








Self-healing in the Scope of Software-Based Computer and Mobile Networks

Natal Vieira de Souza Neto^(✉) , Daniel Ricardo Cunha Oliveira ,
Maurício Amaral Gonçalves , Flávio de Oliveira Silva ,
and Pedro Frosi Rosa 

Faculty of Computing (FACOM), Federal University of Uberlândia (UFU),
Uberlândia, Brazil

{natalneto,drcoliveira,mauricioamaralg,flavio,pfrosi}@ufu.br
<https://sonar.facom.ufu.br/>

Abstract. Self-healing is an autonomic computing fundamental well-disseminated in standalone computer systems. In distributed systems, e.g. computer networks or mobile networks, the introduction of self-healing capabilities poses some challenges, mainly when software-based networks, e.g. Software-Defined Networking (SDN) and Network Functions Virtualisation (NFV), are involved. Such networks impose new control and management layers, and the adoption of self-healing functions means that all layers must be considered. In this paper, we present the challenges of self-healing in the scope of SDN and NFV, by revisiting the self-healing concept in computer and mobile networks, and by presenting the thorough difference between a system that applies fault tolerance from one that applies self-healing functions. We also introduce a framework for solving these challenges, by describing four use cases of self-healing, considering control, management, and data layers. The use cases focus on maintaining the health of the network at run-time, considering control, management, and infrastructure layers. Our framework is a novel Operations, Administration, and Maintenance (OAM) tool, based on a self-management network architecture that was introduced in our previous works.

Keywords: Self-management · Self-healing · Fault tolerance · OAM · SDN · NFV

1 Introduction

Future computer and mobile networks must simultaneously support diverging network requirements, e.g. low-latency, high-throughput, reliability, etc. Software-Defined Networking (SDN) and Network Functions Virtualisation

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (Capes) - Finance Code 001.

© Springer Nature Switzerland AG 2021

D. Ferguson et al. (Eds.): CLOSER 2020, CCIS 1399, pp. 325–344, 2021.

https://doi.org/10.1007/978-3-030-72369-9_14

(NFV) are key enablers for this supporting, as these technologies facilitate network automation, virtualisation, and service composition [22], which are crucial for delivering dynamic allocation and reallocation of resources for critical requirements. Although the advances in specifications and implementations for these technologies in the last years, SDN and NFV have unresolved problems associated with network healthiness, which are discussed in this paper, e.g. controller channel maintenance, bugs and crashes in network applications, distributed resources management, and so on [3, 8].

Before SDN and NFV, distributed protocols running inside the network infrastructure were responsible for network health maintenance. Such distributed protocols, like Open Shortest Path First (OSPF), resolve the network impairments that lead the network to the unhealthy state, e.g. link congestion, network overload, and inconsistency routing rules [16]. SDN and NFV approaches bring new network layers or planes, i.e. the control plane and management plane, with software components for operating in the environment. Such new network planes operate over the data plane, for ensuring the provisioning and maintenance of network resources, including routes in multi-path topologies, compute resources, e.g. storage, memory and processing power, and radio resources. Common SDN and NFV components, e.g. SDN Controller (SDNC) and Virtualised Infrastructure Manager (VIM), conveniently have mechanisms for supporting data plane failures; however, the communication maintenance between the data plane and control/management planes as well as the resilience of control/management components are open issues [17].

In this paper, which is an extended version of our previous work [20], we discuss the autonomic fundamental of self-healing applied to softwarised and virtualised networks. By using SDN and NFV technologies we propose a self-healing framework capable of acting on data, control, and management planes, and we add new use cases for self-healing in comparison to the original paper. To the best of the authors' knowledge, our framework is the first solution that addresses fault issues in all network layers for these technologies. The main objective of our solution is to ensure the network health even with unexpected behaviours in any network planes. The framework acts as an Operations, Administration, and Maintenance (OAM) tool, autonomously applying monitoring and recovery functions.

Autonomic computing fundamentals, e.g. self-healing, self-configuration, and self-optimisation, empower the self-management of computer and mobile networks, which is considered the future of the Network Management (NM) [18]. By the self-healing fundamental, the network must recover itself from faults that can culminate in a degraded or broken performance. This concept allows the network to maintain the Quality of Service (QoS) of applications running in the environment, avoiding these applications to achieve a degraded or broken state. Our framework covers different autonomic computing fundamentals, well-known as self-* – self-optimisation, self-configuration, self-protection, etc. – and, in this work, we chose the self-healing as the start point to evaluate the solution.

The remainder of the paper is structured as follows. In Sect. 2, we discuss the fault tolerance principle in data, control, and management planes; and position our contribution with some related works. In Sect. 3, we present our framework with technical details. In Sect. 4, we describe some use cases of our framework in an SDN/NFV environment and discuss them concerning the QoS point of view. Finally, in Sect. 5, we pose our concluding remarks and suggest some directions for future work.

2 Background

In computer science, a system that applies the self-healing concept can detect or predict faults that lead the system to a nonoperational state [9]. The system usually notices the fault and recovers itself before a completely broken state. In computer networks, this self-healing capability is complex due to the number of diverging nodes – which are Network Elements (NEs), including routers, switches, gateways, proxies, firewalls, etc. – running simultaneously in the network topology [21]. As every NE has its particularities and diverse protocols run in the environment, the detection and recovery of failures are more complex in computer networks than in standalone systems.

A computer network is a graph with NEs representing the nodes and links between the NEs representing the edges. Additionally, nodes can represent hosts or User Equipment (UE) that are connected in the NEs. In traditional architectures, before network softwarisation and virtualisation, the network is healthy when the communication between hosts and UE has no problems. However, some impairments can deteriorate the network health, such as broken/congested links, hardware/software failures, bugs in network applications, etc. [16]. In the current and future computer and mobile networks, the network is unhealthy if QoS requirements are not satisfied, even if the communication is established. Moreover, considering SDN and NFV, new graphs are abstracted in the environment, representing the topologies between the data plane with control and management planes. Such QoS requirements in the data plane and the new graphs pose more complexity in the environment, which requires autonomic functions as self-healing. In this section, we detail the complexity in data, control and management planes, and discuss the self-healing fundamental in these layers.

2.1 Fault Tolerance in the Data Plane

In an SDN topology, the data plane is the layer in which the NEs are placed. For NFV, the compute nodes where network functions run represent the data plane. Considering Fig. 1 as an example, NE_1, NE_2, \dots, NE_8 are SDN NEs whilst Compute Node₁ and Compute Node₂ abstract the representation of NFV data layer components. The network functions running inside the compute nodes are connected by the NEs. In current mobile networks, Network Slices (NSs) [4] are abstracted by the components shown in the Data plane. The SDNC₁ provides the connections between different nodes in the data plane, applying routes inside the

NEs. Such routes are defined and created by SDN Applications running on the top of SDNC₁, through the Northbound Interface (NBI). The network functions running inside the compute nodes, e.g. Virtual Network Functions (VNFs), are instantiated by management components, e.g. OAM₁.

