LUMEN: A Global Fault Management Framework For Network Virtualization Environments

Sihem Cherrared, Sofiane Imadali, Eric Fabre, Gregor Gössler

LUMEN: A Global Fault Management Framework For Network Virtualization Environments

Sihem Cherrared, Sofiane Imadali
Orange labs networks
44 Avenue de la République,
92320 Châtillon, France
Email: firstname.lastname@orange.com

Eric Fabre
INRIA
IRISA, Campus de Beaulieu,
F-35042 Rennes Cedex, France
Email: firstname.lastname@inria.fr

Gregor Gössler
Univ. Grenoble Alpes, INRIA, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
Email: firstname.lastname@inria.fr

Abstract

The advent of 5G and its ever increasing stringent requirements for bandwidth, latency, and quality of service pushes the boundaries of what is feasible with legacy Mobile Network Operators' technologies. Network Function Virtualization (NFV) is one promising attempt at solving some of those challenges that were widely adopted by the industry and the standardization bodies. At its essence, NFV is about running network functions as software workloads on commodity hardware to optimize deployment costs and simplify the life-cycle management of network functions. However, it introduces new fault management challenges including dynamic topology, multi-tenant fault isolation and data consistency and ambiguity; that we propose to define in this paper. To tackle those challenges, we extend the classical fault management process to the virtualized functions by introducing LUMEN: a Global Fault Management Framework. Our approach aims at providing the availability and reliability of the virtualized 5G end-to-end service chain. LUMEN includes the canonical steps of the fault management process and proposes a monitoring solution for all types of Network virtualization Environments. Our framework is based on open source solutions and could easily be integrated with other existing autonomic management models.

Index terms — Fault management, Network Virtualization Environment, Self-diagnosis.

1 Introduction

The telecommunication industry is facing increased competition as new players are entering with emerging software technologies and open source projects. One such project, *Telecom Infra Project* [1] was initiated by Facebook and oriented towards an open source and general purpose hardware for a new generation of Mobile Network Operators (MNOs). The current MNOs infrastructures consist of a large range of proprietary hardware appliances [18]: there is thus a need for investment and adoption of new open technologies such as Software Defined Networking (SDN), Network Function Virtualization (NFV), and cloud-native innovations. Therefore, MNOs are investing a lot of resources considering the softwarization of network functions and are joining open source software communities. Most notably, MNOs contribute to OpenStack for infrastructure, OpenDayLight for SDN controllers, and ONAP for network automation [6] [5].

The current impulse to focus MNOs efforts on softwarization comes from the advent of the next mobile network generation (5G). The objective is the projected benefits the MNOs expects from such a paradigm shift when compared to how traditional networks are operated. The benefits include the reduction of costs (management and deployment), optimization to maximum resource

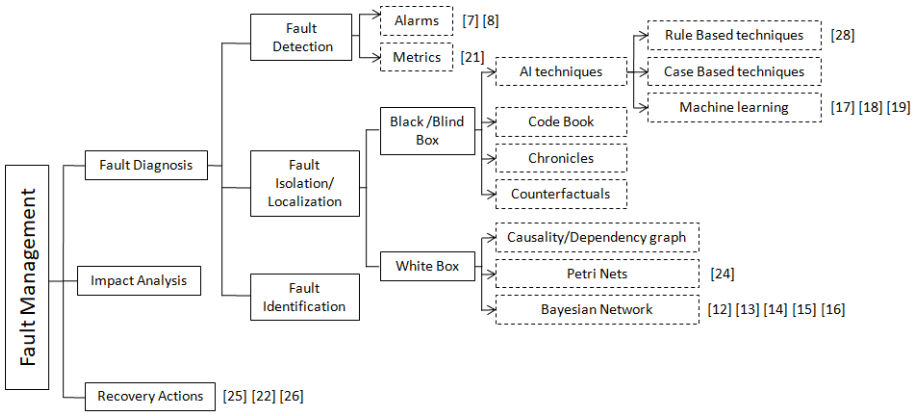

Figure 1: The Fault Management Steps

utilization, on demand programmable services, infrastructure and services sharing, and allowing a large variety of ecosystems to co-exist including open source systems [23].

Network management is critical for service availability, reliability and resiliency of networks. It includes the X-management axes or FCAPS : Fault, Configuration, Accounting, Performance, and Security. In the current SDN and NFV context, in addition to the possibility of multi-tenancy and slicing in 5G [26], new fault management issues and vulnerabilities are expected. The fault management issues in Network Virtualization Environments (NVEs) address scalability of network functions, lack of network visibility, and dynamic network topologies and their impact. The focus of our paper is to tackle those fault management issues in NVEs by introducing a Global Fault Management Framework (LUMEN).

