A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions

Sihem Cherrared, Sofiane Imadali, Eric Fabre, Gregor Gössler, Imen Grida Ben Yahia

A Survey of Fault Management in Network Virtualization Environments: Challenges and Solutions

Sihem Cherrared[1,2], Sofiane Imadali[1], Eric Fabre[2], Gregor Gössler[3], and Imen Grida Ben Yahia[1]

[1]Orange Lab Networks, 92320 Chatillon, France
[2]INRIA, Campus de Beaulieu. F-35042 Rennes, France
[3]Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

Abstract

The advent of 5G and the ever increasing stringent requirements in terms of bandwidth, latency, and quality of service pushes the boundaries of what is feasible with legacy Mobile Network Operators' technologies. Network Functions Virtualization (NFV) is a promising attempt at solving some of these challenges that is widely adopted by the industry and specified by the standardization bodies. In essence, NFV is about running Network Functions (NFs) as virtualized workloads on commodity hardware. This may optimize deployment costs and simplify the lifecycle management of NFs, but it introduces new fault management challenges and issues. In this paper, we propose a comprehensive state of the art of fault management techniques. We address the impact of virtualization in fault management. We propose a new classification of the recent fault management research achievements in the network virtualization environments and compare their major contributions and shortcomings.

Keywords: Fault management, Network Functions Virtualization, Software Defined Network, Machine learning.

1 Introduction

With the advent of 5G, Mobile Network Operators (MNOs) leverage a growing range of proprietary hardware appliances and complex legacy communication protocols and architectures that need to transition to a software model [1]. New network services require the deployment of new hardware servers, maintaining them and eventually replacing the malfunctioning ones to keep the services running correctly. This procedure associated with the NFs lifecycle, has a high cost and is energy inefficient. Another underestimated part of the MNOs work is the time spent in different standardization bodies, like *ITU-T*, *ETSI*, *IETF*, or *NGMN*, to assure compliance among the proprietary hardware devices and protocols.

However, as a fast growing sector, the telecommunication industry is facing increased competition as new players are entering with emerging software technologies and open source projects. One such project is the recent *Telecom Infra Project* [2], initiated by Facebook to create an open source general purpose hardware for a new generation of MNOs. This approach to infrastructure is radically different from what the telecommunication industry is used to and leads MNOs to invest in and adopt new technologies: Software Defined Networking (SDN), Network Functions Virtualization (NFV), cloud infrastructure and live monitoring technologies.

Software-based innovative technologies help with designing the upcoming 5G networks [3]. 5GPP projects such as *Novel Radio service adaptive network Architecture (NORMA)* [4], advocate for API-driven architectural openness, fueling economic growth through over-the-top innovation.

These projects propose adaptive decomposition and allocation of mobile NFs, which flexibly decomposes the mobile NFs and places the resulting functions in the most appropriate location. This comes down to defining a service-oriented architecture for the 5G networks, which is disruptive when compared to how the Long Term Evolution (LTE) network has been designed and implemented.

With the softwarization of 5G, MNOs are seeking new revenue sources and models. NGMN [5], proposes to break the traditional business model of a single network infrastructure ownership to introduce network sharing or multi-tenancy. This approach has the potential to recover up to 35% of the Radio Access Network (RAN) costs [6].

Network Virtualization Environment (NVE) enables the coexistence between multi-tenant SDN and NFV architectures in a single infrastructure without affecting production services [7]. However, to ensure the high availability and reliability of these services, one paramount aspect that MNOs cannot neglect is the management of NVEs. The management procedures are summarized as FCAPS: Fault, Configuration, Accounting, Performance, and Security management. The NVE brings new challenges that highlight the need to rethink the management procedures and adapt them to these environments. Fault management represents one of the most important X-management axes since a failure can cause important losses and impact the business image and credibility. One example of recent cloud failures has been registered in 2013 within IBM. The monetary loss was estimated to 31 million Australian dollars and it was due to IBM server breakdown that brought down the website of a major Australian retailers for one week during the Christmas period [8]. The flexibility of NVE introduces a new way to do network monitoring. The supervision should be done before building the 5G architecture by adding resiliency to the architecture and self-healing actions making use of the dynamism and reproducibility of NVEs. However, even though the supervision of networks is a well- known field with many deployed techniques often involving the Simple Network Management Protocol (SNMP) and software tools such as Nagios [9], virtual networks bring new challenges and issues that change the way failure management is performed.

In this paper, we discuss virtualization impacts on classic fault management techniques. Our work aims primarily at complementing the existing surveys discussed in the related work. In short, our main contributions can be summarized as follows:

- We provide a comprehensive survey of the state-of-the-art fault management techniques and the impact of the virtualization of network fault management.
- We present the recent research achievements and use cases in NVE failure management.
- We propose a new classification of the current efforts that address fault localization challenges, and compare their major contributions and shortcomings.

The remainder of this paper is organized as follows. Section 2 discusses existing survey papers in the management of NVE and presents the scope of our survey. Section 3 depicts the NVEs and fault management steps and methods. Section 4 presents NVE issues, while Section 5 presents novel efforts for the fault management of NVE. Section 6 presents a classification of the different efforts and how best to apply them in different contexts. Section 7 concludes the paper.

2 Related work

Recent survey papers explore fault management efforts in virtual networks but diverge on the type of the addressed virtual environment and management aspect. *Esteves et al.* [7], provide a comprehensive view on the current advances in the management of NVE. The paper first introduces a conceptual management model for NVE describing the entities, relationships and management operations. The paper then reviews representative projects and highlights their main features, benefits, and limitations and proceeds to classify them according to three different perspectives: management targets, functions, and approaches. Survey papers [10–14], focus on the SDN management. *Fonseca et al.* [10], addressed the fault tolerance issues in each SDN layer (i.e., infrastructure,

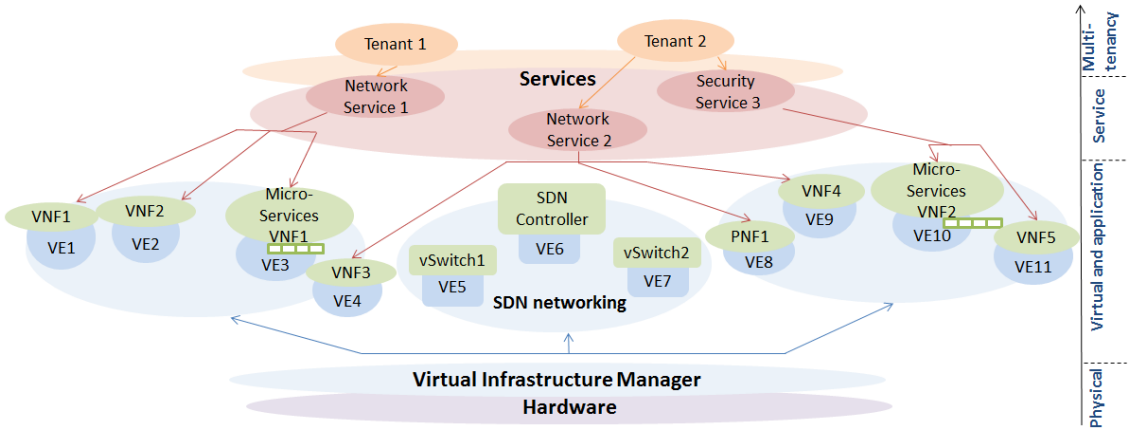


Figure 1: Network Virtualization Environment.

control, and application). They investigated the possible sources of faults that might arise from each layer but also from the interaction between layers. *da Silva et al.* [11], in their SDN survey target the resilience of networks.

The authors have a similar conclusion to *Fonseca et al.* in [10], with a set of major challenges for each resilience discipline. The key difference is that *Fonseca et al.* [10], addressed the challenges according to the application, control and data SDN planes.

Chen et al., first review the fault tolerance on servers and links for traditional networks. They then address the fault detection and recovery mechanisms for the SDN data and control plane based on the OpenFlow protocol. *Hohlfeld et al.* [14], discussed the scalability issue in the three SDN layers and the research contributions in this issue. *Foerster et al.* [15], surveyed the mechanism and protocols for fast and consistent network updates for traditional networks and SDN. *Gonzalez et al.* [13], targeted the dependability of the NFV orchestrator including its resilience against faults.

Recently, many survey papers [16–19], have addressed the application of Artificial Intelligence (AI) and Machine Learning (ML) in network environments. *Boutaba et al.* [16], provide a comprehensive view of the application of ML in networking. The surveyed papers addressed in this survey discuss different types of networks such as cellular networks and wireless sensor networks (WSN), and management aspects such as traffic classification and fault management. The fault management efforts were classified with regards of their network types, ML techniques, data set, features, outputs and evaluation. Papers [17–19], present a literature review of ML methods that were used to address common issues in WSNs, self-organizing networks (SON) and SDN respectively. Fault management efforts were briefly introduced in each of them.

Our work aims to complement the existing surveys and describes new approaches including the latest achievements in fault localization and recovery. We discuss the new features and challenges that come from combining SDN/NFV in a multi-tenant environment. Then we categorize and compare failure management efforts and open source projects in NVE according to the fault management steps, the applied methods and their added values to face the NVE issues. Finally, we discuss their contributions and shortcomings.

3 Background

NVE enables several architectures to coexist on a single infrastructure. Fig. 1 represents a high-level overview of such an environment. It illustrates the vertical and horizontal view of the related entities in a composed per-tenant 5G service. This section describes the underlying SDN and NFV concepts, the coexistence of both architectures, and the composition of the infrastructure that supports NFV. We also provide a definition of the fault management steps, the relevant ML

and AI techniques, and their application in the fault localization process.