From the fault tolerance point of view, data plane issues are well-solved by current solutions because the control plane components are responsible for the operation of the data plane. It means that if NE₇, shown in Fig. 1, presents failures, the SDNC₁ will recalculate and apply routes avoiding such NE. Analogously, if the Compute Node₁ is overloaded, the OAM₁ will recreate new network functions into the Compute Node₂, and it will even migrate current network functions that are presenting problems. As the most known components found in the literature already have mechanisms for dealing with the data plane components, the focus in our framework is associated with QoS.

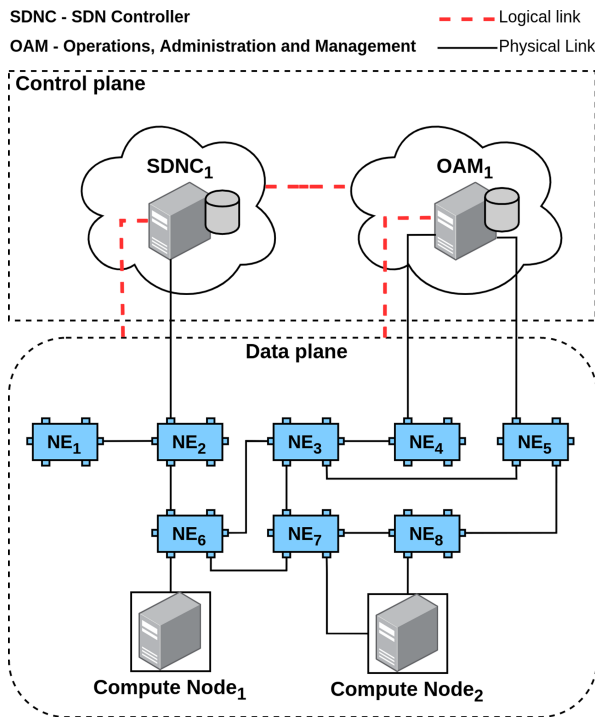


Fig. 1. A layered SDN and NFV typical environment. The Control plane contains operational and management components, represented by SDNC₁ and OAM₁, and the Data plane contains network components and computational infrastructure elements, represented by NEs and Compute Nodes respectively.

Even with solutions for fault tolerance in the data plane, applications with particular QoS attributes require specific fault tolerance actions. For example, Ultra-Reliable and Low Latency Communications (URLLC) applications

demand strict latency during the whole service lifecycle [7]. In this situation, the applications require different attributes: one application can require 1 ms of latency, while another can require 10 ms. If the communications between such applications and end-users are using the same network link and this link has a delay of 5 ms, the first application is not running properly, but the former application has no problem. It means that services and applications with specific QoS requirements, e.g. reliability, low-latency, or high-throughput, are healthy if the network is delivering such requirements.

For future computer and mobile networks, e.g. 5G, Mobile Network Operators (MNOs) will probably share their infrastructure in several logical networks, named NSs [23]. In this approach, network orchestrators are responsible for provisioning each NS throughout the different network segments, e.g. Core Network (CN) and Radio Access Network (RAN). The orchestrators have information about the available resources and orchestrate such resources in a better way, ensuring that the required QoS will be delivered. The OAM tools ensure the continuous delivery of the services with the required QoS. In this way, the data plane must be monitored during the run-time and resource reallocation and network reconfiguration must be performed when the network pass through health problems, i.e. network impairments, such as network overload, is affecting the QoS of some NSs.

In this paper, the network self-healing framework, defined in Sect. 3, takes into account the QoS requirements from NSs for guaranteeing the QoS during the NS entire lifecycle. On the other hand, for fault tolerance of data plane components, e.g. link failures, the framework has no actions, because it is assumed that the SDN and NFV components are capable to deal with infrastructure impairments. Besides the data plane, the control and management layers are under academic investigation, since SDN and NFV were not designed with intrinsic management functions for fault tolerance of the control and management layer themselves [17].

2.2 Fault Tolerance in the Control and Management Planes

Former computer network architectures contain distributed protocols that are responsible for real-time operation; such protocols are responsible for fault tolerance in the data plane. In such architectures, network functions are performed in the same layer in which the data traffic is carried out, as there is no separation between control, management, and data planes. As an example, if a link between two NEs is having problems, the routing protocols, e.g. OSPF, modify the routes to avoid degradation of the link. In SDN, the routing function is performed by SDN Applications that run on top of the SDNC. In this way, the routing application ensures the modification of data plane routes. As pointed in Subsect. 2.1, this mechanism avoids or mitigates data plane failures. However, the integration between the data plane and the control plane, in which the SDN Controller and Applications are placed, requires an uninterrupted communication channel [17].

Figure 1 shows an SDN Controller, i.e. SDNC₁, controlling some NEs, i.e. NE₁, NE₂, ... NE₈; it is assumed that some SDN Applications are running on

top of SDNC₁. As the SDNC₁ controls the NEs from the data plane, it has logical connections, represented by the dashed lines, with all these NEs. However, it has a physical connection, represented by the continuous lines, only with NE₂. The communication between SDNC₁ and NE₇ occurs through the paths SDNC₁-NE₂-NE₆-NE₇, SDNC₁-NE₂-NE₆-NE₃-NE₇, or SDNC₁-NE₂-NE₆-NE₃-NE₅-NE₈-NE₇. Considering the topology in Fig. 1, if NE₂ has a problem, or the link SDNC₁-NE₂ is broken or degraded, the entire data plane will be non-operational since the decisions of routing and forwarding are made by SDNC₁ and its applications. As stated, an SDN environment requires fault tolerance actions for the communication inter-layers. For self-healing purposes, as the network must recover itself from problems, it must be able to recover from data, control, and management issues.

The aforementioned inter-layer communication is the first aspect of the self-healing of the control plane. The second aspect is associated with the control components. The SDNC₁ and its applications are software modules and can present bugs and crashes [8]. For achieving the self-healing fundamental, the network must recover these components from bugs or crashes, as well as increase the computing capability when the network presents overload. The auto-scale of control components is essential. Nonetheless, the auto-scale of data plane components (NEs) is another requirement if such elements are virtual switches (which is not the exemplification of Fig. 1).

