ANSwer: Combining NFV and SDN Features for Network Resilience Strategies

Cristian Cleder Machado, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho
Computer Networks Group – Institute of Informatics – Federal University of Rio Grande do Sul
Porto Alegre, Brazil
Email: {ccmachado, granville, alberto}@inf.ufrgs.br

Abstract— **Software-Defined Networking (SDN) relies on open programmability of network devices, which is achieved by defining new communication interfaces, network operating systems, and changing the traditional decision-making logic of regular TCP/IP networks. Network Functions Virtualization (NFV), in turn, permits virtualizing network functions that are traditionally performed by physical middleboxes (*e.g.*, firewalling and intrusion detection/prevention). Although SDN and NFV improve the flexibility of the management of computer networks, SDN remains vulnerable to major network security problems, such as Distributed Denial of Service (DDoS) attacks. These attacks typically result in the disruption of network services and resources. In this paper, we introduce ANSwer, an architecture that combines NFV and SDN features to create sophisticated network resilience strategies. ANSwer relies on a feedback control-loop which explores SDN features to monitor and analyze the behavior of the network infrastructure, indicating whether parts of an existing resilience strategy can be reconfigured to achieve more satisfactory results, or if an entire resilience strategy needs to be added or replaced. Our experiments demonstrate that ANSwer can rapidly identify and handle distinct anomalies in different scenarios, indicating that the reconfiguration and deployment of resilience strategies can be performed in real-time.**

Keywords*—*Network Resilience Strategies; Network Functions Virtualization; Software-Defined Networking;

I. Introduction

Software-Defined Networking (SDN) together with OpenFlow provide a reformulation of the network operating systems, communication interfaces, and the decision-making logic that are typically proprietary and specific to each vendor in TCP/IP networks. SDN presents a clear separation between application, forwarding/data, control, and management planes. Such separation allows moving part of the decision-making logic from network devices to external elements often referred to as controllers, thus turning network devices into simpler packet forwarding elements. Network controllers have an overall view of the network infrastructure, and switching elements can be configured through the OpenFlow protocol [1]. Finally, SDN relies on the open programmability of network devices, aiming to customize network infrastructures, according to the needs of network operators [2]. As a result, in an OpenFlow-based SDN architecture, it is easier to collect and analyze network information, facilitating the deployment of mechanisms to maintain network resilience [3].

In addition to the benefits mentioned above, SDN also offers a suitable environment for the deployment of Network Functions Virtualization (NFV) [4]. NFV enables virtualizing network functions, *e.g.*, firewalls and Intrusion Detection/Prevention Systems (IDSes/IPSes), that are traditionally performed by proprietary hardware, into so called Virtualized Network Functions (VNFs). NFV can enhance SDN's resilience in several aspects. First, VNFs can be instantiated (and terminated) on-demand to solve specific network anomalies. Second, a VNF may be replaced by other VNF when anomalies have not been resolved properly, for example, an IDS/IPS can be replaced by a firewall if the former is not being effective against a port scan attack. Third, VNFs with the same function can be replicated aiming at load balancing and/or survivability. Finally, a set of VNFs can be configured in advance, stored in a repository, and automatically selected and deployed based on policies or by analyzing network traffic behavior [5].

Despite the aforementioned SDN features and NFV approaches that have been used to improve network resilience, SDN remains vulnerable to a number of network security problems, such as Distributed Denial of Service (DDoS) attacks. DDoS attacks can lead to overload of network devices through an overwhelming number of illegitimate requests, resulting in network resilience issues, such as limitation or disruption of network services and resources. In addition to these problems, in SDN environments the standardization of interfaces, communication protocols, and network operating systems can make a successful attack easier to be performed, because the attacker has no need to analyze specific protocols for each individual network device. We advocate that these problems remain unresolved because the combination of SDN features (like controller devices with the ability to have an overall view of the network infrastructure) with NFV aspects (like replacement of VNFs on-demand) to improve network resilience mechanisms has not been extensively investigated.

We introduce in this paper ANSwer (***A****rchitecture for combined* ***N****etwork functions virtualization and* ***S****oft****W****ar****E****-defined netwo****R****king features*), an architecture that combines VNFs with SDN features to create a set of strategies for network resilience. The key to our architecture is a feedback control-loop that continuously monitors and analyzes the behavior of the network infrastructure and its services. This control-loop allows continuously capturing feedback from the network infrastructure after the deployment of different resilience strategies. On the one hand, each VNF comprises a set of dynamic remedial actions, such as forwarding traffic to a specific VNF. On the other hand, SDN features are used to monitor and warn the occurrence of anomalies, *e.g.*, network traffic overload, service disruption. In addition, we also use SDN to employ remedial actions, *e.g.*, network/host isolation, when VNFs alone cannot cope with the anomalies.