3.1 Network virtualization environments

We describe SDN and NFV, highlighting their relationship.

3.1.1 Software Defined Networks

SDN is an emerging network architecture that separates the network control, management, and forwarding functions, enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [20]. In the SDN architecture an external controller decides the control plane functions that include the system configuration, management, and exchange of routing table information. Due to this separation, network switches become simple forwarding devices and the control logic is implemented in the controller. This simplifies policy enforcement and network reconfiguration using the OpenFlow protocol, for example. It enables the controller to directly interact with the forwarding plane of network devices such as switches and routers. SDN is divided into three layers: the infrastructure, control, and application layer. The infrastructure layer also known as data plane represents the layer holding the forwarding devices i.e., physical and virtual switches). The application layer hosts the NFs and communicates with the SDN control plane through the northbound Application Programming Interfaces (APIs) regarding the status of the network and its particular requirements. The controllers, situated in the control plane, give instructions to the data forwarding devices through the southbound APIs. Recently, open source communities proposed various SDN controllers with varying degrees of maturity and functionality: POX, Beacon, Ryu, Trema, OpenContrail, and ONOS are well known examples [21].

3.1.2 Network Functions Virtualization

NFV is the concept of transforming pure hardware appliances hosting NFs (e.g. NAT, firewall, intrusion detection and DNS) into software functions hosted on commodity hardware servers. These functions decoupled from the underlying hardware are known as Virtual Network Functions (VNFs). ETSI-NFV presented a set of NFV use cases in mobile networks in white papers [22]. One interesting use case is the VNF forwarding graph (VNF-FG) that allows the deployment of end-to-end 5G services. The notion of VNF-FG is defined as a sequence of VNFs interconnected to provide a network service with specific functionalities [23]. Authors of [24], summarize representative industrial products and solutions of VNFs. For instance, the Clearwater project by Metaswitch, is a Virtual IP Multimedia Subsystem (vIMS) [25].

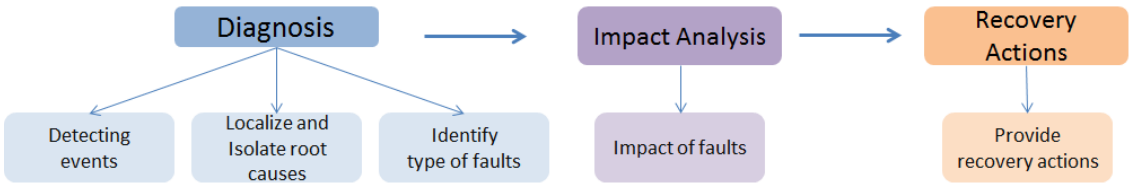


Figure 2: The Fault Management Process.

3.1.3 NFV and SDN coexistence

SDN and NFV are independent concepts and arguably complementary. In essence, NFs can be virtualized and deployed without SDN technologies, and non-virtualized functions can be controlled by SDN. Meanwhile, both models can be complementary and mutually beneficial technologies; NFV can support SDN by providing the infrastructure upon which the SDN software can be

run. For instance, we can consider the software of the SDN controller as a VNF. Furthermore, SDN infrastructures can be used for the data forwarding between VNFs. From the Open Networking Foundation (ONF) viewpoint, a VNF for an SDN controller is just another resource, a node function in a network graph with known connectivity points and known controllable transfer function [26].

Combining NFV with SDN in one infrastructure makes sense from a network operator's standpoint, to reduce the costs of NF deployment and management. Moreover, the SDN scalability and elasticity allows for a dynamic deployment of NFV that suits the on-demand NFV chain placement and the communications requirements for both virtual and physical networking infrastructures [26].

3.1.4 Multi-tenancy

To make a better use of virtualization, operators may share their non-occupied resources with customers, who become tenants of the same infrastructure. The definition of multi-tenancy differs between vendors and entities. Generally speaking, it means that multiple tenants or clients are sharing the same virtual compute, storage and network resources [27]. For instance, in Fig. 1 two tenants are sharing resources in the same infrastructure. Multi-tenancy is enabled by the notion of slicing, which allows the traffic of multiple tenants to be partitioned through the same infrastructure, with tunnels established to forward traffic within a given slice [28].

3.2 Infrastructure and running environments

The ETSI-NFV industry specification group decomposes the NFV architecture into major components including VNFs, NFV management and orchestration (MANO), and Network Functions Virtualization Infrastructure (NFVI) on top of the traditional network components like Operations and Business Support System (OSS/BSS) [29]. The Virtual Infrastructure Managers (VIMs) represent an important part of the NFV-MANO architecture and are critical to realize the business benefits enabled by the NFV architecture. The VIM coordinates physical resources to deliver network services. In the NFV market, many vendors propose VIM solutions: Kubernetes, VMware, Openstack and OpenVIM for instance. Virtual Machines (VMs), containers and unikernels provide virtual compute and storage resources crucial to VNFs. Containers emerged as a way of running applications in a more flexible and agile way. Containers enable the running of lightweight applications directly within Linux OS, whereas each VM runs an independent OS. Unikernels are a lightweight alternative that package the VNFs with required libraries; unlike VMs that provide an entire guest OS. Unikernels use a *"library operating system"* implementing kernel features compiled with the unikernel image code. This makes the sizes of unikernel images similar to those of containers. However, this restriction does not allow for system diagnosis tools (like ICMP Ping) to be embedded. VM and containers enable novel ways of virtualization discussed in [24]. One way is running containers in VMs. Another approach, called *"clear containers"*, is about running containers in a VM hosting a lightweight OS based on Clear Linux with booting times closer to those of a container.

3.3 Fault management

Fault management (FM) is the process of locating, analyzing, fixing and reporting network problems such as link failures and network overload. Fig. 2 depicts the FM steps from detection to healing. *Fault diagnosis* aims at achieving three complementary tasks: fault detection, localization and identification [30]. In the following subsections, the different steps of fault management and efforts applied to legacy telecommunication networks are depicted.

3.3.1 Fault detection

Fault detection is at the same time an elementary and difficult task. This step determines whether the system works in normal conditions or a fault has occurred. A fault is the *root cause* that may

lead the system to an error state. A failure occurs when an error causes a *malfunctioning* of network devices or software leading to symptoms or alarms. Normal and faulty behaviour may depend on the Service Level Agreement (SLA) and the performance requirements of each network service provider. Faults can be classified into different types depending on their behaviour and cause. Faults can be permanent, transient for a short period or repetitive. They can originate from maintenance, configuration errors, or malicious intrusion [15]. Moreover, a fault may lead a system to a complete crash or only degradation. Table 2 illustrates a number of faults in NVE that will be further discussed in Section 4.2 and 5.2. Network administrators use two kinds of data to determine the state of the network: *alarms* and *metrics*.

Alarms Alarms (or symptoms) are external manifestations of failures. These notifications may originate from management agents like SNMP traps [31], or in the form of system log files generated by the syslog protocol (or other protocols) [32]. Syslog allows devices to send event notification messages over networks to any predefined collector [32]. The information carried within syslog messages may include: the IP address of the object that generated the alarm, a timestamp, an alarm identifier, a measure of severity of the failure condition and an additional textual description of the failure. Syslog messages are generally stored in logs. These messages contain important information about the health and operation of the system, some are just informative and others present urgent notifications. However, even if alarms provide precious clues about the root cause and the type of fault, some faults may be partially observable or unobservable. Log management in complex networks raises several issues:

- A single failure may generate multiple alarms due to the connectivity and dependency of devices;
- False positives in the case of alarms generated during reconfiguration or maintenance operations or caused by the network device state that takes a long time to stabilize;
- Loss, delay, or different time formats of alarms that do not always respect the *ISO8601* standard [33];
- The problem of the clock synchronization in distributed systems that affects the order of the received alarms.

Since most faults are not directly observable, the management system has to infer their existence from information provided by the received alarms, hence the necessity of fault correlation and inference techniques.

Metrics Another way to detect a faulty behaviour is by collecting network performance metrics. Metrics are a quantitative and qualitative way to verify desired aptitudes and to measure degradation. Network metrics include: delay, jitter, throughput or network utilization. For instance, jitter is the inter-packet delay variation, and network utilization is a measure of how much of the capacity is currently in use. In the fault detection process, system level metrics are continuously collected and compared to an acceptable quality level. If the metric measures degradation or violation of SLA a notification is raised.