From the NFV perspective, the communication between management and data planes components is similar to the aforementioned SDN approach. In Fig. 1, the OAM₁ could represent a VIM Controller, and the infrastructure where the virtual machines are deployed is on the represented compute nodes. Following the previous analogy, OAM₁ has logical connections with the Compute Node₁ and Compute Node₂, but the physical connection is made by the links from the data plane topology. It means that any overload in the NE₆ can compromise the communication between the OAM₁ and its managed virtual machines. The figure does not represent all NFV components, but the connectivity between the controllers/orchestrators is similar: a VNF Manager (VNFM) requires uninterrupted communication with its managed VNFs; a VIM controller requires uninterrupted communication with the compute nodes in the NFV Infrastructure (NFVI); etc.

In addition to the communication aspect, self-healing in an NFV environment must act over software and hardware problems. As an example, VIM tools commonly used in industry, e.g. Openstack [15] or Kubernetes [11], have mechanisms for self-healing and auto-scale of the containers and virtual machines previously instantiated. However, the VIM tool itself usually has no mechanisms for healing from problems. It means that a problem in the OAM₁ will leave Compute Node₁ and Compute Node₂ without management.

To achieve the self-healing fundamental, the network must also consider the placement of components. The SDNC₁ can run on virtual machines and, in this situation, it can be instantiated in different places in the topology. In Fig. 1, if the control traffic is bigger in NE₇, maybe the SDNC₁ could be instantiated

inside the Compute Node₂. As stated in this section, the self-healing functions must consider communication, bugs and crashes in software and hardware, and components placement. Additionally, the number of aspects is not exhaustive and, for this reason, we designed a framework capable of deal with the three exemplified aspects, but also architectural capable of deal with new aspects in the future. In Subsect. 2.3 we position our contribution concerning other frameworks.

2.3 Related Work

SDN and NFV technologies bring many advantages for network and infrastructure operation [22]; nonetheless, there are some unresolved problems in these technologies, mainly associated with network and service management [16]. SDN transfers the operation over the data plane for the control and application layers, whilst the management of such layers is not standardised. NFV introduces new components for the management of virtual functions, whilst the management of the components themselves are not standardised as well. Besides the framework proposed in this paper has mechanisms to act in all layers, the focus is on the control and management layers.

The first unresolved problem in SDN and NFV is associated with the controller channel reliability, i.e. the communication between controllers and their controlled elements [16, 17]. In SDN, the SDNC controls all the NE in the domain. In NFV, management components handle various virtual functions, including creating, maintaining and decommissioning such functions. For these reasons, the network must ensure the connectivity between the SDNC and the NEs, as well as the communication between NFV components and compute nodes in which the VNFs are placed. The Southbound Interface (SBI) protocol, i.e. the protocol for the communication between the control and data planes, e.g. Open-Flow [13], usually have procedures to create the paths between all NEs and the SDNC when every NE starts. The problem is when routes on such paths are not available: management functions must reroute previous paths. In the NFV, there are no default paths, hence the routes are dynamically defined at run-time, which means that network overload can compromise the control and management communication.

In literature, there are some works for solving the controller channel reliability in SDN, such as [5] and [6]. The first presents some functions for monitoring and recovery of control paths, i.e. paths between the NEs and the SDNC. The former introduces a control path migration protocol for virtualized environments, focused on control paths as well. To the best of our knowledge, there are no works considering other control and management components instead of the SDNC, such as the SDN Applications and NFV controllers and orchestrators. Our solution aims to build a catalogue of control and management entities, relating them to their controlled/managed elements. In this way, the network can achieve the self-healing fundamental, as the catalogue has information about what are the management relations in the environment.

Another challenge in SDN and NFV environments is the definition of control, management, and data components. In general, virtual environments allow the deploying of control/management components in the same infrastructure of data components. As an example, the Compute Node₁ in Fig. 1 can have a virtual instance, e.g. a Virtual Machine (VM), used for data processing, as well as the controller or orchestrator managing this VM. From the network topology point of view, even with the separation between control and data, as proposed by the SDN approach, the traffic of control and data primitives utilises the same NEs. It means that the control primitives are not processed by the NEs, but the traffic of such primitives is in-band, i.e. the same NEs used for forwarding data primitives are used for forwarding control primitives [19]. The deploy of out-of-band topology for control and management is not feasible as it is expensive and complex [19], and hence, a self-management solution must consider that data and control will share the same infrastructure. The key is the identification of control and management traffic and components, and prioritisation of them. The catalogue proposed in this paper is a novel solution for this achievement.

A third unresolved problem in the software/virtual-based network environments is the QoS of End-to-End (E2E) applications. In general, the projects found in the literature are considering Network Slicing as a Service (NSaaS) [23] as a key enabler for the acceptance of SDN and NFV technologies. In this way, the management functions are responsible for provisioning and maintenance of NSs that contain strict QoS attributes. There is not any unique solution for the operation of these NSs during the entire NS lifecycle. From the maintenance at run-time perspective, a 3GPP working group defined self-healing as essential [2].

The number of unresolved problems in software and virtual networks is not exhaustive, as management aspects are usually not considered at the design step in network solutions. For this reason, we use a self-management architecture as the starting point to build a framework capable of dealing with the problems described in this section, which are under investigation in other research groups, as well as future problems that future applications will pose.

3 Self-healing Framework

For applying self-healing in SDN and NFV environments, we built a framework based on Self-Organising Networks Architecture (SONAr) [10], which is an architecture for self-management of computer and mobile networks inspired by Self-Organising Networks (SON) [1] concepts. In this section, we first describe the framework architecture and show the placement of the self-* functions; we next detail the monitoring functions of our solution, followed by the recovery functions details; and finally, we present the Management Catalog and discuss some use cases.

3.1 Framework Architecture

In this section, we present some technical details about the project we have developed from our previous position paper. The SONAr has capabilities for self-configuration, self-protection, self-optimisation, and so on; however, we focus

on the self-healing fundamental in this paper. Figure 2 shows a layered vision of our framework. In SDN and NFV, the Infrastructure Layer contains NEs, e.g. switches, and compute elements, e.g. servers. The Control Layer has the SDN components, e.g. controller and applications, and NFV elements, e.g. controller and orchestrators. Separated from the Control Layer, we present the Management Layer, in which our framework operates.