The remainder of this paper is organized as follows. In Section II we present our architecture for network resilience. In Section III we present the prototype, experiments, and the results achieved. In Section IV we discuss the related work. Finally, in Section V we conclude the paper with final remarks and proposals for future work.

II. Architecture for Combined NFV and SDN Management

In this section, we introduce ANSwer, an architecture that comprises a set of Virtualized Network Functions (VNFs) combined with SDN features for maintaining resilience in SDN environments.

A. Combined SDN/NFV Resilience Strategies

The proposed solution aims to resolve (or alleviate) network resilience problems. When anomalies are identified (*e.g.*, network traffic overload) a *resilience strategy* is triggered. We define a *resilience strategy* as a set of resilience-related functions or mechanisms performed by network devices. For example, a firewall can perform one or more *resilience strategies*, such as (*i*) dropping unauthorized access coming from the Internet or Local Area Network (LAN); (*ii*) forwarding traffic to be analyzed by other components; and (*iii*) masking destination IP addresses to protect legitimate services from attacks. To build each *resilience strategy*, we systematically analyzed the set of features in the universe of SDN and NFV, and how they could be combined. It is important to emphasize that the set of features in the universe of SDN and NFV is very large, and certainly was not fully explored in this work. However, to the best of our knowledge, we have identified several combinations that somehow may be explored. For example, on the one hand, SDN presents the controller devices with the ability to have an overall view of the network infrastructure. On the other hand, in NFV, VMs provide independent and isolated processing environments and can be instantiated and/or removed on-demand. When they are combined, the network overall view can improve the process of instantiating VMs, thus avoiding possible bottlenecks, or places with low throughput. Table I shows some of the possible combinations.

After the analysis of the possible combinations of NFV and SDN features, we defined a set of *resilience strategies*. In ANSwer, these *resilience strategies* were classified into four main types according to their purposes. This classification is based on the concepts and discussions presented in the works of Chandola *et al.* [7], Patcha and Park [8], Singh *et al.* [9] and it is extended in this work. According to the authors, with these four types, an infrastructure can achieve satisfactory levels of resilience against different issues, for example, survivability of controllers and servers; traffic tolerance and dependability of switches, controllers, and services, among others. The four types are: (*i*) *Trigger*, which causes the deployment of strategies based on the behavior of other strategies; (*ii*) *Analyze*, which investigates an anomaly to generate log files and messages; (*iii*) *Deceive*, which simulates services and functions; and (*iv*) *Drop*, which blocks a possible anomaly based on network traffic information. Further, *resilience strategies* can be designed in two ways: (*i*) by using SDN features (such as the controller ability to have an overall view of the network infrastructure); or (*ii*) by using NFV aspects (such as replacement of VNFs on-demand). *Resilience strategies* can be used to deal with one or more anomalies. Notwithstanding, one or more *resilience strategies* may be needed to resolve a specific anomaly. Table II illustrates the main *resilience strategies* that we anticipate.

B. Resilience Configuration

The key aspect of our approach is a *feedback control-loop*. Its main goal is to identify the optimal strategy to resolve a specific type of network anomaly. To achieve this, the feedback control-loop tries to learn the optimal strategy from its history of effective interaction with the environment. To determine the optimal strategy, the *feedback control-loop* continuously receives information about all known network infrastructure elements (*e.g.*, switches, routers, hosts, and links) to analyze and identify network and service conditions (*e.g.*, amount of packets and bytes traversed in each switch, state of network elements, available bandwidth and link delay). This aims to provide an early warning about possible anomalies (*e.g.*, an increased throughput in a switch device) and trigger *resilience strategies*. Network conditions are obtained and measured in two ways: (*i*) by sending Internet Control Message Protocol (ICMP) packets; (*ii*) and by collecting traffic flow information from each switch. On the one hand, ICMP packets are used to identify available bandwidth and link delay. On the other hand, for each traffic flow, we analyze the mean and standard deviation of the number of packets and bytes traversed in each switch in order to create a traffic profile, *i.e.*, to determine the behavior of each traffic type. The mean and standard deviation are created according to Silva *et al.* [6]. As a result, the *control-loop* is configured with a set of parameters that (*i*) indicate that network traffic is normal if it is within a particular standard deviation, *e.g.*, lower than 10% of the average number of packets and bytes; and (*ii*) indicate that network traffic is abnormal if this outlier appears on numerous occasions, *e.g.*, more than 5 times continuously.