3.3.2 Fault localization and identification

After a faulty state has been detected due to an alarm notification or degradation in the system performance, much ambiguity remains to be resolved. As the detection process provides only few indications, fault evidence may be inconclusive, inconsistent and incomplete [34]. Fault localization (also named fault isolation or Root Cause Analysis (RCA)) represents the procedure of deducing the exact root cause of a failure from a set of alarms, notifications and indications. Fault localization addresses several challenges and issues caused by the ambiguity and inconsistency of

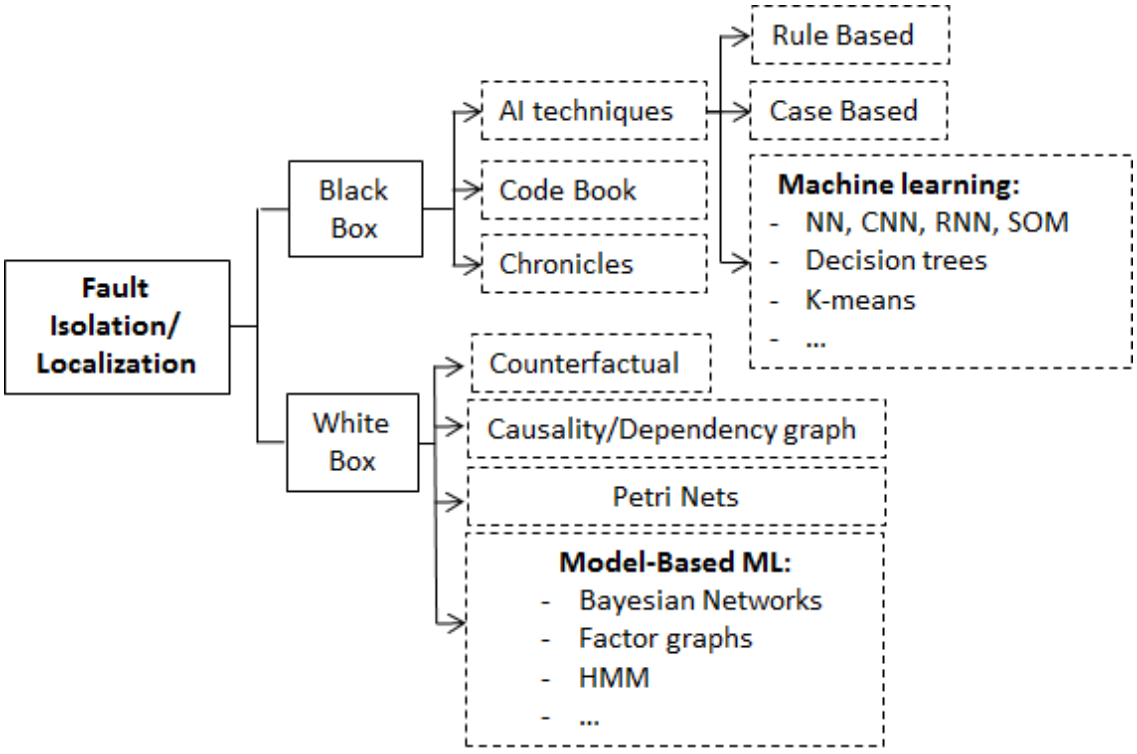


Figure 3: Fault localization approaches.

alarms and errors propagation [34]. Moreover, fault localization should provide the human operator with clear explanations about the root cause and type of faults and should take into account the network topology, the running services and the configuration and maintenance operations.

In Fig. 3, we provide a new classification of network fault localization techniques. The proposed classification complements the methods presented in survey [34]. Fault localization techniques are classified into white and black-box techniques. Fig. 3 depicts the fault localization methods that have proven to be relevant for RCA of networks. As a definition, white-box techniques use an explicit model of the network and its topology. Whereas, in black-box techniques an implicit representation is constructed. White-box approaches give an interpretation of a failure and the propagation of faults by modeling the relationships between nodes, events, alarms and faults. Black-box methods learn the model from data, but the learned model is often difficult to understand and explain. AI and particularly the ML branch is getting a significant amount of attention in the world of telecommunications since it allows automatic management. ML uses mathematical models trained on data to make decisions without being explicitly programmed to perform the tasks [35]. ML methods address four types of problems: clustering, classification, regression and rule extraction [16]. Classification aims to match a set of novel input data to a set of discrete output values. In regression the outputs are of continuous type. In clustering problems the goal is to gather resembling data in one group, whereas rule extraction derives symbolic rules from data. Fig. 4 presents the two different ML methodologies classified as black-box and white-box methods in the localization techniques (cf. Fig. 3). Model-Based MLs (MBMLs) such as Bayesian Networks (BNs) seek to construct an explicit diagnosis model for the inference process, whereas black-box ML methods (e.g. Neural Networks (NNs)) learn the model from a training and validation data. The data could be totally, half or not labelled, for supervised, semi-supervised and unsupervised learning, respectively.

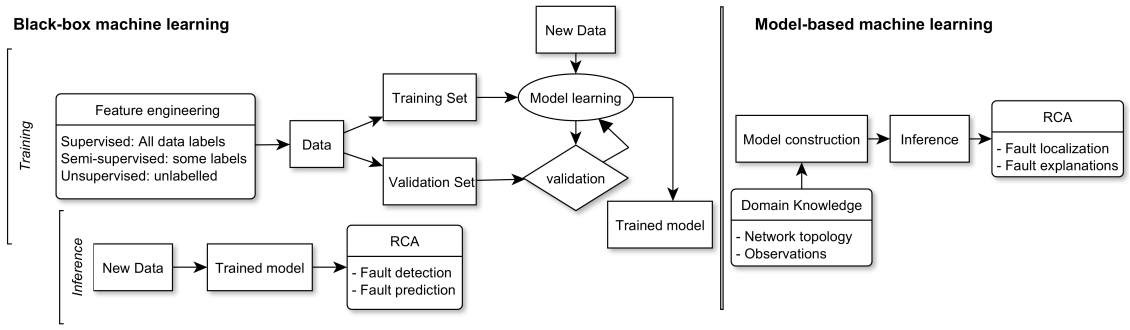


Figure 4: Black-box vs Model-based ML methods.

White-box techniques White-box methods have two main structures: the domain model and problem solving methods [36]. The domain model is the structure of the connected components and alarms propagation, while the solving methods are event management algorithms used to identify the root cause in the model. Nevertheless, the model-based approach has two weaknesses that are the definition of the model and their ability to deal with the large scale networks. The model-based causality or dependency graphs (cf. Fig. 3) represent the relationship between failures, alarms, intermediates faults and the failures generated. The edges between the nodes represent a causality relationship. It can be a "must cause" or a "may cause" [37]. The network topology and fault propagation knowledge comes from administrators' expertise based on hardware and software specifications. Many efforts [38–40], used dependency graphs for fault localization. Petri nets are a basic model of concurrent and distributed systems that describe state changes in a system with transitions and places. The transitions depend on the availability of input data. Petri nets are used to represent faults propagation in networks, where the places are alarm patterns that are generated by devices, and the transitions are events that change the network state. This approach was used in the Synchronous Data Hierarchy (SDH) network in [41]. The authors of [42], used Petri nets to detect network SLA violation. In [42], each alarm had its associated Petri net represented as action plans to find the root cause.

More recent RCA approaches [37, 43], use BNs. A BN represents a probabilistic extension of causality and dependency graphs. A BN is a directed acyclic graph (DAG) that represents cause and effect relationships between observable and unobservable events such that when a set of symptoms is observed, the most likely causes can be determined [44]. The nodes in the DAG are random variables (e.g. network elements, events and faults), while the edges denote the existence of direct causality between the connected variables. The strength of the causality is expressed with conditional probabilities. To construct a BN model a deep human expert knowledge of the cause and effect relationships in the domain is required. This allows humanly understandable RCA explanations compared to black-box techniques [37].

Authors of [37, 43] proposed self-diagnosis approaches for telecommunication networks based on BNs. Authors of [37] proposed a self-modeling and diagnostic engine based on the formalism of generic BNs for fault localization in IP Multimedia Subsystem (IMS). IMS has the specifications of a large network with multiple layers and segments. The defined model describes dependencies between resources used by an IMS service. When a failure occurs in the IMS service, the algorithm locates the on-line instances of that model in a given network topology and generates the corresponding BN instances. Authors of [43], proposed a three-layer self-reconfigurable probabilistic model for diagnosis in Fiber To The Home (FTTH) networks. The model integrates two fields, a decision and an artificial learning field, the decision field is based on Bayesian reasoning and the artificial learning field brings self-reconfiguration capabilities.

Another interesting example of MBML is multiplicative factor or constraint graphs. Constraint graphs are defined as a set of variables, a domain value for each variable and a set of constraints [45]. Constraint graphs could be represented as a cluster graph with two types of nodes in the graph,

variable nodes and function nodes, representing the constraints between the linked variables. A node is a number of variables linked with constraints. Two nodes are linked if they share the same variables. The RCA algorithm applied in factor graphs is called the Cluster Tree Elimination (CTE) algorithm. This algorithm unifies a variety of previously known algorithms: forward-backward algorithm, sum-product algorithm, Viterbi algorithm and belief propagation. Authors of [46], proposed to deploy factor graph based algorithm to monitor link loss rates in WSNs. The idea was to apply the data aggregation communication paradigm of WSNs to implement a link loss monitoring algorithm that infers link loss rates by observing the arriving packets.

Another way of reasoning in fault diagnosis is counterfactual reasoning [47–50], which takes the form of:

"If A were true, then B would have been true?"

where A and B are events or symptoms. A is the counterfactual antecedent to the consequent B. A is an event that is contrary to the real observations, while B is the result that is expected in the alternative world where the antecedent is true [47]. Recent definitions of actual causality [48] and fault ascription [50] use counterfactuals in order to pinpoint the events or components responsible for the violation of a safety propriety, for instance, defined by VNF requirements.

Black-box techniques represent methods where only the entries and results are known. The learning process is often difficult to explain. Chronicles [51], one of the black-box techniques proposed to filter alarms or logs and look for known patterns. Each observable received event is time-stamped and classified in an event flow. The explanation of the fault is given by an expert when there is a match between a chronicle (i.e., a set of known events that describe a failure) and the event flow received. This technique allows learning from logs automatically. On the other hand, the codebook approaches [52], associate symptoms with problems in a matrix with binary values, each problem is represented by a vector of symptoms. The rule-based techniques [53, 54], describe a number of rules in the form of conditions: if <symptoms> then <root cause>. While the case-based techniques [55, 56], learn from past experience and past situations.