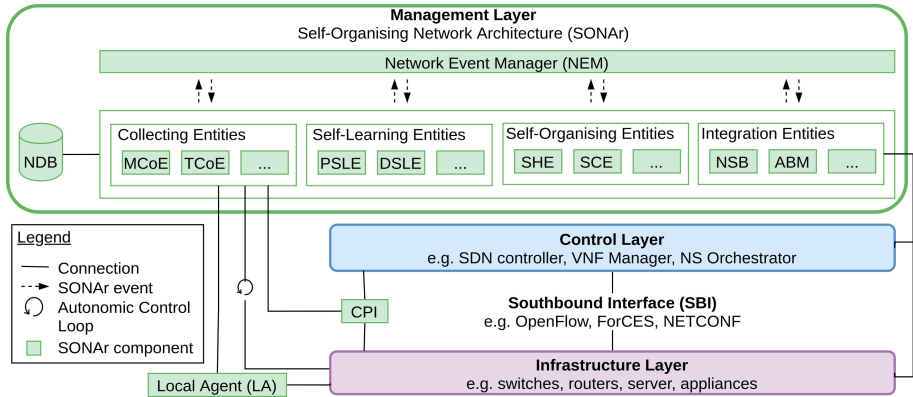


Fig. 2. The framework insight-architecture, i.e. SONAr. The Control and Infrastructure layers are monitored by the Management Layer. The collected data is transformed into events and the modules for self-management, i.e. SLE, SOE, and IE, address such events for real-time and prediction-based recovery.

From the self-healing aspect, the operation in any network topology is commonly performed by OAM systems and divided into two phases: (i) monitoring and (ii) recovery. In (i), the OAM framework, e.g. SONAr, retrieves information from the infrastructure. In (ii), the retrieved information is analysed for detecting or predicting failures and actions are executed for recovering from these potential/eventual failures.

3.2 Monitoring of Different Network Layers

SONAr applies three different monitoring techniques, as summarised as follows:

- Autonomic Control Loop (ACL): Every Collecting Entities (CoE) applies ACL for retrieving information from both Control Layer and Infrastructure Layer periodically, e.g. the Metrics CoE (MCoE), responsible for collecting metrics, can collect statistic usage from NEs' ports in the Infrastructure Layer or memory usage from an SDN application in the Control Layer.
- Local Agents (LAs): The LAs are placed inside the NEs in the Infrastructure Layer, for summarising information and sending it to the CoEs. This technique avoids the high cost of control loops and flooding.

- Interception of primitives in the SBI: an entity named Control Primitives Interceptor (CPI) is deployed as a proxy between the data and control planes. The CPI intercepts control primitives and send them to the CoEs. If the primitives are malicious or cause a crash in the destination (control components), SONAr can recover the destination.

Each aforementioned technique has advantages and disadvantages. According to our experience, ACL and LA techniques are sufficient to monitor the entire SDN and NFV environment. Additionally, the CPI is a novel contribution from SONAr and enables the recovery of crashes and bugs caused by control/management primitives.

The CoEs transform the collected/received information in events that are published in the Network Event Manager (NEM) publisher/subscriber platform. NEM works with topics and each event is categorised in a specific topic, e.g. the collected metrics are published in the topic named ‘metric’, and any other entity in the framework can subscribe to one or more topics. In this way, the collected events or the generated events, which are created by SONAr’s components, are sent and received by any entity. From the self-healing point of view, the Self-Healing Entity (SHE) and Self-Learning Entities (SLEs) are essential for network recovery. The first has algorithms for real-time analysis and recovery actions, whilst the former has Artificial Intelligence (AI) algorithms for prediction actions that can avoid or mitigate future failures.

3.3 Recovery by Detection and Prediction

The recovery in SONAr is based on the analysis of every event travelling in the framework. For some events, the framework has no immediate actions, and such events are just stored in a distributed structure, i.e. Network Database (NDB). The SLEs performs analysis and Machine Learning (ML) algorithms using the stored information. For other events, urgent actions are necessary, e.g. a link down event. The SHE, which is a group of microservices for network healing, treats the real-time events. The basic microservices in this entity are summarised as follows: the topology microservice, which is subscribed to topology information topics, is responsible for building the network topology and logical control/management topology between data plane NEs and control/management components; the catalogue microservice, which creates a catalogue containing the information about the NEs and the components responsible for the management of such elements; the path engine microservice, which run routing algorithms to define the paths between every NEs and virtual function and their control/management components; the metric microservice, which analysis every collected metric to define whether such metric represents a failure (or a possible failure); and finally, the recovery microservice, which is responsible to build necessary commands to recover from a failure.

After the aforementioned analysis, the SHE has a list of commands that will be requested to control and manage components primarily, or to infrastructure elements directly, when control and management components are not available.

As an example, if a control primitive containing a link down warning is intercepted by the CPI, the CPI sends the primitive to the Topology CoE (TCoE), which transforms this primitive into an event and publish it into a topic 'topology' in NEM. SHE microservices, subscribed to the topic 'topology', receive the event. The topology microservice can decide whether this primitive will affect the communication between the SDNC and some NEs. In a positive situation, the recovery microservice builds new commands, which are placed into new events. Such new events are published into NEM, in a topic named 'recovery commands'. The events are received by all microservice from any entity that is subscribed to the topic 'recovery commands'. The Network Service Broker (NSB), responsible for the integration between SONAr and the components into Control and Infrastructure Layers, receives the events and builds command requests that are sent to the bottom layers. In the current example, a telnet command is necessary to modify the control paths between the affected NE and the SDNC.

The Self-Organising Entities (SOEs), divided into SHE, Self-Protection Entity (SPE), Self-Optimisation Entity (SOPE), and Self-Configuration Entity (SCE), focus on real-time actions. For fault tolerance, a real-time action means detection of failures in a reactive perspective, i.e. the failure already occurred and will be treated. However, for critical applications and services, e.g. URLLC or Enhanced Vehicular to Everything (eV2X), the time for detection and recovery can not be enough for maintaining the QoS of such applications. For this reason, SONAr has specified the SLEs, responsible for analysing the information stored into NDB to find behaviours that can implicate in future failures. If a future failure is predicted, the framework can mitigate or, in most cases, avoid such failure. A common example of this situation is the congestion. Since the LAs, ACL, and CPI provide information about metrics, a link usage metric is commonly stored into the NDB. In this scenario, if a link has a linear growth, it is possible to determine when the link will be congested.

The reason for the division between SOEs and SLEs is associated with the computing cost of AI algorithms. As some failures require immediate recovery actions, the SOEs need high performance for analysing and processing recovery actions as soon as possible. On the other hand, AI algorithms, e.g. ML procedures, can spend high processing; therefore, the SLEs are separated to not interfere in real-time actions from SOEs. Any entity listed in Fig. 2 can communicate with any other by events. Besides the SHE, which groups the healing microservices, other entities can also participate in the recovery process. As an example, if the SHE defines that a reboot is necessary for some NEs, the SHE will publish the recovery event into NEM. This event will be received by the SCE, which has the best strategies for network bootstrapping. The SCE can generate a new event, which will be received by the Auto-Boot Manager (ABM). The ABM has particular bootstrapping procedures and will send the necessary commands to Control and Infrastructure Layers.