The *control-loop* can be scheduled to run at each time step t. When ANSwer identifies a change in the network behavior, the time step t is decreased by half, and it starts monitoring parts of the network with suspected anomalies. If the anomaly remains during the next time step, the first *resilience strategy* is activated. We have established an initial sequence and also use the set of possibilities to combine and deploy the resilience strategies based on Table II. The initial sequence is #1, #10, #11, #4, #6, and #15. Based on the discussions presented in the works of Chandola *et al.* [7], Patcha and Park [8], Singh *et al.* [9] this sequence may decrease the utilization of computer resources, such as memory and processing in the network devices, and bandwidth or link delay. After triggering and deploying a *resilience strategy*, the *control-loop* can analyze and determine if this strategy is being effective or not. Depending on the result of this analysis, *resilience strategies* can be removed, added or replaced until finding the optimal way to resolve a specific anomaly.

C. ANSwer Operational Phases

To ensure network resilience using combined SDN and NFV features, we propose a three-phase process: (*i*) *Initialization Phase* executes a network discovery process to identify network elements (such as switches, hosts, and links), by collecting Link Layer Discovery Protocol (LLDP) packets. This process is able to discover the amount of network elements and their links in order to establish better routes to network services and functions, such as Web, FTP, Firewall, IDS/IPS; (*ii*) *Detection Phase* performs the flow analysis process to analyze the behavior of the network infrastructure and its services in order to convey information for the feedback control-loop; and (*iii*) *Mitigation Phase* executes the strategy deployment process to trigger *resilience strategies* (shown in Table II) based on the *Feedback control-loop*.

III. Prototype and Experimental Evaluation

In this section we describe our prototype overview and discuss the experimental evaluation and initial results achieved.

TABLE I: Possible combinations between SDN and NFV features.

SID	SDN Features	Combinable NFV Features	NID	NFV Features	Combinable SDN Features
1	Centralized Decision-making Logic	all	A	Virtualized Network Appliances	#1, #3, #4, #6, #7, #8, #9, #10, #11, and #12
2	Application Plane	#A, #B, #C, #D, #F, #G, #H, #I, and #J	B	Virtualized Network Functions	#1, #2, #3, #4, #6, #7, #8, #9, #10, and #11
3	Control Plane	all	C	Virtualized Network Services	#1, #2, #3, #4, #6, #7, #8, #9, #10, and #11
4	Forwarding/Data Plane	all	D	Virtualized Network Applications	#1, #3, #4, #6, #7, #8, #9, #10, #11, and #12
5	Management Plane	#A, #B, #C, #D, #E, #F, #G, and #H	E	Management and Orchestration	all
6	Open Programmability	all	F	Forwarding Graphs	#1, #3, #4, #5, #6, #7, #8, #9, #10, and #11
7	Traffic Isolations	all	G	Chaining/Steering	#1, #3, #4, #5, #6, #7, #8, #9, #10, and #11
8	Network Slices	#A, #B, #C, #F, #G, #H, #I, and #J	H	Scaling Up/Down	all
9	Network Overall View	all	I	Easily Migrated, Relocated, and/or Duplicated VMs	all
10	Easily Collect Network Information	all	J	Instantiation/Extinction of VMs On-demand	all
11	Standardization (*e.g.*, Flow tables)	all			
12	Secure Communication Channel	#A, #B, #D, #F, #G, #H, #I, and #J			

TABLE II: Main network resilience strategies.