Recently, machine learning is the new term that refers to the techniques that adapt to new circumstances and detect and extrapolate patterns. NNs learn progressively by considering examples (i.e., input data). The learning system represents an association of neurons forming a more or less complex graph. An artificial neuron is an autonomous processing unit that receives one or more inputs; the inputs are weighted and summed with an activation or transfer function to produce an output. In the training mode, the neuron is trained with examples (i.e., inputs and outputs), until reaching a given precision, the trained neuron can then be used to provide outputs for real data (cf. Fig. 4). NNs, with their particular ability to derive meaning from raw data, can be used to extract patterns and detect failures in logs which represent a complex task to either administrators or other computer techniques. Out of other classes of NNs emerged the fault management methods including Convolutional Neural Networks (CNNs), Deep Neural Networks (DNNs), Probabilistic Neural Networks (PNNs), Recurrent Neural Networks (RNNs) and Self-Organizing Map (SOM). Further than NNs classes, ML includes a larger number of methods applied in network management: decision trees (e.g. Random forest), Support Vector Machines (SVM), Long Short-Term Memory (LSTM), K-means and K-nearest neighbors (K-NN) [16].

Papers [57–60], applied black-box ML methods for the fault localization. Authors of [57], used NNs with syslog messages. The syslog message is split into fields such as "Time" and "IP Address" and converted into numerical values to be used as inputs in the learning process. The outputs are then interpreted with decision-making rules.

Authors of [58], used RNN to detect faulty nodes in a WSN. RNNs are a class of NN where connections between neurons represent one cycle or more allowing internal feedback to their own inputs. The RNN model nodes correspond to the nodes of WSN topology. In addition, the output of the RNN is an estimation of the operation of the WSN that is compared with real WSN behaviors to achieve fault detection. Efforts [59] and [60], proposed K-means clustering solutions for the fault detection of IP networks and WSN respectively. K-means aims to partition

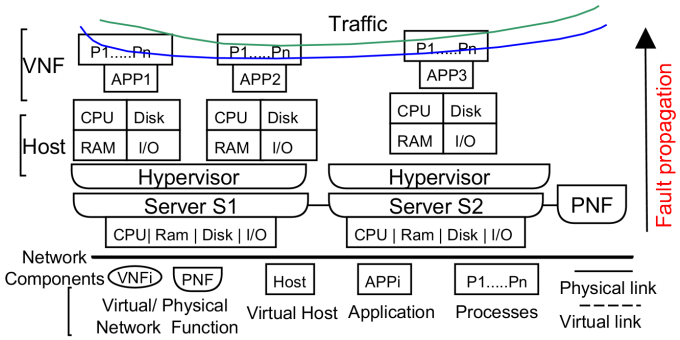
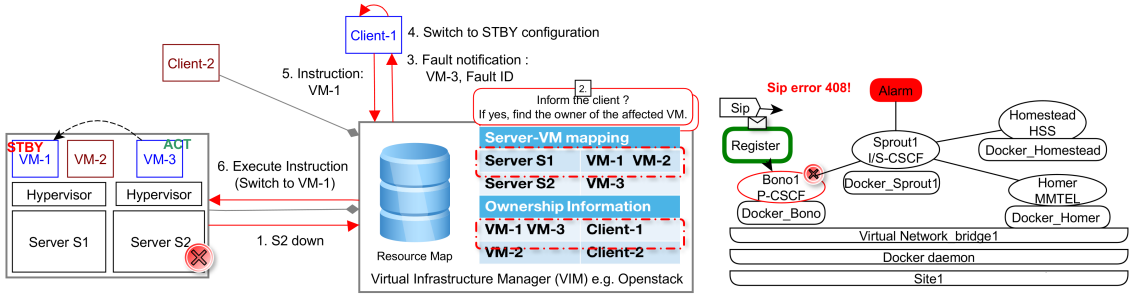


Figure 5: Virtual network Components



(a) OPNFV Server crash use case

(b) vIMS registration failure

Figure 6: Virtual Networks Fault Scenarios

n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

3.3.3 Impact analysis and recovery

Impact analysis is generally confused with fault isolation. However, finding the root cause is not always sufficient to perform recovery actions, knowing the affected components and the degree of network damage is also important. Operators need to prioritize severe failures for the troubleshooting based on their impact on end-user services [61]. Impact analysis is concerned with identifying the specific nature of faults (e.g. their size, propagation, importance, damage etc.). Moreover, the impact analysis helps administrators understand the risks of potential process or product failures. Several methodologies have been developed to quantify the effects and impacts of failures: graph-based [54]; Cause-Consequence Analysis [62]; Failure Modes and Effects Analysis (FMEA); Failure Modes, Effects and Criticality Analysis (FMECA) [63]. In traditional networks, most of the recovery actions were accomplished by the network experts since most of the failures that are due to hardware servers crash implies the administrators to repair or change the equipment. Still, some auto-recovery actions were supported in the network links recovery. The network paths recovery process handles two general failure recovery approaches: *protection* and *restoration* [64]. In the protection approach, the routes are reserved and computed before a failure occurs. On the other hand, restoration, the new link allocation occurs after the failures. Restoration may increase the recovery time since it is preceded reactively, while the protection may cause congestion due to links reservation.

4 NVE fault management challenges and issues

This section presents the NVE issues and discuss the limitation faced when using existing fault management techniques.

4.1 Issues

Table 1 lists SDN, NFV and multi-tenancy challenges. The issues summarized in Table 1 are addressed according to different fields from a fault management point of view. In the following a description of these issues is presented.

4.1.1 Scalability

Network scalability is the ability of a network to maintain its performance when traffic increases. The centralization of control in SDN raises scalability issues. In fact, the growing number and requests of switches connected to the SDN controller can congest the control links, which implies using redundancy with clusters of controllers [65]. The controller architecture can be designed in different ways to reduce congestion. Moreover, load-balancing techniques are required to share the excessive load between controllers in the cluster [66]. The growing number of services in NFV implies more entities to manage with a growing number of faults and alarms. The alarms may originate from different network layers and resource types. For instance, these alarms can be traditional OS Syslog messages or other specific alarms defined for each NFV service. The distribution of VNFs over sites also provides more robustness to service failures. However, this brings a new management concern: whether to distribute or centralize the logs and the management system. Furthermore, in NFV a failure may propagate through layers and sites, increasing the number of alarms. Dealing with alarm storms requires modern storing and filtering techniques with early notifications of severe failures [15].

4.1.2 Topology

The flexibility of NFV enables consistent topology changes and components relocation in the network. Considering the actual network topology when a fault occurs is crucial to avoid false positives, i.e., pinpoint a faulty network component that is not running in the network anymore. Moreover, the coexistence of physical and virtual entities in the NVE architecture entails new types of dependencies that should be considered when building the diagnosis model. The topology is multi-layered as presented in Fig. 1. Therefore, while building the fault management engine, one should take into consideration the different management levels and consider the dependencies between layers. For instance, an infrastructure owner may consider managing only faults from the physical and virtual levels and charge the tenants for the management of their services. In the case of multiple tenants, challenging questions may arise such as the separation of clients alarms, notification of tenants, lack of visibility due to the clients' access restriction to the owner infrastructure, and sharing the management task between infrastructure owners and their clients.

4.1.3 Granularity

As depicted in Fig. 5, NVEs entities are divided in two types, *virtual* or *physical*. The managed target could be a node, a link, a network or a service. A node represents a Physical Network Function (PNF), a VNF, a virtual host (e.g. container, Virtual Machine (VM) or Unikernels) or a site. The VNFs are linked through virtual links, and sites through physical ones. Network management, as defined in [7], monitors a set of connected virtual and physical nodes. A service is a structured chain of VNFs and PNFs to support a specific application such as voice over IP. For a good accuracy of fault localization, one should consider a finer granularity like application processes and sub-components such as CPU and network interfaces.

4.1.4 Fault tolerance

The fault tolerance issues of SDN are mostly related to issues already present in traditional networks, i.e., link and node failures. However, the centralization of the SDN computation logic brings novel challenges. A single SDN controller presents a single point of failure which compromises the ability of the network nodes to forward packets, and eventually affects the entire network. In NVE, a physical server may host a large number of applications. In the case of a server failure a large number of services are impacted compared to traditional dedicated hardware.

4.1.5 Performance

Performance represents the ability to deliver the required Quality of Service (QoS). Performance is estimated through a number of network metrics (e.g. delay, jitter, and throughput) and factors such as the running environment, resilience of protocols and nodes, and network topology. Configuration choices, such as the maximum frequency at which network devices can be queried for information, or maintaining sufficient redundancy, may influence overall performance of the system. Another factor is the migration of tenant solutions with the same configuration parameters. Migration costs time and generates traffic that are crucial factors for guaranteeing the required QoS.

4.1.6 Other issues

Dealing with the above issues may generate additional problems. In SDN, to prevent scalability issues and increase performance, a number of controller cluster schemes are proposed. However, this raises synchronization and controller placement issues [14]. Moreover, even though the routing network updates has been studied intensively in traditional networks, in SDN the dynamic network topology and the consistent network updates come with additional challenges. Besides, rule updates communicated by the controllers to the network nodes may be delayed and asynchronous [13]. One way to prevent VNF failures is to deploy redundant nodes, and maintain a sufficient redundancy for better performance. Redundancy schemes and deployment techniques will be further addressed in Section 5.2.