3.4 The Network Database

The NDB is a NoSQL database that contains operational information used by SONAr and the events that are collected and generated by the framework. Additionally, the Management Catalog is stored in this structure and requires some details. First, SONAr CoEs retrieve the information about the topology and stores in NDB. Then, the ABM and NSB retrieve meta-information from controllers and orchestrators running in the environment. Such information allows the identification of a controller and its controlled NEs, as well as the identification of an orchestrator and its managed resources. This meta-information is important because enable the SHE and SLEs to detect and predict failures.

As an example of the Management Catalog, an SDNC, e.g. SDNC_1 , controlling a topology with four NEs, e.g. NE_1 , NE_2 , NE_3 , and NE_4 is considered, in addition to another SDNC, e.g. SDNC_2 , controlling another topology with two NEs, e.g. NE_5 , NE_6 . In the catalogue, some rows are indicating what are the controllers and their managed elements. As the catalogue is a 2-tuple, the example is a set as follows: $\{(\text{SDNC}_1, \text{NE}_1), (\text{SDNC}_1, \text{NE}_2), (\text{SDNC}_1, \text{NE}_3), (\text{SDNC}_1, \text{NE}_4), (\text{SDNC}_2, \text{NE}_5), (\text{SDNC}_2, \text{NE}_6)\}$. For NFV components, SONAr keeps the information inside the catalogue as well. For example, a VNFM, e.g. VNFM_1 , that had instantiated two VNFs, e.g. VNF_1 and VNF_2 , is stored in the catalogue as follows: $\{(\text{VNFM}_1, \text{VNF}_1), (\text{VNFM}_1, \text{VNF}_2)\}$. The 2-tuple is the key from the catalogue rows, but the column families have many meta-information, indicating the logical addresses, type of NFV components, type of SDNC, vendors, versions, and so on.

The catalogue also stores properties information, which enables that SHE recovery microservice to build recovery commands. As an example, an Open Network Operating System (ONOS) controller has an open rest-Application Programming Interface (API) different from the OpenDaylight (ODL) controller [14]. With the properties information, the SHE can build the proper commands to request the open APIs from different SDNCs and different NFV components as well.

4 Use Cases

As exposed in the last section, we provided a self-management framework based on SONAr specification, allowing the implementation of different self-healing functions. The modular and flexible characteristics of the framework enable the monitoring and recovery of several scenarios of failures. In this section, we present four use cases focused on the control and management connectivity, showing the framework capacity. Nevertheless, there is a non-exhaustive number of use cases that the framework can treat, as the reference model allows the creation of new microservices on demand.

4.1 Control Paths Use Case

The Control Paths use case is associated with the logical connection between NEs, placed in the data plane, and the SDNC and its applications, placed in the

control plane, i.e. the SBI connectivity. Figure 3a shows a basic SDN multi-path topology with four NEs connected, NE₁, NE₂, NE₃, and NE₄, and an SDNC, i.e. C₁. In this topology, C₁ has a single link connection with NE₁, which means that the communication between C₁ with NE₂, NE₃, and NE₄ is not directly, e.g. the control primitives sent from C₁ to NE₄ will follow by one of the routes NE₁-NE₄, NE₁-NE₂-NE₄, or NE₁-NE₃-NE₄. The OpenFlow protocol, which is a well-known SBI protocol in literature, has procedures for creating the initial flow rules between a NE and the controller at the initialisation time (bootstrapping). In the current example, the initial paths are represented as dashed lines in Fig. 3b. As the initial path from C₁ to NE₄ is C₁-NE₁-NE₄, all control primitives from NE₄ to the SDNC will follow throughout the path NE₄-NE₁-C₁; the other NE have analogous behaviours.

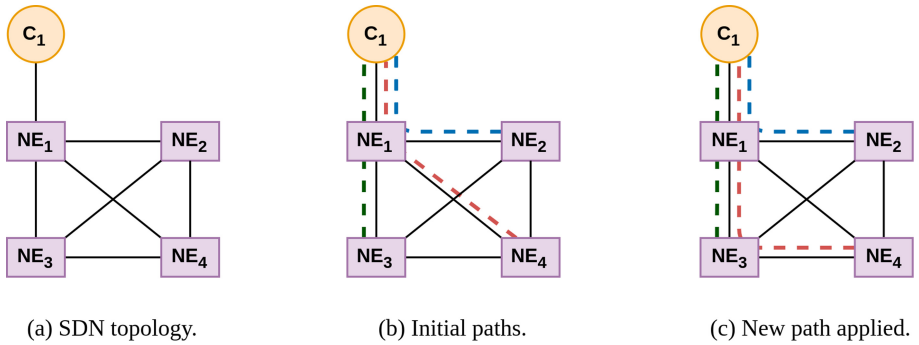


Fig. 3. SDN control paths use case. At bootstrapping time, the SBI protocol has procedures for creating the initial logical paths from the controller to every NE, as shown in (b). If a degradation/failure in a link happens, e.g. in the link NE₁-NE₄, SONAr framework recalculates and applies a new logical control path, avoiding the degraded/broken link as shown in (c). Adapted from [20].

As the OpenFlow protocol defines the initial control paths, SONAr does not interfere in the NE bootstrapping in this situation. When the network is operational, SONAr retrieves the topology and logical paths information by using the SDNC open APIs: the NSB requests a service to the SDNC and stores the information, received in the response, in NDB. If the topology changes, the TCoE receives the changing information and updates the NDB. At this point, the SHE path engine microservice calculates all possible paths between C₁ and every NE. The CoEs, LAs, and CPI keep monitoring the data, control, and management planes. If a link degradation/broken is recognised by the SHE topology microservice, the SHE recovery microservice creates recovery commands for applying an alternative path. This situation is shown in Fig. 3b and c, in which the link NE₁-NE₄ presents a degradation. The NSB receives the recovery command events and applies flow rule modifications in the NE₁ and NE₄, and add flow rules in NE₃. Figure 3c shows the new logical path applied, avoiding the link NE₁-NE₄.