Component	Id	Action description	Type	Dependency
Monitor	1	Analyzing network traffic, *e.g.*, packet and byte counts; or available delay or bandwidth.	Analyze	N/A.
	2	Monitoring network elements, *e.g.*, network services and functions; or network controllers; or VNFs.	Analyze	N/A.
	3	Triggering remedial action in a specific element, *e.g.*, NFV Orchestrator, VNF Manager, Controller.	Trigger	#1 or #2.
IDS/IPS	4	Analyzing anomaly. It uses a known signatures database. Initially, it only analyzes network traffic from Internet. However, the Monitor component can identify a possible anomaly from LAN, requesting for the IDS a traffic analysis.	Analyze	N/A or #3.
	5	Warning the occurrence of anomaly.	Trigger	#4.
	6	Dropping TCP/UDP protocol or source IP address.	Drop	#4.
Firewall	7	Masking destination IP address. Used to forge legitimate destination IP address.	Deceive	#3 or #5
	8	Forwarding source IP address. Used to forward source IP addresses for fake services or functions. Also used to forward network traffic to be handled by a specific VNF, *e.g.*, IDS/IPS, Monitor.	Deceive	#3 or #7.
	9	Dropping source/destination IP address.	Drop	N/A or #3.
	10	Dropping protocol, *e.g.*, HTTP, FTP, ICMP, Telnet.	Drop	N/A or #3.
HoneyPot	11	Forging network services. Used to forge legitimate network services, *e.g.*, Web, FTP, Mail services. It is set up with all the IP addresses of the legitimate network servers.	Deceive	N/A or #3 or #8 or #12.
Controller	12	Delegating decision-making logic to slave Controller.	Trigger	#2 or #3.
	13	Isolating network traffic. Used for VNF chaining.	Trigger	#3.
	14	Modifying master Controller. Used to indicate the (new) master Controller for a specific switch.	Trigger	#3.
Switch	15	Dropping source/destination IP address.	Drop	#3 or #5.
	16	Dropping all network traffic, *e.g.*, all traffic from Internet, LAN, DMZ. Used to isolate networks with anomalies.	Drop	#3 or #5.
	17	Dropping protocol, *e.g.*, HTTP, FTP, ICMP, Telnet. Used to drop unauthorized access.	Drop	N/A or #3 or #5.
	18	Duplicating TCP/UDP packet to be analyzed by a specific strategy, *e.g.*, #4, #11	Trigger	#3.
	19	Forwarding TCP/UDP packet to be handled by a specific VNF, *e.g.*, Monitor, Firewall, IDS/IPS, Controller, HoneyPot.	Trigger	#3 or #12.
	20	Modifying TCP/IP packet header. Used to forge fields of the TCP/IP packet header, *e.g.*, illegitimate source/destination MAC addresses, source/destination IP addresses.	Trigger	#3 or #12 or #14.

A. *ANSwer Prototype Overview*

This section describes the architecture proposed to combine NFV and SDN features for ensuring network resilience. Figure 1 shows an overview of the main components and modules that comprise this architecture. As a proof-of-concept, we chose network functions that can be combined to ensure the resilience of the network, such as firewall, IDS/IPS, and honeypot. Part of our choice is substantiated by the study of Patcha and Park [8], which shows that firewalls and IDSes/IPSes are complementary functions. On the one hand, firewalls establish policies which release or block access to certain services based on the TCP/IP header information, such as source and destination IP addresses, transport protocol, among others. However, firewalls by themselves do not have the ability to identify whether a network traffic is legitimate or malicious, because they are unaware of the traffic behaviors. On the other hand, IDSes/IPSes alert and block anomalous traffic based on known attack signatures. However, IDSes/IPSes alone do not have the ability of thoroughly filtering traffic that firewalls have. Finally, we also have added a honeypot to collect information from active anomalies during some time in order to analyze their behavior on the network, and establish which optimal strategies must be deployed. Our main goal is to show that with a small number of network functions, we can ensure network resilience. However, ANSwer is flexible and extensible to incorporate several network functions, which can further improve network resilience, such as Deep Packet Inspection (DPI), Anti-virus, and/or Anti-spam.

As can be observed, the components are distributed in all SDN planes. Components such as vFirewall, vHoneyPot, vMonitor, vIDS/IPS, and vControllers, perform one or more *resilience strategies* according to Table II. Components such as vControllers, NFV Management and Orchestration (MANO), Database, and VNF Repository comprise the architecture core. It is important to note that the Open vSwitches also perform *resilience strategies*. Such *resiliente strategies* are performed using the standard flow tables, and no customization in the Open vSwitches was performed. Details on each component and the phases where it is executed are described in the following:

vFirewall – analyzes network traffic, *i.e.*, packet-by-packet, to determine which actions must be performed in response to anomalies. It analyzes network packets coming from the Internet to determine which actions it must perform, *e.g.*, allow access to certain network services. It also introduces rules to forward suspect packets (identified by the network monitor and coming from anywhere, *i.e.*, Internet or LAN) to be further analyzed by an IDS. In addition, it presents rules