4.2 Discussion

It is clear that the benefits of network virtualization outweigh their vulnerabilities, and our goal is not to claim that the NVEs are inherently less fault tolerant than current networks. Fault management in the same time faces new challenges, but also benefits from virtualization. In fact, thanks to the flexibility and programmability of NFV, VFs lifecycle and healing actions are automated. Another positive aspect of NFV, is that the distributed nature of the NFV infrastructure allows better network robustness. One example of robustness is the NVE ability to avoid fault propagation between services thanks to their distribution in different locations [67]. However, the requirements of fault management is remarkably changing, the management system should take into consideration the various granularity including the logical and virtual resources. But also the dynamic network topology, the distribution of the tenant resources in different locations, and the hardware and software dependencies. Moreover, in NVE, automation becomes unavoidable. Hence, the need to define self-X management axes, particularly self-healing and self-modeling of networks. The traditional fault management state of the art addressed in Section 3.3 presents a number of weaknesses next to the nature of NVE:

- Rule-based and case-based techniques should be adapted to the scalability and dynamicity of NVE. Templates and rules are used for static networks with the presence of expert knowledge. Acquiring this knowledge in a dynamic network such as NFV is difficult.
- Codebook and chronicles are rather fast and flexible pattern recognition techniques that identify problems from symptoms and alarms. Their weaknesses are that they provide less

explanation to what happened in the network and they are also sensitive to alarm losses and dealing with alarm storms.

- Petri nets and counterfactual can only reason on sequences of events and do not scale well.
- BNs and factor graphs can provide guidance in diagnosis. However, NVE's multi-layer dynamic topology increases the difficulty of defining an appropriate diagnosis model.
- Although ML methods are attractive because of their ability to interpret data efficiently, in NVEs the training data should be adapted to the features of virtual data and the dynamicity of NVE.

To showcase the NVE management issues and features we propose two fault scenarios illustrated in Fig. 6. The OPNFV DOCTOR use case in Fig. 6a, presents the different steps that should be followed for a rapid fault detection and notification of the affected tenants [68]. In this use case, one of the servers hosting the Openstack VMs is down. The fault propagates trough the virtualization layer and affects the running VM(s). Once the VIM has identified which VM(s) are affected, the concerned client(s) will be informed to get recovery instructions. In this case, client-1 switches to the standby VM-1. In the second use case (cf. Fig. 6b), a vIMS registration service fails with a timeout code 408. In this case, an important process in the proxy *Bono* application is out of service. In order to register, a client sends an authentication message to the proxy which forwards it to the router *Sprout* and tries to authenticate the client through *Homestead* (i.e., Home Subscriber Server (HSS)). Since the proxy is down, the client could not register and the router raises an alarm to indicate that the proxy is unreachable. This use case showcases the importance to consider a smaller granularity to capture sub-components faults like processes that may affect services. Other types of failures are expected in NVE such as SDN controller failures, controller-switch link failure or saturation due to an excessive load (cf. Tab. 2), and even orchestrator and monitoring subsystems failures that may impact a wide range of networks components [13].

5 Novel NVE fault management proposals

This section discusses current efforts to tackle the NVEs issues. These efforts involve the use of automatic actions in their solutions, introducing new concepts and reviving old ones that became feasible and useful in NVEs.

5.1 Self-modeling

To perform fault detection in most of the model-based techniques, building the model is an important step towards fault diagnosis and healing. A model of a system is the representation of its structure and functional nature. The model involves the network topology, the malfunction, and their consequences. The model can be constructed using expert knowledge, standards, collected data, network node dependencies, and statistical tests. For early network deployments, the construction of the model was much easier. The network was smaller with a fixed topology, and only standards and expert knowledge were sufficient to build the model.

However, with virtualization, the topology becomes dynamic which complicates the extraction of a generic model. Consequently, *Sánchez et al.* [69] and *Cherrared et al.* [70], proposed the use of the topology knowledge extracted from the controller in the case of SDN and from VIM and monitoring tools in the case of NFV, respectively. To model SDN components, *Sánchez et al.* [69], define two types of templates: the network node template and the link template. These templates represent the relationships between virtual and physical sub-components inside each network element. A template instantiation algorithm is then used to model the network according to the topology knowledge from controllers. *Cherrared et al.* [70], proposed a monitoring framework to provide real time topology data in NVE. This data will help providing a diagnosis model that matches the actual topology. This approach leverages Opensource monitoring tools with real time network infrastructure view. Metrics and alarms can also be provided by OpenStack tools like

ceilometer, AODH [71], and Monasca [72]. Other Opensource tools Nagios [9] and Zabbix [73], also provide efficient alarm notifications in virtual clouds.

5.2 Self-healing

Self-healing represents the autonomic ability for a system to recover from failures. In NVE, self-recovery reduces the time of healing as troubleshooting can occur while keeping the services running. To achieve self-recovery in NVE, several mechanisms have been developed and improved. Table 2 illustrates taxonomy of possible faults in NVE in each virtual network layer and the associated self-healing actions. As described in Section 4.2, the first scenario is a physical crash. In this case, the first action is to migrate the affected VM(s) from the server. The second use case represents an application crash, for which a solution is to instantiate another proxy docker or restart the process. In the following, important self-healing mechanisms and associated efforts are depicted:

Opensource recovery tools: OpenStack VIM offers some basic recovery actions when a VM is down using the Heat module that tries to rebuild the non-responsive VM. However, it does not support any failover or redundancy mechanism. Several solutions fill this gap by adding OpenStack failover plug-ins like Pacemaker [99]. However, the risk with redundancy is still remaining, in the case where even the failover VMs fail. *Moghaddam et al.* [84], proposed to face this problem, using a multi-agent system or HAstack plug-in to monitor the compute nodes and instances in OpenStack. The idea is to apply both the functionalities offered by PaceMaker and OpenStack modules. When a VM is down a new VM with the same configurations and services runs instead while the HAstack agents recover the failed VM, this process reduces considerably the recovery time to face the failover problem.

SDN controller failover: Redundancy is also a way to secure the fault tolerance of controllers. Indeed, OpenFlow, does not support the control plane restoration mechanism. A master and slave role can be assigned to controllers in recent OpenFlow versions; the master controller has the main control while slaves can only read the state of switches. However, to provide an effective failover mechanism we should answer arising questions like which controller is the best replacement for the failed one, how many updates are needed to keep the redundant controller aware of the current network topology or how many redundant controllers are needed. To address these questions, several contributions [85, 86], proposed more sophisticated controller failover solutions. For instance, *Pashkov et al.* [85], proposed a High-available controller (HAC) architecture based on adding a cluster middleware between the controller core and controller network services and applications to improve the failover process. However, proposing multi-controller architectures implies dealing with controller synchronization and network updates consistency overhead [13]. *Guo et al.* [100], propose to conduct effective state synchronizations among controllers only when the load of a specific server exceeds a certain threshold to reduce the synchronization overhead between controllers.

Channel or Link failover: OpenFlow-Switch can identify a failed link but has to wait for the controller to establish alternative routes. Some efforts [87–89], discussed the possible link failure recovery in SDN using restoration and protection models addressed in Section 3.3.3. *Hwang and Tang* [87], presented a failover solution for both the control and data channels. For the data channel, both the restoration and protection mechanisms are used. In the restoration process, the SDN controller computes backup paths based on the complete bipartite graphs of the network topology, whereas, in the protection mechanism the controller configures the flow entries of SDN switches. For the control channel recovery, *Hwang and Tang* [87], used a restoration mechanism to re-build the control path for switches based on weighted functions. To address the drawbacks of the restoration and protection process, i.e. the increase of failover times with the growing number of switches and dealing with dynamic changes of network states, respectively, *Ko et al.* [89], combined both approaches in a dynamic network hypervisor-based framework. The proposed method consists in the calculation of backup paths in a finer timeframe (e.g., 5 seconds), and the recovery flow rules are configured only if a physical network failure actually occurs. *Giatsios et al.* [88], proposed a proactive failover procedure to the tenant traffic slice, and whose traffic is

typically forwarded through tunnels. Whenever a link in the primary path fails, the flows using the tunnel are rerouted through a backup sub-path, while making this process transparent to tenants using the same tunnels. A link failure may also affect a NFV chain. *Soualah et al.* [90], proposed to minimize penalties induced by service interruptions due to link failures in the way VNFs are placed and chained in the presence of physical link failures. A decision tree approach is used to select reliable paths to prevent link failures and reactively reorient impacted virtual links in safer physical paths once an outage occurs.

NFV chain service failover: In Service Function Chain (SFC), the performance of a network service is determined by the performance of each Service Function (SF). *Lee and Shin* [91], addressed the problem of the service recovery delay when waiting for the SFC control plane to provide an alternative Service Function Path (SFP). The proposed scheme temporally recovers SFCs from the failure by redirecting traffic in case of an SF failure to another function stored in the list of a service function forwarding node (SFF) with only dataplane signaling. *Sauvanaud et al.* [75], proposed an approach to detect SLA violation and preliminary symptoms to identify the anomalous VM at the origin of the detected SLA violation. The proposed solution applied a random forest algorithm and a fault injection tool. The work targets the identification of anomalous behaviors from the monitoring data collected every 15sec from two sources: the hypervisor hosting the VMs and the OS of the VMs. A fault injection tool is used in order to precisely handle the occurrence of faults during the training phase. The experimental testbed was deployed on the vIMS developed by the Clearwater Opensource project [25].

Rollback: This is one of the well-known techniques providing fault tolerance. Rollback (or snapshot) is a way to recover a system transparently with low cost [92]. The snapshot represents the correct running state of a VNF. This state is periodically recorded into a stable storage to restore it when a failure occurs, significantly reducing the amount of lost computation. However, the rollback procedure extends the service downtime compared to normal system reinitialisation, primarily due to the significant size of the snapshot which is proportional to the VM size. *Cui et al.* [92] and *Shi et al.* [93], aim to reduce the rollback latency problem. For instance, *Cui et al.* [92], proposed a rollback system which takes advantage of the similarity among VMs, and leverages multicast to transfer the identical pages across VMs to disperse hosts with a single copy, rather than delivering them independently.