This use case has other scenarios that we can explore. A second scenario is overload or congestion. If the link NE_1 - NE_4 is not broken, it could be congested at some time, caused by an overloading of NE_1 or NE_4 , or even high traffic in this link, caused by data plane communication. As the MCoE keeps monitoring metrics such as link usage, the Prediction SLE (PSLE) keeps analysing such metrics at run-time. If by any reason the PSLE predicts congestion in any link, e.g. NE_1 - NE_4 , it publishes an event in NEM, then SHE receives this event and the SHE recovery microservice proactively defines recovery actions. In this situation, the logical control path between C_1 and NE_4 is modified before the congestion begins.

Diverse other scenarios are possible in the control path use case. The SOE must ensure uninterrupted communication between the SDNC and its controlled NE, as well as between SDN applications and the SDNC, which is not represented in Fig. 3. To ensure the availability of the paths, it is important that SONAr microservices first apply the flow rules for the new paths, and then remove the old flow rules. In this case, NEs duplicate some control primitives and the SDNC can receive both primitives. Assuming that the SDNC discards the duplicated primitives, this procedure is better than the opposite (first remove old flow rules to then create the new ones), because no control information is lost. Other use cases considering the control paths are enabled by SONAr, but they are not described in this document.

4.2 Management Paths Use Case

Another use case we selected is the management paths example. The control path use case applies to routes between the SDNC and its controlled NEs. However, in NFV there are other control/management components that we named OAM in this paper, e.g. a VNFM provisions and manages diverse virtual functions, i.e. VNFs. As SONAr primarily focuses on management traffic, any OAM requires uninterrupted communication with its managed resources. Figure 4a shows the network topology with four NEs, and additionally, a VNFM, i.e. V_1 , managing two VNFs previously created, i.e. VNF_1 and VNF_2 . In this scenario, the complete topology must consider the NEs topology and the NFV topology, which has different available logical paths connecting V_1 with VNF_1 and VNF_2 .

Unlike the Control Paths use case, in the Management Paths use case there are no initial paths. The communication routes between V_1 with its VNFs are defined at run-time, by the routing application running on the top of the SDNC, or by the distributed routing algorithms placed inside the routers in traditional architectures. At first, there is no problem with this real-time routing mechanism. However, SONAr is prepared to provide functions to configure static and primary routes. The highest priority is given to management routes, i.e. the routes for communication between the management components and their managed resources. For this, there are two additional functions performed by SONAr entities: (i) the creation of the catalogue containing the management components and their managed resources; and (ii) the definition and application

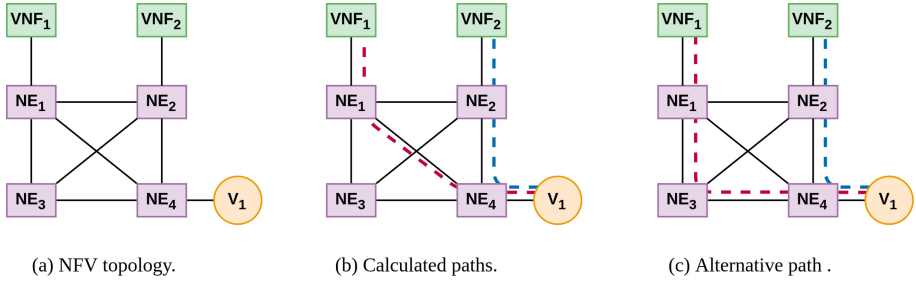


Fig. 4. NFV management paths use case. A VNFM component has instantiated two VNFs and manages them. SONAr retrieves the information about the current VNFs and applies static routes between V_1 and VNF_1 , as well as between V_1 and VNF_2 , as shown in (b). If a degradation or failure happens in the link NE_1 - NE_4 , SONAr applies an alternative route, avoiding the degraded/broken link as shown in (c). Adapted from [20].

of initial paths between the components in the catalogue. After such functions, the initial paths are applied in the NEs, as shown in Fig. 4b.

We assume that the failures in network and infrastructure resources, managed by NFV components, are solved by such components. For example, failures in VNF_1 and VNF_2 are normally solved by V_1 . Common failures include hardware and software problems, which demand re-instantiating the VNFs; an overload of a virtual function, demanding the increasing of VNFs; and problems inside the VNFM itself. SONAr must perform algorithms for ensuring the communication between V_1 and the VNFs, and to solve problems inside V_1 . Analogous to the control path use case, if a problem occurs in a link utilised by the management paths, e.g. link NE_1 - NE_4 in Fig. 4b, SONAr must apply an alternative path, as shown in 4c. The initial path calculated and applied by v microservices was V_1 - NE_4 - NE_1 - VNF_1 . After a detected/predicted degradation in the link NE_1 - NE_4 , the alternative path applied was V_1 - NE_4 - NE_3 - NE_1 - VNF_1 .

4.3 SDN Controller Migration Use Case

Considering a cloud environment, with SDN and NFV components running over VMs or containers, it is reasonable that such components can modify the virtual locations. The migration of these components is expected in SONAr since it is necessary for recovery from some failures. In Fig. 5a, the same network topology used in the previous detailed use cases is shown. Assuming that compute nodes are available and physically connected with all NEs, the initial SDNC placement is in NE_1 . The control traffic between C_1 with any NE obligatorily passes throughout NE_1 .

The SONAr can operate to solve some failures that can occur after some specific impairments associated with the SDNC placement. The first one is associated with the issues that occur in the NEs directly associated with the SDNC, e.g. NE_1 in Fig. 5a. An overload or crash in NE_1 can culminate in a non-operational

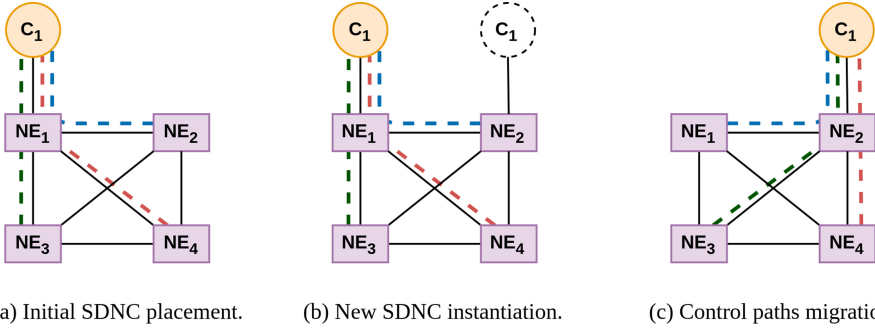


Fig. 5. Migration of an SDN controller use case. SONAr can determine the best SDN controller placement based on failure identification or network traffic. If a controller is initially plugged in a NE, e.g. NE₁ as shown in (a), but the highest traffic of SBI control primitives is in another part of the topology, e.g. NE₂, SONAr migrates the controller, which is usually running on a VM, to another NE, as shown in (c).

data plane. As SONAr has procedures, in the SHE and SLEs, for failure detection or prediction respectively such impairments, recovery actions are possible if the C_1 runs over a VM or container. Assuming that NE₂ has a compute node directly connected with it, SONAr can instantiate a new VM and start C_1 inside this VM, as shown in Fig. 5b. For this, SONAr works as follows: CoEs collected information accusing the failure; SHE recovery microservice analyses the information and creates new events containing the solution, i.e. commands to instantiate a new controller instance; the new events reach the NSB, which transforms such events in the necessary commands; NSB requests the creation of a new VM to a VIM running in the environment; and finally, the NSB copies C_1 settings to the new VM, then it starts the applications, and finally it copies the current state from C_1 . At the final, the new VM has the same configuration of C_1 running in the old VM. For avoiding loss of information, the ideal procedure is to maintain C_1 state in a structure shared by the old VM and the new VM until the final of the migration.