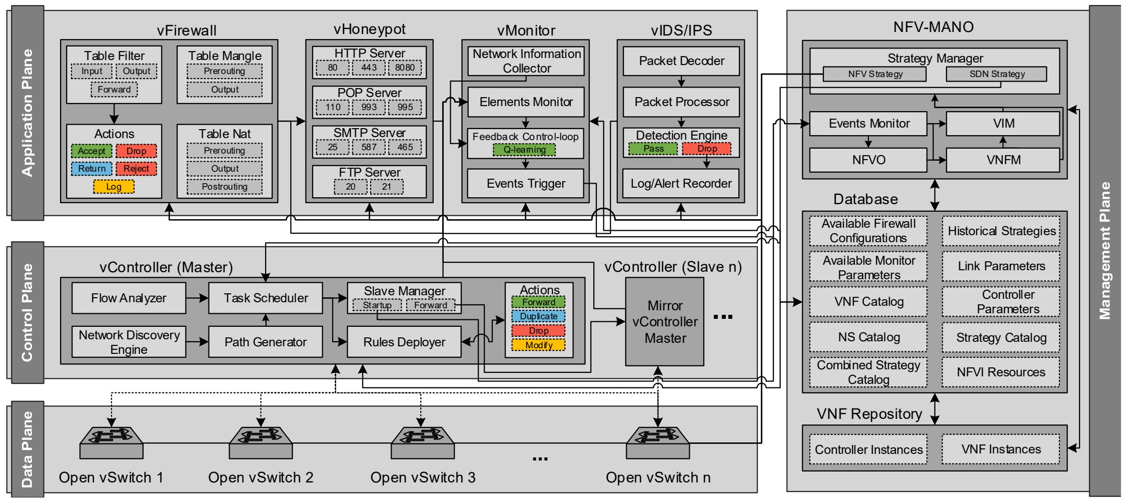


Fig. 1: Overall network resilience architecture.

for forwarding suspect traffic to fake servers and controllers. vFirewall executes in the Mitigation phase.

vHoneyPot – simulates network services, such as Web, FTP, POP3, and SMTP to deceive possible attackers in the Mitigation phase. We only use a honeypot to maintain attackers in a controlled environment, protecting legitimate services and resources. However, a honeypot can be used to collect information about the attack (*e.g.*, attack signature) and/or attacker (*e.g.*, source IP address) in the Detection phase.

vMonitor – in the Initialization and the Detection phases, it checks network conditions, such as link status (*e.g.*, enabled, disabled) and throughput of switches and links (*e.g.*, amount of packets and bytes forwarded by each switch); and service conditions, such as service status (*e.g.*, on-line, off-line) and response time (*e.g.*, standard, delay). Moreover, in the Detection phase, it collects information from the network elements to perform the feedback control-loop. This information is used to analyze and trigger actions in specific elements during the Mitigation phase, *e.g.*, (*i*) decision on network traffic behavior and gathering network information performed by network controllers; (*ii*) initialization of VNFs and decision on replacing strategies performed by NFV Management and Orchestration (MANO) elements.

vIDS/IPS – is divided into two modules/systems. The *Intrusion Detection System* (IDS) monitors network packets, analyzing both IP packet header and data field to identify anomalous traffic in the Detection phase. This identification is performed through a knowledge base (also called signature base), containing a set of previous attack signatures and known system vulnerabilities. In addition, it writes log files and/or sends alerts about the occurrence of anomalies to be analyzed by vMonitor. The *Intrusion Prevention System* (IPS), in the Mitigation phase, provides real-time remedial action in response to an anomaly.

vControllers – are divided into two types: (*i*) The vController Master, in the Initialization and the Detection phases, identifies network elements, such as switches, hosts, and links. In addition, it is responsible for gathering network information, such as number of hops, delay, and available bandwidth between network elements. Moreover, it deploys a set of rules with the idea of best-effort packet delivery, network isolation, and traffic forwarding; (*ii*) the vController Slave is an identical copy of the vController Master. It is used to replace a vController Master when it is down, and to manage a network segment that present an anomaly. vController Slave performs both in the Detection phase and in the Mitigation phase.

NFV Management and Orchestration (MANO) – is split into four main modules: (*i*) NFV Orchestrator (NFVO) manages and configures each network service (NS) and network function (NF), *e.g.*, it can write rules in the firewall for traffic forwarding; (*ii*) VNF Manager (VNFM) instantiates, terminates, and manages each VNF, *e.g.*, initiates a new controller instance; (*iii*) Virtualized Infrastructure Manager (VIM) analyzes and indicates to network controllers how the network must be configured to fulfill the VNF chaining, *e.g.*, after the firewall has carried out its functions, the traffic needs to pass by the IDS/IPS to be analyzed; and (*iv*) Strategy Manager analyzes and decides which strategies must be instantiated and in which elements they must be deployed. MANO continuously performs in all phases.