5.3 Fault prediction and proactive recovery

The architecture, operation and management of 5G networks are still in the design phase. The opportunity is then crucial for operators to perform some proactive operation that quickly detects the signs of critical failures and prevents future defects. To perform fault prediction, most efforts [74,75,78,96], proposed ML solutions deployed in large data or logs. *Sauvanaud et al.* [75], applied a random forest algorithm to predict the anomalous VMs. Papers [74, 78], proposed an offline methods to extract trends in network data that can enable the operator to proactively tackle similar faults in the future. *Gonzalez et al.* [78], used random forest enhanced with summarization and operations techniques for the automatic identification of dependencies between system events, to help network operators predict the root causes of error. Furthermore, *Cotroneo et al.* [74], proposed a dependability benchmark to support NFV providers making decisions about which management solutions can achieve the best dependability. For that purpose, they define a set of measures both to evaluate the impact of the injected faults on SLA, and to quantify the coverage and latency of fault management mechanisms. *Zhang et al.* [96], apply RNN LSTM to predict faults in NVE. In the simulation, 6 types of faults were injected in a number of VNFs simulated in OPNET. Results showed that the more serious the faults, the more accurate is the prediction. Prediction time was estimated to 800 s.

5.4 Fault injection

The fault injection process consists in a number of defects injected in the network. Different fault scenarios are possible depending on the injection target and the fault type. The target represents

the components and granularity of injection it could be a CPU usage, Process or network link. The faults could be applied to stress the network (e.g. CPU overload) or to disable the injection target (e.g. stop a physical server). Usually, the fault injection process is applied to test the network robustness such as the chaos monkey techniques used by Netflix. The aim is to collect network performance statistics under faults and evaluate the network ability to provide and maintain an acceptable SLA. *Cotroneo et al.* [74], proposed a dependability benchmark, which evaluates the quality of service in the presence of taxonomy of faults defined by the authors.

However, in NVE, so as to accelerate the occurrence of faults to provide the learning and validation data set for ML methods, the fault injection process is applied. In fact, providing this data could take several days, one such example is *Gonzalez et al.* [78], that waited 206 days for around 21,442 network events. Papers [75, 96], proposed the use of fault injection models to shorten this time. *Sauvanaud et al.* [75], triggered two methods of injections: at a local level by means of local injections in single VMs or, at a global level, by submitting heavy workloads. *Zhang et al.* [96], simulated in OPNET the fault model of six different severity types of faults with an occurrence that obeys to a Poisson distribution.

5.5 Log analytics and data mining

Log analytics was first performed by experts, but due to the growing number of connected elements, and a larger range of monitoring information, automation of log analytics became crucial. With the recent growth of storage capacity in servers and the distribution of data in clouds, storing huge amounts of data as big data lakes became possible. Data mining refers to the action of extracting pertinent information from big data sets. Anomaly detection with log analytics or mining remains an efficient technique in virtual networks since data is always a source of precious information. In the data mining process the first three steps (i.e., data integration, cleaning and selection) are used to prepare the data to apply data mining techniques [101]. As a final step data mining techniques like clustering and association analysis are deployed to discover the interesting patterns and evaluate them to make final decisions. Efforts [77,95], used this approach to extract interesting patterns for the anomaly detection in NVE. *Baseman et al.* [77], presented a method for anomaly detection based on syslog data collected from VMs. They extract the Infomap clusters on textual data; and relational features on numeric data and combine their features with keyword counts to create a single data set. Each message is assigned to a cluster according to the percentage of its textual tokens contained in each Infomap cluster.

5.6 Fault management Frameworks

With regards to the automation of fault management and the design of fault management architectures, several approaches [42,76,83,98], define FM frameworks. Most of the defined frameworks are based on autonomic management and standardization institutes definition such as the ITU-T's M30xx recommendation series that describe a general framework for Telecommunication Network Management [102]. The fault management framework is generally composed of interacting components. Each component has a specific function (i.e., modeling, detection or RCA). In most recent works, framework blocks are replaced by Opensource tools that provide the required functions. More recent efforts [70, 76, 83, 97, 98], proposed fault management frameworks for NVEs. For instance, *Luchian et al.* [98], used Openstack as a VIM for NFV and Nagios to collect the Available Transfer Rates and One-Way Delays of the links within the cloud, in order to provide the optimization of the Network Virtualization Functions. While *Song et al.* [76], used protocols in the composition of the framework: SNMP and Syslog for the event detector component and Openflow for communicating with a SDN controller. *Sanchez et al.* [83], presented a self-healing framework for the resiliency of 5G networks. The proposed Self-healing framework acts in the three planes of SDN (application, control, and data) and in the service plane, by making observations on the network and launching recovery actions. The OPNFV project Doctor proposes Vitrage [97], an OpenStack RCA tool that provides rapid alarm notifications using deduced alarms for virtual

and physical entities. Vitrage gets its data source from OpenStack modules and external Opensource tools such as Nagios and Zabbix. A real time topology mapping is provided. The diagnosis process is based on templates to express the different rules used for the RCA (RCA). *Cherrared et al.* [70], proposed LUMEN a global FM framework for NVE. LUMEN is a four-layer framework (i.e., source, sink, extraction and decision), the first three layers defined for network data monitoring are ensured by the Opensource Elastic stack [94].

6 Discussion

Tables 3 and 4 summarize the fault management efforts including Opensource projects. Table 3 depicts a classification of traditional and virtual networks efforts according to the different fault management steps. We can notice that most efforts focus on one or two fault management steps since it is hard to address all of them. The most recent efforts [69, 90, 91], tend to use automatic diagnosis and self-healing, but also Opensource tools [70, 84, 98]. Furthermore, some solutions like log analytics, self-modeling and fault prediction existed in traditional networks [37, 43, 57], and remain interesting approaches in NVEs [69,77]. Table 4 depicts the efforts that addressed the NVE issues. We notice that most current efforts [84,85,87–93], are proposing failover mechanisms. This is possible thanks to virtualization, in contrast to the case of PNF. The coexistence of physical and virtual functions is likely to happen in telecommunications networks and should be considered. Fault localization proposals [75,78,97], are focusing on the management of the virtual layer. Faults in application or process levels that generate service alarms such as those discussed in Section 4.2 are not addressed. In NVE, working with ML techniques is not straightforward, since we should translate the complex network data to define the learning model that is appropriate for each learning use case. For instance, if we wish to detect a faulty state of the network, we should transform the logs to be considered as an entry for our ML algorithm. In the case of *K-means* logs should be transformed to significant numeric values. In fault localization [75, 78], random forest with decision trees is the most applied technique, since decision trees model decisions and their possible consequences, which is a very intuitive way for data modeling. Moreover, since logs are text files, efforts such as [96] tend to apply ML techniques that deal better with text such as LSTM.

Recent approaches applied self-modeling and data filtering techniques to deal with the limitation of fault localization techniques addressed in Section 4.2. However, the research efforts on fault management of NVE are still ongoing. There are still issues that need to be addressed and others that are insufficiently explored by present efforts. Black-box ML techniques perform very well when it comes to extracting features from data and predicting the future behavior of the network state [16]. However, the learned features should provide a non-acquired knowledge and avoid learning knowledge that is easily extracted from data such as the dependencies due to network topology [78]. For instance, the dependency between a VM and the hosting server is explicit. Furthermore, in the case of accurate fault localization with a fine granularity, white-box approaches such as BNs provide better fault explanations since sub-components such as processes can be considered in the model.

To reduce the diagnosis uncertainty a real time self-diagnosis is crucial. The efforts presented in Section 5 consider a time window for the data collection or alarms corelation (e.g. a time window of 15sec in [75]). While in dynamic networks the choice of a time window may affect the diagnosis accuracy and provide false or outdated results.

7 Conclusion

5G relies on the coexistence of a variety of architectures: NFV and SDN. The coexistence of these distinct and complementary technologies opens up new fault management issues that can hardly be addressed by classical methods. This survey reviews the canonical fault management steps and efforts applied to classical telecommunication networks. We also highlighted significant

issues to be considered in the fault management of NVE, described some of the recent research achievements and future directions.

Acknowledgment. We would like to thank the reviewers for their thoughtful comments and efforts towards improving our survey. We also thank *Christian JACQUENET* and *Ian HAY* for their careful reading of the survey and their constructive remarks.

References

[1] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, pp. 90–97, Feb 2015.

[2] "Telecom Infra Project." `https://telecominfraproject.com`, [Online; accessed 09/08/2019].

[3] K. Katsalis *et al.*, "5g architectural design patterns," in *IEEE International Conference on Communications Workshops (ICC)*, 2016.

[4] "NORMA." `https://5g-ppp.eu/5g-norma/`, [Online; accessed 11/08/2019].

[5] E. H. Rachid *et al.*, "Ngmn alliance 5g white paper; reliability; report on models and features for end-to-end reliability v1.0," 2015. `https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0_01.pdf`[Online; accessed 23/10/2019].

[6] "BEREC Report on infrastructure sharing." `https://berec.europa.eu/eng/document_register/subject_matter/berec/reports/8164-berec-report-on-infrastructure-sharing` [Online; accessed 09/08/2019].

[7] R. P. Esteves *et al.*, "On the management of virtual networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 80–88, 2013.

[8] C. Cérin *et al.*, "Downtime statistics of current cloud solutions," tech. rep., 2013. `http://iwgcr.org/wp-content/uploads/2013/06/IWGCR-Paris.Ranking-003.2-en.pdf.`, [Online; accessed 23/10/2019].

[9] "Nagios." `https://www.nagios.com/`, [Online; accessed 09/08/2019].

[10] P. Fonseca and E. Mota, "A survey on fault management in software-defined networks," *IEEE Communications Surveys Tutorials*, 2017.