LUMEN includes the canonical fault management steps and provides a new vision for monitoring all types of NVEs (e.g. SDN, NFV, monolithic or distributed functions, single and multi-tenant environments). One of LUMEN's objectives is to provide availability and reliability for VNFs deployed end-to-end service chains (NAT, firewalling, intrusion detection, DNS, caching and more). Our framework could also be integrated with other fault management Self-X models: Self-modeling, Self-diagnosis and Self-healing. Our contributions in this paper include:

- A definition of the different fault management steps and a classification of main approaches in the state of the art;
- A description of multi-tenant Network Virtualization Environments and the related challenges;
- Our proposal, LUMEN, to tackle the reliability, availibility, and resiliency of VNF chains in multi-tenant NVE;
- A discussion of LUMEN's advantages over related solutions.

The remainder of this paper is as follows. Section 2 discusses related work addressing fault management in traditional and softwarized networks. Section 3 summarizes the fault management issues and challenges for NVEs. Section 4 presents the different steps of LUMEN framework. Section 5 discusses the advantages of using our solution. Section 6 concludes the paper with some perspectives.

2 Related work

In this Section, we provide a definition of the classical fault management steps and techniques, the recent approaches in NVEs and the efforts done towards the automation of the fault management

procedures and their architecture.

2.1 Fault Management

Fault management covers the detection and storage of events and alarm notifications, filtering procedures for these alarms, and diagnostic checks for root cause localization, impact analysis and corrective actions. Events can be related to metrics and Key Performance Indicators (KPIs), such as delay, jitter, or response times. Notifications can be SNMP traps [10] or system log-files entries using syslog protocol [16]. The Fault-diagnosis process consists of the detection of the faulty state or the failure, the localization of faulty entities in the network and the identification of the fault types [21, 15, 34].

We propose the taxonomy in Figure 1 to help classify some of the main network fault management techniques. In the fault detection step the main approaches use alarms or metrics and once the faulty state is detected comes the fault localization and root cause analysis methods. We classify the state of the art into two main branches: *white and black box* techniques. As a definition, *white box are techniques that give a clear view about the building process such as the dependency graphs methods*, while in the *black box techniques, it is harder to deduce from the network what exactly has the framework learned.* Neural Networks for instance is in the latter case.

The two approaches can be combined. White box models such as Bayesian networks [32, 19, 9, 31, 27] use the network dependency graph to pinpoint faulty components and black box Artificial Intelligence (AI) techniques can be applied to detect a faulty state [35, 36, 20]. The second phase is the recovery process. In the recovery process, the first step consists of identifying the impact of faults i.e. the affected entities and the more critical faults to be treated first. After that, the framework can proceed to the healing phase. These approaches usually rely on the expertise of network administrators and their knowledge of the underlying infrastructure.

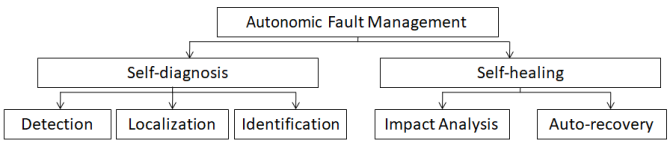

Figure 2: Autonomic Fault Management Steps for NVEs

2.2 Fault Management in Network Virtualization Environments

Autonomic management is a topic of interest for many organization (ETSI, IETF and SON) and is proposed in the design of the 5G architecture management and orchestration [14]. Recent efforts [19, 32, 27, 29, 30], introduced automation in the fault management steps. For instance, Hounkonnou et al [19] proposed a self-modeling concept to face the problem of scalability of the IP Multimedia Sub-system (IMS). The authors use instances of generic Bayesian networks adapted to the actual topology. Still, the experts knowledge on the IMS architecture helps to extract the generic model, which is not possible in dynamic networks based on NFV where the topology changes by integrating new tenants on the shared infrastructure. The automation of the steps was also introduced in some recent research papers that address SDN and NFV environments with self-diagnosis methods using Bayesian networks for certain efforts [27]. While others [29, 30], propose Self-healing for NFV chain failures.

Figure 2 summarizes the different fault management steps and presents a new fault management outline with the Self-X concepts.

In the telecommunication industry, the ITU-T's M30xx recommendation series describe a general framework for Telecommunication Network Management [22]. The framework, while applied to certain deployments, has been enriched to accommodate new challenges and overcome the general inflexibility of their hierarchical setups [33].