At the moment that the new C_1 VM is ready to assume the traffic, SONAr configures every NEs with flow rules to send the control traffic to the new VM, i.e. that one plugged on NE₂. This step is represented in Fig. 5c. The final step from SONAr is to turn down the old VM. The described situation shows the failure in the NEs where the C_1 was placed. However, other situations demand real-time controller migration. One of them is the high-traffic in some part of the network topology. As an example, the topology shown in Fig. 5 has the C_1 connected with the NE₁ at the beginning; however, if the control traffic is higher in NE₂, the controller can be migrated to NE₂, avoiding control traffic flooding in the network. SONAr is capable to work with this situation because SLEs have algorithms for dealing with metrics indicating the heat map of the topology, i.e. the part of the topology graph where the control/management traffic is higher. Besides that, Fig. 5 shows a basic four-node topology. In different topologies,

there is the option to allocate the SDNC next to the NEs that demands more control traffic.

4.4 SDN Controller Instantiating Use Case

The SDNC Instantiating use case is similar to the migration use case. However, in this new use case, both SDNCs operate at the same time. Usually, an SDNC controls a segment inside a domain. In Fig. 6a, the C_1 controls a domain with four NEs in the topology, meaning that the domain has just one segment. This basic four-nodes topology will probably not have overload or performance problems, but it is enough to illustrate this use case. In this particular topology, NE_1 , NE_2 , NE_3 , and NE_4 send control primitives to C_1 logical address, e.g. IPv4 or IPv6. For exemplifying, NE_3 will send all its control primitives to NE_1 , which will send such primitives to C_1 . When C_1 needs to send a request/response control primitive to NE_3 , it will send the primitive to NE_1 ; NE_1 has flow rules configured to forward primitives with destination ‘ NE_3 ’ to NE_3 .

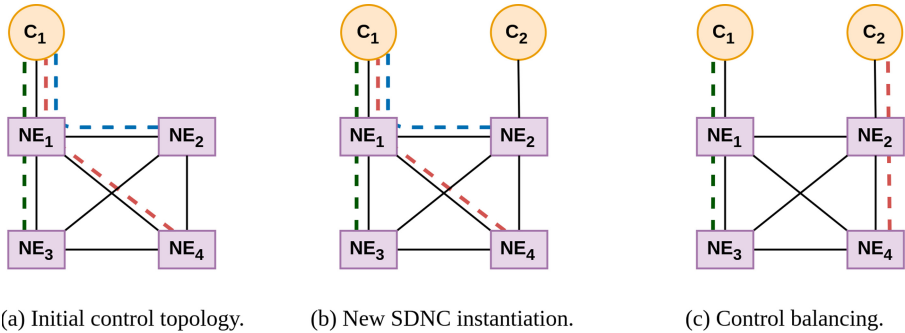


Fig. 6. Control traffic balancing use case. Some current SDN controllers allow the distribution of controller instances. In this use case, SONAr identifies a high-traffic of control primitives to a unique instance as shown in (a). Next, SONAr instantiates a new controller instance, as shown in (b), and then migrates the control paths, balancing the control traffic as shown in (c).

Mainly for performance reasons, SONAr can assume that traffic balancing is necessary to maintain the high performance of the control primitives traffic. In this situation, it is necessary to instantiate a new SDNC instance and split the traffic. For this, the first step is to create a new instance as shown in Fig. 6b. Notice that, differently from the migration use case – Fig. 5b shows only one active C_1 node at a moment – in this use case the new instantiated SDNC is an additional instance as shown in Fig. 6b: there are two SDNC active instances, i.e. C_1 and C_2 . Once the new instance is created and configured, SONAr integrates C_1 and C_2 in a way that they start to share the same state. It is crucial to mention that the procedures described in this section just work with SDNC prepared to work with a horizontal scale-up, e.g. the ONOS controller [12].

Since C_1 and C_2 are operational as shown in Fig. 6b, SONAr starts the migration of NEs to the new instance. SONAr defines the NEs that will be migrated to C_2 and modifies the flow rules inside such NEs. In the example shown in Fig. 6c, SONAr kept NE_1 and NE_2 logically connected with C_1 ; and modified NE_3 and NE_4 to send control primitives to C_2 . The balancing algorithms performed by SONAr uses basic equations of traffic distribution to determine the number of new instances to be created and the distribution of NEs to these new instances. If more than one SDNC instance is available in the topology, the balancing use case is possible as detailed in this section. Additionally, it means that the use case related in Subsect. 4.3 is optimised, as the steps to instantiate new instances are not necessary.

5 Concluding Remarks and Future Work

In this paper, we introduced a new framework to apply self-healing functions on SDN and NFV environments. The framework is based on SONAr, which is a reference architecture for computer/mobile networks self-management. Our framework runs on the Management Layer and has procedures for autonomously monitoring the other layers. The difference between SONAr and other projects in literature is in the system placement: SONAr places at the Management Layer and it is entirely application-based. The common computer and telecommunications protocols are developed at SONAr, and its modular characteristic enables the adoption of SONAr even in environments in which SDN, NFV, or legacy NEs are already deployed.

We exploited the self-healing autonomic computing fundamental for illustrating SONAr's capabilities. There is a thorough difference between a fault tolerance system and a self-healing system. The first is a system capable to deal with faults, i.e. the system tolerates failures, which means that the system will continue to work because the components have backups or are partially distributed. The former means that the system has the notion that some of its parts are degraded or broken, and therefore it can self-recover itself. Considering a computer or mobile network, the entire network is the system, i.e. the graph representing the topology with nodes (NEs) and edges (links). In an SDN and NFV environment, the system is the infrastructure topology and the control/management layers. In this way, an SDN/NFV environment achieves the self-healing characteristic by considering all layers, i.e. Control, Management, and Infrastructure layers.