[11] A. S. da Silva, P. Smith, A. Mauthe, and A. Schaeffer-Filho, "Resilience support in software-defined networking: A survey," *Computer Networks*, vol. 92, no. Part 1, pp. 189 – 207, 2015.

[12] J. Chen, J. Chen, F. Xu, M. Yin, and W. Zhang, *When Software Defined Networks Meet Fault Tolerance: A Survey*, pp. 351–368. Cham: Springer International Publishing, 2015.

[13] K. Foerster, S. Schmid, and S. Vissicchio, "Survey of consistent software-defined network updates," *IEEE Communications Surveys Tutorials*, vol. 21, pp. 1435–1461, Secondquarter 2019.

[14] O. Hohlfeld *et al.*, "Guest editorial scalability issues and solutions for software defined networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2595–2602, 2018.

[15] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard, "Dependability of the nfv orchestrator: State of the art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3307–3329, 2018.

[16] R. Boutaba *et al.*, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, p. 16, Jun 2018.

[17] M. A. Alsheikh, S. Lin, D. Niyato, and H. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Communications Surveys Tutorials*, 2014.

[18] P. V. Klaine, M. A. Imran, *et al.*, "A survey of machine learning techniques applied to self-organizing cellular networks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2392–2431, 2017.

[19] J. Xie *et al.*, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys Tutorials*, 2019.

[20] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan 2015.

[21] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: Sdn for big data and big data for sdn," *IEEE Network*, vol. 30, pp. 58–65, January 2016.

[22] G. ETSI, "Network functions virtualisation (nfv); use cases," *V1*, vol. 1, pp. 2013–10, 2013.

[23] M. Mechtri, C. Ghribi, and D. Zeghlache, "A scalable algorithm for the placement of service function chains," *IEEE Transactions on Network and Service Management*, vol. 13, pp. 533–546, Sept 2016.

[24] B. Yi, X. Wang, K. Li, M. Huang, *et al.*, "A comprehensive survey of network function virtualization," *Computer Networks*, 2018.

[25] "Clearwater." `http://www.projectclearwater.org/`, [Online; accessed 07/08/2019].

[26] M. Zimmerman, D. Allan, M. Cohn, *et al.*, "Openflow-enabled sdn and network functions virtualization," *Solution Brief, ONF, Solution Brief sbsdn-nvf-solution. pdf*, vol. 1, 2014.

[27] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[28] I. Afolabi, T. Taleb, K. Samdanis, *et al.*, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys Tutorials*, 2018.

[29] N. ETSI, "Network functions virtualisation (nfv); management and orchestration," *NFV-MAN*, vol. 1, p. v0, 2014.

[30] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annual Reviews in Control*, 2013.

[31] J. C. M. F. M. Schoffstall and C. Davin, "Rfc 1157: Simple network management protocol (snmp)," *IETF, April*, 1990.

[32] R. Gerhards, "The Syslog Protocol." RFC 5424, Mar. 2009.

[33] ISO, *ISO 8601:2000. Data elements and interchange formats — Information interchange — Representation of dates and times.* 2000.

[34] M. lgorzata Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, vol. 53, no. 2, pp. 165 – 194, 2004.

[35] S. Russell and P. Norvig, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[36] L. N. De Barros, M. Lemos, *et al.*, "Model based diagnosis for network communication faults," in *proceedings of the international workshop on artificial intelligence for distributed information networking*, 1999.

[37] C. Hounkonnou and E. Fabre, "Empowering self-diagnosis with self-modeling," in *8th international conference on network and service management (cnsm)*, pp. 364–370, 2012.

[38] I. Ben-Gal, *Bayesian Networks*. John Wiley & Sons, Ltd, 2008.

[39] J. Lu, C. Dousson, and F. Krief, "A self-diagnosis algorithm based on causal graphs," in *The Seventh International Conference on Autonomic and Autonomous Systems, ICAS*, vol. 2011, 2011.

[40] M. Hasan, B. Sugla, and R. Viswanathan, "A conceptual framework for network management event correlation and filtering systems," in *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on*, pp. 233–246, IEEE, 1999.

[41] R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, and A. Benveniste, "A petri net approach to fault detection and diagnosis in distributed systems. i. application to telecommunication networks, motivations, and modelling," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 1, pp. 720–725 vol.1, Dec 1997.

[42] P. Varga and I. Moldovan, "Integration of service-level monitoring with fault management for end-to-end multi-provider ethernet services," *IEEE Transactions on Network and Service Management*, June 2007.

[43] S. R. Tembo, J. L. Courant, and S. Vaton, "A 3-layered self-reconfigurable generic model for self-diagnosis of telecommunication networks," in *SAI Intelligent Systems Conference (IntelliSys)*, 2015.

[44] I. Ben-Gal, F. Ruggeri, F. Faltin, and R. Kenett, "Bayesian networks, encyclopedia of statistics in quality and reliability," 2007.

[45] R. Dechter, *Constraint Processing*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[46] and F. R. Kschischang, , and S. Pasupathy, "A factor graph approach to link loss monitoring in wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 820–829, April 2005.

[47] D. Lewis, "Causation," *The journal of philosophy*, vol. 70, no. 17, pp. 556–567, 1974.

[48] J. Y. Halpern, "A modification of the halpern-pearl definition of causality," *CoRR*, vol. abs/1505.00162, 2015.

[49] A. Balke and J. Pearl, "Counterfactual probabilities: Computational methods, bounds and applications," *CoRR*, vol. abs/1302.6784, 2013.

[50] G. Gössler and J.-B. Stefani, "Fault ascription in concurrent systems," in *International Symposium on Trustworthy Global Computing*, pp. 79–94, Springer, 2015.

[51] P. Laborie and J.-P. Krivine, "Automatic generation of chronicles and its application to alarm processing in power distribution systems," in *International Workshop on Principles of Diagnosis (DX'97)*, 1997.

[52] G. Reali and L. Monacelli, "Definition and performance evaluation of a fault localization technique for an ngn ims network," *IEEE Transactions on Network and Service Management*, 2009.

[53] R. N. Cronk, P. H. Callahan, and L. Bernstein, "Rule-based expert systems for network management and operations: an introduction," *IEEE Network*, vol. 2, pp. 7–21, Sept 1988.

[54] S. Shankar and O. Satyanarayanan, "An automated system for analyzing impact of faults in ip telephony networks," in *Network Operations and Management Symposium.10th IEEE/IFIP*, pp. 1–4, 2006.

[55] L. Lewis, "A case-based reasoning approach to the management of faults in communications networks," in *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*, 1993.

[56] C. Melchiors and L. Tarouco, "Fault management in computer networks using case-based reasoning: Dumbo system," *Case-Based Reasoning Research and Development*, pp. 721–721, 1999.

[57] G. Zhaojun and W. Chao, "Statistic and analysis for host-based syslog," in *2010 Second International Workshop on Education Technology and Computer Science*, vol. 2, pp. 277–280, March 2010.

[58] A. I. Moustapha and R. R. Selmic, "Wireless sensor network modeling using modified recurrent neural networks: Application to fault detection," in *2007 IEEE International Conference on Networking, Sensing and Control*, pp. 313–318, April 2007.

[59] M. Adda, K. Qader, and M. Al-Kasassbeh, "Comparative analysis of clustering techniques in network traffic faults classification," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 5, pp. 6551–6563, 5 2017.

[60] A. Akbari and N. Beikmahdavi, "Cluster-based and cellular approach to fault detection and recovery in wireless sensor networks," in *3rd International Conference ICACTE*, 2010.

[61] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *ACM SIGCOMM Computer Communication Review*, 2011.

[62] A. S. Roque, D. Pohren, T. J. Michelin, *et al.*, "Eft fault impact analysis on performance of critical tasks in intravehicular networks," *IEEE Transactions on Electromagnetic Compatibility*, 2017.

[63] S. Huang, Z. Deng, and S. Fu, "Quantifying entity criticality for fault impact analysis and dependability enhancement in software-defined networks," in *Performance Computing and Communications Conference (IPCCC), 2016 IEEE 35th International*, pp. 1–8, 2016.

[64] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network recovery: Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[65] T. Hu *et al.*, "Multi-controller based software-defined networking: A survey," *IEEE Access*, vol. 6, pp. 15980–15996, 2018.

[66] A. A. Neghabi *et al.*, "Load balancing mechanisms in the software defined networks: A systematic and comprehensive review of the literature," *IEEE Access*, vol. 6, pp. 14159–14178, 2018.

[67] C. J. Bernardos, A. Rahman, J.-C. Zúñiga, *et al.*, "Network Virtualization Research Challenges," Internet-Draft draft-irtf-nfvrg-gaps-network-virtualization-06, Internet Engineering Task Force, July 2017.

[68] "Opnfv doctor project:." `https://wiki.opnfv.org/display/doctor/Doctor+Home`, [Online; accessed 11/08/2019].

[69] J. M. Sánchez *et al.*, "Self-modeling based diagnosis of software-defined networks," in *Network Softwarization (NetSoft)*, IEEE, 2015.

[70] S. Cherrared, S. Imadali, E. Fabre, and G. Goessler, "Lumen: A global fault management framework for network virtualization environments," in *21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Feb 2018.

[71] "Openstack telemetry tools:." `https://wiki.openstack.org/wiki/Telemetry`, [Online; accessed 07/08/2019].

[72] "Openstack monasca." `http://monasca.io/`, [Online; accessed 09/08/2019].

[73] "Zabbix:." `https://www.zabbix.com/`, [Online; accessed 02/08/2019].

[74] D. Cotroneo, L. De Simone, and R. Natella, "Nfv-bench: A dependability benchmark for network function virtualization systems," *IEEE Transactions on Network and Service Management*, 2017.

