.2019).
[67] S. Sarmadi, M. Li, S. Chellappan, A statistical framework to forecast duration and volume of internet usage based on pervasive monitoring of netflow logs, in: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications, AINA, IEEE, 2018, pp. 480–487.
[68] I. Ozcelik, R. Brooks, Deceiving entropy based dos detection, Comput. Secur. 48 (1) (2015) 234–245.
[69] B. Al-Duwairi, A. Al-Hammouri, Fast flux watch: A mechanism for online detection of fast flux networks, J. Adv. Res. 5 (4) (2014) 473–479.
[70] M. Bhuyan, D. Bhattacharyya, J. Kalita, An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection, Pattern Recognit. Lett. 51 (1) (2015) 1–7.
[71] S. Salah, G. Macia-Fernandez, J. Diaz-Verdejo, A model-based survey of alert correlation techniques, Comput. Netw. 57 (5) (2013) 1289–1317.
[72] S. Mirheidari, S. Arshad, R. Jalili, Alert correlation algorithms: A survey and taxonomy, in: Proceedings of the 5th International Symposium on Cyberspace Safety and Security, CSS, Zhangjiajie, China, number 8300, 2013, pp. 183–197.
[73] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, IEEE Comput. Intell. Mag. 10 (4) (2015) 12–25.
[74] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, P. Owezarski, Modeling internet backbone traffic at the flow level, IEEE Trans. Signal Process. 51 (8) (2003) 2111–2124.
[75] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011).
[76] ETSI, GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework, 2014, https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.02.01_60/gs_nfv002v010201p.pdf. (Accessed on 05.10.2017).
[77] Openstack: Open source sotware for creating private and public clouds. https://www.openstack.org. (Accessed on 31.10.2017).
[78] Amazon Web Services. https://aws.amazon.com. (Accessed on 05.01.2018).
[79] Microsoft Azure. https://azure.microsoft.com. (Accessed on 05.12.2017).
[80] Telemetry service overview. https://docs.openstack.org/mitaka/install-guide-ubuntu/common/get_started_telemetry.html. (Accessed on 13.01.2018).
[81] S. Makridakis, M. Hibon, The M3-Competition: results, conclusions and implications, Int. J. Forecast. 16 (4) (2000) 451–476.
[82] E. Gardner, D. Dannenbring, Forecasting with exponential smoothing: Some guidelines for model selection, Decis. Sci. 11 (2) (1980) 370–383.
[83] G. Groff, Empirical comparison of models for short-range forecasting, Manage. Sci. 20 (1) (1973) 22–31.
[84] S. Hillmer, G. Tiao, An ARIMA-model-based approach to seasonal adjustment, J. Amer. Statist. Assoc. 77 (377) (1980) 63–70.
[85] R. Brown, Exponential smoothing for predicting demand, Oper. Res. 5 (1) (1957) 145.
[86] C. Holt, Forecasting seasonals and trends by exponentially weighted moving averages, Int. J. Forecast. 20 (1) (2004) 5–10.
[87] S. Makridakis, S. Wheelwright, R. Hyndman, Forecasting: Methods and Applications, John Wiley and Sons, New York, 1998.
[88] R. Hyndman, A. Koehler, J. Ord, R. Snyder, Prediction intervals for exponential smoothing state space models, J. Forecast. 24 (2005) 17–37.
[89] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, KDD, Portland, OR, US, 1996, pp. 226–231.
[90] J. Cao, D. Spooner, S. Jarvis, G. Nudd, Grid load balancing using intelligent agents, Future Gener. Comput. Syst. 21 (1) (2005) 135–149.
[91] RabbitMQ: Messaging that just works. https://www.rabbitmq.com. (Accessed on 05.12.2017).
[92] Openstack Senlin: Clustering service for OpenStack clouds. https://wiki.openstack.org/wiki/Senlin. (Accessed on 11.11.2017).
[93] Ceilometer measurements. https://docs.openstack.org/ceilometer/pike/admin/telemetry-measurements.html. (Accessed on 07.02.2018).
[94] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. Witten, The WEKA data mining software: an update, ACM SIGKDD Explor. 11 (1) (2009) 10–18.
[95] Drools: Business Rules Management System (BRMS). https://www.drools.org. (Accessed on 30.12.2017).
[96] Flask—A Python microframework. http://flask.pocoo.org. (Accessed on 08.11.2017).
[97] The Httperf HTTP load generator. https://github.com/httperf/httperf. (Accessed on 15.10.2017).
[98] CVE-2016-9877. https://www.cvedetails.com/cve/CVE-2016-9877/. (Accessed on 16.01.2018).