For implementing all self-healing functions for current and future applications, we decided to build a framework with an extensible flavour. Hence, the basic self-management procedures are built at the beginning, but new procedures can be introduced in the future without modification in the initial ones. For demonstrating this, we choose four use cases focused on control plane connectivity: (i) the first use case shows an SDN topology with degraded/broken links, and a real-time or preventive migration of control paths to avoid such links; (ii) the second use case shows the same scenario, but considering an NFV topology, in which static routes are necessary to ensure the connectivity between NFV

components and their managed resources; (iii) the third use case illustrates a live migration of an SDN controller; and finally, (iv) the fourth use case illustrates an instantiating of a new SDN controller and the control traffic balancing.

The aforementioned use cases have proven the efficiency of SONAr for dealing with common problems that are found in control and management planes. For the best of our knowledge, SONAr is the first work to allow the implementation of these procedures in a flexible model, enabling diverse use cases. At this moment, we have the implementation of the basic SONAr components, which means that new researchers from our group can start the implementation of different microservices for the healing of mobile/computer networks. The self-healing was chosen in this paper as a study case, but the platform is prepared for self-configuration, self-optimisation, self-protection, and so on. As future work, we long for presenting the performance experiments of SONAr considering the four use cases just presented, as well as new self-* use cases.

References

1. 3GPP: Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements. Technical Specification (TS) 32.500, 3rd Generation Partnership Project (3GPP) (2018). <http://www.3gpp.org/-DynaReport/-32500.htm>
2. 3GPP: Telecommunication management; Study on management and orchestration of network slicing for next generation network. Technical Report (TR) 28.801, 3rd Generation Partnership Project (3GPP), January 2018. <http://www.3gpp.org/DynaReport/28801.htm>, version 15.1.0
3. Abdelsalam, M.A.: Network Application Design Challenges and Solutions in SDN. Ph.D. thesis, Carleton University (2018)
4. Afolabi, I., Taleb, T., Samdanis, K., Ksentini, A., Flinck, H.: Network slicing and softwarization: a survey on principles, enabling technologies, and solutions. *IEEE Commun. Surv. Tutor.* **20**(3), 2429–2453 (2018). <https://doi.org/10.1109/COMST.2018.2815638>
5. Basta, A., Blenk, A., Belhaj Hassine, H., Kellerer, W.: Towards a dynamic SDN virtualization layer: Control path migration protocol. In: 2015 11th International Conference on Network and Service Management (CNSM), pp. 354–359, November 2015. <https://doi.org/10.1109/CNSM.2015.7367382>
6. Canini, M., Salem, I., Schiff, L., Schiller, E.M., Schmid, S.: A self-organizing distributed and in-band SDN control plane. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 2656–2657, June 2017. <https://doi.org/10.1109/ICDCS.2017.328>
7. Chen, H., Abbas, R., Cheng, P., Shirvanimoghaddam, M., Hardjawana, W., Bao, W., Li, Y., Vucetic, B.: Ultra-reliable low latency cellular networks: Use cases, challenges and approaches. *IEEE Commun. Mag.* **56**(12), 119–125 (2018). <https://doi.org/10.1109/MCOM.2018.1701178>
8. Cox, J.H., et al.: Advancing software-defined networks: a survey. *IEEE Access* **5**, 25487–25526 (2017). <https://doi.org/10.1109/ACCESS.2017.2762291>
9. Ganek, A.G., Corbi, T.A.: The dawning of the autonomic computing era. *IBM Syst. J.* **42**(1), 5–18 (2003)

10. GonÇalves., M.A., de Souza Neto., N.V., Oliveira., D.R.C., de Oliveira Silva., F., Rosa., P.F.: Bootstrapping and plug-and-play operations on software defined networks: a case study on self-configuration using the sonar architecture. In: Proceedings of the 10th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, pp. 103–114. INSTICC, SciTePress (2020). <https://doi.org/10.5220/0009406901030114>
11. Kubernetes: Kubernetes (2020). <https://kubernetes.io/>
12. ONOS: ONOS (2020). <https://wiki.onosproject.org/>
13. Open Networking Foundation: OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06) (2020). <https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf>
14. OpenDaylight: OpenDaylight (2020). <https://www.opendaylight.org/>
15. Openstack: Openstack (2020). <https://www.openstack.org/>
16. Fonseca, P.C., Mota, E.S.: A survey on fault management in software-defined networks. *IEEE Commun. Surv. Tutor.* **19**(4), 2284–2321 (2017). <https://doi.org/10.1109/COMST.2017.2719862>
17. Rehman, A.U., Aguiar, R.L., Barraca, J.P.: Fault-tolerance in the scope of software-defined networking (SDN). *IEEE Access* **7**, 124474–124490 (2019). <https://doi.org/10.1109/ACCESS.2019.2939115>
18. Sanchez, J., Yahia, I.G.B., Crespi, N., Rasheed, T., Siracusa, D.: Softwarized 5G networks resiliency with self-healing. In: 1st International Conference on 5G for Ubiquitous Connectivity, pp. 229–233, November 2014. <https://doi.org/10.4108/icst.5gu.2014.258123>
19. Schiff, L., Schmid, S., Canini, M.: Ground control to major faults: towards a fault tolerant and adaptive SDN control network. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), pp. 90–96, June 2016. <https://doi.org/10.1109/DSN-W.2016.48>
20. de Souza Neto., N.V., Oliveira., D.R.C., GonÇalves., M.A., de Oliveira Silva., F., Rosa., P.F.: A self-healing platform for the control and management planes communication in softwarized and virtualized networks. In: Proceedings of the 10th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, pp. 415–422. INSTICC, SciTePress (2020). <https://doi.org/10.5220/0009465204150422>
21. Thorat, P., Raza, S.M., Nguyen, D.T., Im, G., Choo, H., Kim, D.S.: Optimized self-healing framework for software defined networks. In: Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication., pp. 1–6 (2015)
22. Yousaf, F.Z., Bredel, M., Schaller, S., Schneider, F.: NFV and SDN–key technology enablers for 5G networks. *IEEE J. Sel. Areas Commun.* **35**(11), 2468–2478 (2017). <https://doi.org/10.1109/JSAC.2017.2760418>
23. Zhou, X., Li, R., Chen, T., Zhang, H.: Network slicing as a service: enabling enterprises’ own software-defined cellular networks. *IEEE Commun. Mag.* **54**(7), 146–153 (2016). <https://doi.org/10.1109/MCOM.2016.7509393>