Database – stores both the information about the behavior of the network infrastructure (obtained by the vController and the vMonitor), network resilience strategies, and configurations for network elements. For example, the database stores all the possible links between two elements, amount of elements, and available bandwidth and delay. In addition, the database maintains a list of all network resilience strategies, *e.g.*, drop packet, forward traffic, analyze host. The information in this repository will be used later, by VNFs, vControllers, and MANO elements. It is used in all phases.

VNF Repository – stores both the VNF instances (vFirewall, vMonitor, vIDS/IPS, and vHoneyPot) and controller instances (vController). VNF Repository is used in all phases.

B. Experimental Evaluation

We present in this section experiments and initial results obtained with the implemented architecture. To perform the experiments, we created a case-study topology based on a university campus network and medium-sized datacenter networks (depicted in Figure 2). The topology and its elements are described in Table 2. In summary, the topology consists of an internal network composed of thirty two hosts, a DMZ comprising a few network services, a defense area containing the mechanisms for resiliency, and forty hosts simulating access via Internet. The topology was created using the Mininet

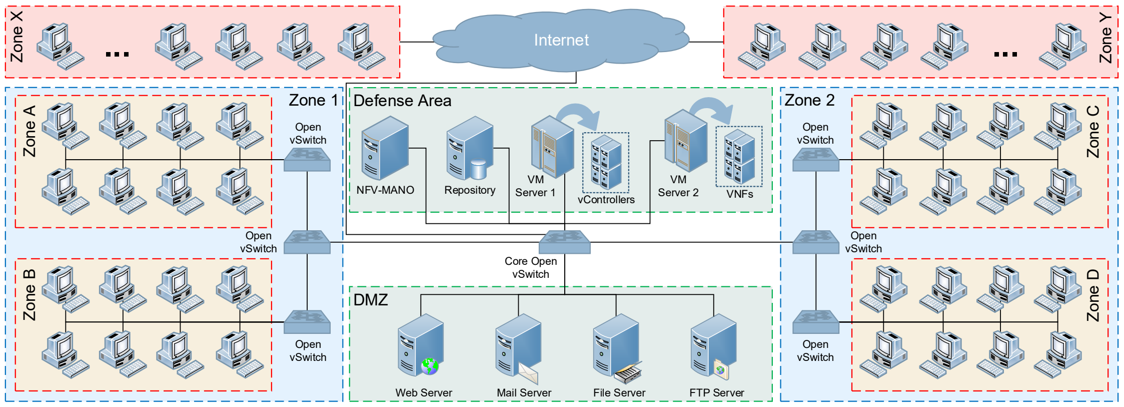


Fig. 2: Case-study topology.

TABLE III: Description of network elements used in the experiments.

Host/Server	Network	Location	Amount	Description
Hosts (clients)	Internal	Zone A	8	Simulate *known* network traffic (*known* packets and protocols, *e.g.*, HTTP[80, 443, 8080], FTP[20, 21], POP[110, 993, 995], and SMTP[255, 587, 465]). Also, they simulate *unknown* network traffic (*unknown* packets and protocols, *e.g.*, packets to random destination ports, *e.g.*, 223, 224, 225, 1220, 1234, 23415, 65211, ...). Moreover, they perform DDoS attacks.
Hosts (clients)		Zone B	8	
Hosts (clients)		Zone C	8	
Hosts (clients)		Zone D	8	
Hosts (clients)	Internet	Zone X	20	
Hosts (clients)		Zone Y	20	
NFV-MANO	Internal	Defense Area	1	Orchestrates and manages VNFs.
Repository			1	Stores network information and system configurations.
VM Server 1			1	Hosts virtualized controllers, *e.g.*, vController Master, vController Slave 01.
VM Server 2			1	Hosts VNFs, *e.g.*, vFirewalls, vIDSes/IPSes, vMonitor, vHoneyPot.
Web Server		DMZ	1	Performs the hosting service.
Mail Server			1	Performs mailing service.
File Server			1	Performs storing service.
FTP Server			1	Performs transferring file service.

emulator and experiments were performed on an Intel Core i7 3.6 GHz Octa Core with 16 GB RAM memory.