[75] C. Sauvanaud *et al.*, "Anomaly detection and root cause localization in virtual network functions," in *27th International Symposium on Software Reliability Engineering (ISSRE)*, 2016.

[76] S. Song, S. Hong, X. Guan, B.-Y. Choi, and C. Choi, "Neod: network embedded on-line disaster management framework for software defined networking," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, pp. 492–498, IEEE, 2013.

[77] E. Baseman, S. Blanchard, Z. Li, and S. Fu, "Relational synthesis of text and numeric data for anomaly detection on computing system logs," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 882–885, Dec 2016.

[78] J. M. N. Gonzalez *et al.*, "Root cause analysis of network failures using machine learning and summarization techniques," *IEEE Communications Magazine*, 2017.

[79] K. Slavicek *et al.*, "Mathematical processing of syslog messages from routers and switches," in *4th International Conference on Information and Automation for Sustainability*, 2008.

[80] H. Tsunoda and G. M. Keeni, "Managing syslog," in *16th Asia-Pacific Network Operations and Management Symposium*, pp. 1–4, 2014.

[81] S. Shengyan, S. Xiaoliu, Z. Jianbao, and M. Xinke, "Research on system logs collection and analysis model of the network and information security system by using multi-agent technology," in *Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security*, IEEE Computer Society, 2012.

[82] Y. Cui *et al.*, "Fault propagation reasoning and diagnosis for computer networks using cyclic temporal constraint network model," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[83] J. M. Sanchez Vilchez, I. Grida Ben Yahia, N. Crespi, T. Rasheed, and D. Siracusa, "Softwarized 5G Networks Resiliency with Self-Healing," in *5GU - 1st International Conference on 5G for Ubiquitous Connectivity*, (Levi, Finland, Finland), Nov. 2014.

[84] F. F. Moghaddam, A. Gherbi, and Y. Lemieux, "Self-healing redundancy for openstack applications through fault-tolerant multi-agent task scheduling," in *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*, pp. 572–577, IEEE, 2016.

[85] V. Pashkov *et al.*, "Controller failover for sdn enterprise networks," in *Modern Networking Technologies(MoNeTeC)*, IEEE, 2014.

[86] M. Obadia *et al.*, "Failover mechanisms for distributed sdn controllers," in *Network of the Future (NOF)*, IEEE, 2014.

[87] R.-H. Hwang and Y.-C. Tang, "Fast failover mechanism for sdn-enabled data centers," in *International Computer Symposium (ICS)*, IEEE, 2016.

[88] D. Giatsios, K. Choumas, P. Flegkas, T. Korakis, and D. Camps-Mur, "Sdn implementation of slicing and fast failover in 5g transport networks," in *Networks and Communications (EuCNC), 2017 European Conference on*, pp. 1–6, IEEE, 2017.

[89] K. Ko, D. Son, J. Hyun, J. Li, Y. Han, and J. W.-K. Hong, "Dynamic failover for sdn-based virtual networks," in *Network Softwarization (NetSoft), 2017 IEEE Conference on*, pp. 1–5, IEEE, 2017.

[90] O. Soualah *et al.*, "A link failure recovery algorithm for virtual network function chaining," in *IFIP/IEEE Symposium on Integrated Network and Service Management*, IEEE, 2017.

[91] S.-I. Lee and M.-K. Shin, "A self-recovery scheme for service function chaining," in *Information and Communication Technology Convergence (ICTC), 2015 International Conference on*, pp. 108–112, IEEE, 2015.

[92] L. Cui, Z. Hao, Y. Peng, and X. Yun, "Piccolo: A fast and efficient rollback system for virtual machine clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2328–2341, 2017.

[93] B. Shi, D. Chen, L. Cui, J. Zheng, and B. Li, "Mercurial: A traffic-saving roll back system for virtual machine cluster," in *Proceedings of the IEEE/ACM 7th International Conference on Utility and Cloud Computing*, UCC '14, (Washington, DC, USA), 2014.

[94] "Elastic Stack." `https://www.elastic.co/products/elasticsearch`, [Online; accessed 09/08/2019].

[95] U. S. Hashmi, A. Darbandi, and A. Imran, "Enabling proactive self-healing by data mining network failure logs," in *International Conference on Computing, Networking and Communications (ICNC)*, 2017.

[96] L. Zhang, X. Zhu, S. Zhao, and D. Xu, "A novel virtual network fault diagnosis method based on long short-term memory neural networks," in *86th Vehicular Technology Conference (VTC-Fall)*, IEEE, 2017.

[97] "Vitrage." `https://wiki.openstack.org/wiki/Vitrage`, [Online; accessed 08/08/2019].

[98] E. Luchian *et al.*, "Advanced monitoring of the openstack nfv infrastructure: A nagios approach using snmp," in *12th IEEE (ISETC)*, pp. 51–54, IEEE, 2016.

[99] "Pacemaker, cluster labs." `http://clusterlabs.org/`, [Online; accessed 11/08/2019].

[100] Z. Guo *et al.*, "Improving the performance of load balancing in software-defined networks through load variance-based synchronization," *Computer Networks*, vol. 68, pp. 95–109, 08 2014.

[101] E. Rozaki, "Network fault diagnosis using data mining classifiers," *Eleni Rozaki International Journal of Data Mining & Knowledge Management Process*, 2015.

[102] I.-T. R. M.3010, "Principles for a telecommunications management network," 2000.

Table 1: SDN and NFV multi-tenant environments: fault management issues and consequences.

	ISSUES	SDN?	NFV?	Multi-tenant?	CONSEQUENCES
Scalability	Numerous requests to the SDN controller	✓			Limited SDN controller capacity
	Huge number of faults and alarms	✓	✓	✓	Alarms correlation and fault detection
	A growing number of resources			✓	Optimization of resources allocation
Topology	Dynamic topology		✓		Difficulty of modeling the network topology
	Virtual network multi-layers		✓		Multi-layer, multi-resolution faults dependencies
	Multi-tenants			✓	Lack of network visibility
Granularity	Complex FM entities	✓	✓	✓	Choose the right granularity for FM accuracy
Fault Tolerance	A single SDN controller failure	✓			Disabled network communications
	Link failure	✓	✓		Loss of network connections
	end-to-end service chain crash		✓		Rapid detection and recovery for clients services
	Physical and virtual faults	✓	✓		A physical failure may impact multiple virtual entities
	Multi-tenants faults			✓	Tenants faults isolation and notification
Performance	SDN controller and vSwitch performance	✓			SDN performance depends on the network topology.
	Running environments		✓		VNF performance depends on the virtual host
	Migration of tenants applications or VNF		✓	✓	Keeping the same performance while migrating a VNF

Table 2: Taxonomy of faults in NVEs.

Layers	Fault types	SDN/NFV faults	Self-Healing actions
Physical	Maintenance	Maintenance of servers	1- Send Maintenance notification to the clients, 2- Turn the status of the server to a maintenance state.
	Physical Crash	Failure in a switch physical port.	Transfer the logical interfaces into another physical port
		SDN Controller failure.	Balancing to a secondary controller.
		Link down	Look for alternative path
		Crash of a physical machine (PM).	1- Change the state of (PM), 2-Transfer the VNFs running on the PM, 3- Empty the PM,for maintenance.
Virtual	Crash/ Saturation	VM crash or VM high cpu load.	Instantiate other VM, or extend the memory and CPU of the VM.
		Saturation of a vSwitch port.	Instantiate another port or vSwitch.
Application/ Service	Configuration	Switch ignores how to reach clients.	Reconfiguration of the controller or switch
		Bridge misconfigured.	Bridge reconfiguration.
		SFC Misconfiguration.	SFC reconfiguration.
	Crash/ Saturation	Too many requests to a VNF service.	1- Replicate the VNF, 2- Increase memory and CPU of VNF host.
		Crash of end-to-end service.	1- Reinitiate involved VNFs, 2- Restart end-to-end service.
		VNF application crash.	Restart or Instantiate the VNF host.

Table 3: A classification of fault management efforts.

	Fault detection		**Fault**	**Impact**	**Fault**	**Self-**	**Log analytics**	**Self-**	**FM Frame-**
Papers Ref.	Metrics	Alarms	**localization**	**analysis**	**prediction**	**modeling**	**data mining**	**healing**	**work**
[37, 69]		✓	✓			✓			
[46, 71, 72, 74–76]	✓								
[57, 70, 77–80] [9, 73, 81, 82]		✓							
[43, 75, 82]			✓						
[54, 62, 63]				✓					
[83–93]								✓	
[77, 94, 95]							✓		
[74, 75, 78, 96]					✓				
[42, 70, 76] [83, 97, 98]									✓

Table 4: Novel fault management efforts in NVE.

Papers Ref.	**Environment**	**Added values**	**FM steps**
[75]	vIMS VMs	Random forest	Fault prediction, & training set with fault injection
[70]	NFV	Network data filtering	Multi-tenant fault isolation & NFV data filtering
[97]	Openstack VMs	Rule-based	Rapid fault detection and recovery & clients notification
[78]	VMs	Random forest	Proactive FM
[69]	SDN	BN	Self-modeling
[84]	Openstack VMs	Failover mechanism	Self-healing
[85]	SDN controller	Failover mechanism	Self-healing
[87], [89] and [88]	SDN channel or link	Failover mechanism	Self-healing
[90]	NFV link	Failover mechanism	Self-healing
[91]	SFC	Failover shifting mechanism	Self-healing
[74]	NFV vIMS VMware ESXi/vSphere and Linux/Docker	Dependability benchmark	Dependability, performance & evaluations with fault injection
[96]	NFV	RNN LSTM	Fault prediction & training set with fault injection
[92] and [93]	VMs	Rollback	Self-healing