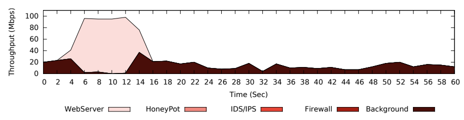
Before performing our experiments, we used a simulation of real traffic that contains several service types, such as HTTP, FTP, SMTP, POP, and many others, to train our *control-loop*. Based on our training, we have established an initial value of up to 10% above the standard deviation to update its value. Thus, any value that exceeds 10% will be considered an anomaly, and will not be used to update the mean and standard deviation information.

We executed three experiments that consisted of performing DDoS attacks, by combining attackers from inside and outside of the network in order to demonstrate the ability of the architecture to operate on different situations. For each experiment, we simultaneously performed three types of DDoS attacks against the web server: TCP Spoofed SYN Flood, TCP ACK Flood, and HTTP(S) GET/POST Flood. Each attack was randomly assigned to each attacker (host). Sometimes, we have a combination of attacks coming from different places. Each experiment lasted sixty seconds. The DDoS attacks were started five seconds after the beginning of each experiment. Each experiment was run thirty times, in order to obtain a 95% confidence level. The tools used to launch the attacks were SlowHTTPTest, h3ping and T50 Sukhoi PAK FA Mixed Packet Injector. During the experiments, approximately 12 Mb of background and legitimate traffic were generated to the web server using Scapy and iPerf3. Our main goals with these experiments are to discuss (*i*) the step-by-step deployment of resilience strategies when an anomaly is identified on the network; (*ii*) the importance of analyzing the behavior of legitimate flows while anomalies occur and how this analysis may influence on the choice of new resilience strategies; (*iii*) the result of each resilience strategy implemented; and (*iv*) the time taken to revert to the network normal state.

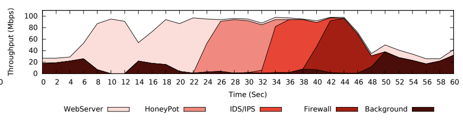
Figures 3(a), (b), and (c) represent, for zone A-B-C-D, zone X-A-B, and zone X-Y, respectively, the background traffic (according to Table III), the access to the web server (which is the attack target), and the throughput in the components of our prototype when the DDoS attacks are addressed to them. Figures 4(a), (b), and (c) represent, for zone A-B-C-D, zone X-A-B, and zone X-Y, respectively, the throughput in the switching elements. Due to space constraints, we will discuss the experimental results more generally.

As can be seen in Figures 3(a) and 4(a) the time taken to mitigate an anomaly exclusively coming from the internal network (zones 1 and 2) is very fast. Even for attacks of the Spoofed type (which mask the source address), our prototype identifies anomalous behaviors based on the mean and standard deviation of specific traffic features, by analyzing flows in each switch. Thus, since the traffic volume has grown rapidly in internal switches (Zones 1 and Zone 2), the prototype/ANSwer quickly deploys a strategy to drop the anomalous traffic at its source. As a result, in approximately 12 seconds the anomaly is contained. It is important to emphasize that the strategy only drops the anomaly addressed to a particular service, in this case, the web traffic. Thus, the traffic addressed to other services remain unchanged.

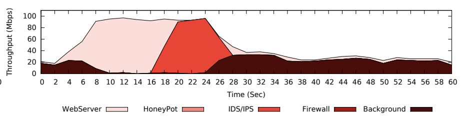
Additionally, when observing Figures 3(b) and 4(b) we conclude that the prototype takes around 44 seconds to completely contain an anomaly. In Figure 4(b) it is observed that at around 12 seconds a strategy for traffic containment in the internal switches (in this case, Zones A and B) was established. However, attackers positioned in the Internet increased the



(a) Attacks from Zone A-B-C-D

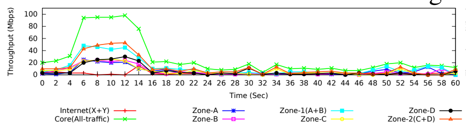


(b) Attacks from Zone X-A-B

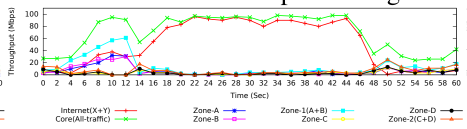


(c) Attacks from Zone X-Y

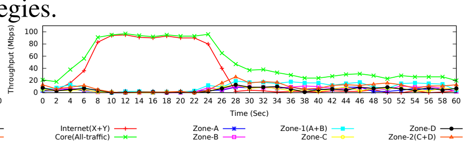
Fig. 3: Traffic behavior while performing strategies.



(a) Attacks from Zone A-B-C-D



(b) Attacks from Zone X-A-B



(c) Attacks from Zone X-Y

Fig. 4: Throughput in switching elements while performing strategies.

resources consumption. Consequently, the prototype deployed the initial sequence of strategies in order not to block possible false positives. Finally, when observing Figures 3(c) and 4(c), we conclude that the prototype established the initial sequence of strategies. However, given that the anomaly affected the background traffic almost completely, a strategy for dropping traffic at the source has been deployed.

IV. Related Work

Several researchers have been investing efforts on producing strategies for resilience in SDN environments. Schaeffer-Filho *et al.* [10] presents a framework based on management patterns that uses policies to decide which pattern should be applied for each situation. The framework provides abstractions (management patterns) to orchestrate resilience in network services deployed into distributed controllers. The main objective is to present a network management model that specifies how resilience mechanisms must be (re)configured to meet a certain network challenge, *e.g.*, DDoS attack. Fonseca *et al.* [11] emphasizes the use of replication techniques to achieve certain levels of resilience to controller devices in SDN. The authors investigate how different replication techniques can cooperate and how each one can perform specific tasks. Smith *et al.* [12] introduce a multi-level distributed architecture that allows the deployment of strategies and mechanisms to solve several resilience challenges. The framework includes implementation guidelines, processes, and toolsets that can be used to assist the design of resilience mechanisms at various levels in the network infrastructure. Although the above research efforts have achieved satisfactory results, these solutions rely solely on the use of SDN or NFV features to promote network resilience. Instead, we advocate an integrated approach that relies on a control-loop for continuously analyzing the behavior of the network infrastructure, by combining SDN and NFV features.

V. Concluding Remarks

Networks based on SDN and NFV are susceptible to malicious attacks and traffic anomalies that may compromise their resilience. To confront these issues, we introduce ANSwer, an architecture that combines NFV and SDN features to create a set of strategies for network resilience. The key aspect of our approach is a *feedback control-loop* that aims to identify the optimal strategy to resolve (or at least alleviate) a specific type of network anomaly. As main contributions, we present and discuss several ways to combine SDN and NFV features to produce more efficient strategies for ensuring network resilience. In addition, we show that with a small set of network functions we can produce several resilience strategies and different combinations. Further, the experimental evaluation demonstrates that ANSwer can rapidly identify and handle distinct anomalies in different scenarios, indicating that the reconfiguration and deployment of resilience strategies can be performed in real-time. As part of our future work, we plan to add to ANSwer other network functions, such as Load balancer, Deep Packet Inspection (DPI), and Antivirus, to validate the broader applicability of our work. We also intend to explore reinforcement learning algorithms to improve the feedback control-loop. Finally, we plan to extend our architecture to support other aspects of network resilience, such as disruption tolerance and security.

References

[1] J. Wickboldt, W. De Jesus, P. Isolani, C. Both, J. Rochol, and L. Granville, "Software-defined networking: management requirements and challenges," *Communications Magazine, IEEE*, vol. 53, no. 1, pp. 278–285, January 2015.

[2] C. C. Machado, L. Z. Granville, A. Schaeffer-Filho, and J. A. Wickboldt, "Towards SLA policy refinement for QoS management in software-defined networking," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, May 2014, pp. 397–404.

[3] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245 – 1265, 2010, resilient and Survivable networks.

[4] NFV, "Network functions virtualization - an introduction, benefits, enablers, challenges and call for action," 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[5] J. Ferrer Riera, E. Escalona, J. Batalle, E. Grasa, and J. Garcia-Espin, "Virtual network function scheduling: Concept and challenges," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*, June 2014, pp. 1–5.

[6] A. S. d. Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in sdn," in *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, Sept 2015, pp. 134–141.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul 2009.

[8] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[9] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609 – 8624, 2015.

[10] A. Schaeffer-Filho, P. Smith, A. Mauthe, and D. Hutchison, "Network resilience with reusable management patterns," *Communications Magazine, IEEE*, vol. 52, no. 7, pp. 105–115, July 2014.

[11] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "Resilience of SDNs based on active and passive replication mechanisms," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Dec 2013, pp. 2188–2193.

[12] P. Smith, D. Hutchison, J. Sterbenz, M. Scholer, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, "Network resilience: a systematic approach," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 88–97, July 2011.