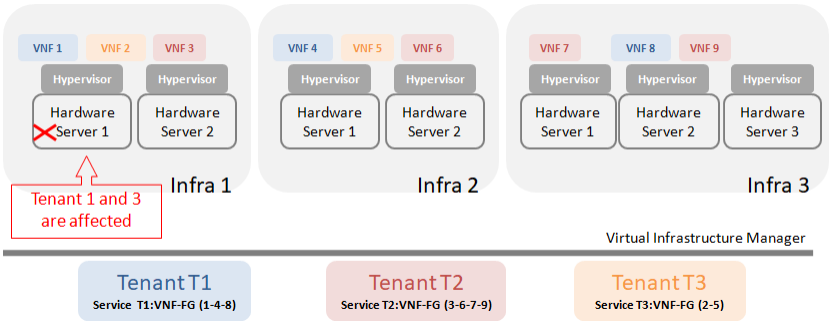

Figure 3: Distributed VNFs in a Multi-tenant environment

With regards to automation of fault management and the design of fault management architectures, several approaches can be found in the literature. Sánchez et al. [28], presented a self-healing framework for the resiliency of 5G networks. The proposed Self-healing framework acts in the three planes of SDN (management, control, and data) and in the service plane, by taking observations from the network and launching recovery actions. The Self-healing framework focuses more in the autonomic recovery actions and the SDN environment issues. Abhishek et al. [8], presented a bug detection, debugging, and isolation (BuDDI) middlebox architecture for SDN controllers. The proposed solution focuses on software bugs, it includes a common mode failure and dependencies (CMFD) migration module that prevents the system to recur into the same software failures after a fail over.

Authors of [33] proposed a fault management framework (ISF) integrated to a service-level monitoring for an end-to-end Ethernet service. The framework handles different management modules. The root cause analysis module is based on Petri nets. While the architecture is general enough to accept heterogeneous use cases, authors of [28] limited their case studies to network faults and Service Level Agreement (SLA) violation on the network, regardless of the network functions and their operational conditions that might trigger them.

The Open Network Function Virtualization (OPNFV) is an open source project (hosted by the Linux Foundation) for facilitating the development and evolution of NFV components across various open source ecosystems with automated testing to enable accelerated NFV development and reproducible deployments [25]. The OPNFV project Doctor proposes Vitrage, a project proposed by NTT DOCOMO and developed by the OPNFV DOCTOR community in collaboration with Nokia [3]. Vitrage is an OpenStack root cause analysis 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 Root Cause Analysis (RCA). The templates represent static detection and alarming scenarios written by the infrastructure administrators and network functions experts.

Previously mentioned tools and research prototypes do not address the fault management of the whole NVEs including NFV deployments (monolithic and distributed) in a multi-tenant environments.

3 NVEs challenges and issues

NVEs enables distinct network architectures to coexist in a single infrastructure without affecting the network performance [12]. NVEs include interactions between NFV and SDN components, physical and virtual functions in multi-tenants and multi-services environment. The network infrastructure connecting Virtual Network Functions (VNFs) into a chain will be handled by the SDN controller while each SDN application (e.g. Virtual Switch) will itself represent a VNF.

However, a new architecture implies new challenges and new ways to perform management. In the following, we discuss the main issues of NVEs with a fault management viewpoint. Most of the challenges described in this section are addressed by our framework in Section 4.

- **Number and type of managed objects**: NVEs features mentioned above enable deployment of more services involving more entities to be managed. The managed entities have different granularities: Logical resources (e.g. container, Virtual Machine (VM) or libOs.), physical and virtual network functions (e.g. Physical Switch and Virtual Switch) and subcomponents (e.g. CPU and network interfaces). Moreover, the number of faults is expected to augment due to the large amount of services and virtual entities. Network administrators will face the problem of huge alarms quantities, alarm loss, delay and consistency. Therefore, managing logs and metrics without automated solutions became impossible.

- **Dynamic network topology**: 5G will enable deployment of real-time services tailored to customers' requests. The service is represented as a VNF Forwarding Graph (VNF-FG) [13, 17], which is a chain of connected VNFs. Each VNF deploys a specific function on the service. A good example of such chain is the Clearwater project for IMS [7]. Other examples are Firewalls, Traffic Detection Function (TDF), or Traffic Steering Support Function. This real time service deployment makes the evolution of network topology and the entities dependencies in the network unpredictable. The managed system should consider the real time topology changes to pinpoint the faulty component and avoid false or outdated results.

- **Lack of network visibility**: The distribution of resources and virtual functions in different infrastructures and distinct locations leads to a network visibility problem. Figure 3 depicts how a unique VNF-FG service is distributed in different locations. Moreover, The state of each VNF is related to the state of hardware servers where the VNF is running. Therefore, to keep the service availability, a global network view is necessary.

- **Multi-tenancy and fault isolation**: MNOs benefits from sharing their infrastructure by maximizing resource usage. Multi-tenancy consists of sharing an infrastructure among multiple tenants or clients, each with a subset view of the owner's infrastructure resources (called a slice). As a consequence, fault isolation problems may occur [24]. Moreover, to allow a rapid notification of tenants faults, the infrastructure owners should identify the affected tenant entities to enable tenants to perform recovery actions. Figure 3 illustrates the multi-tenant fault isolation issue. In this example, a hardware server crash affects two VNFs (i.e. VNF1 and 2) of tenant 1 and 3 respectively. In this case, a rapid isolation of faults and notification of the two tenants is crucial to provide necessary recovery actions.

- **Ambiguity and consistency of data**: In the detection and localization process, events in the form of logs or metrics are collected to identify the failures. These events contain important information about the health and operations of the system. However, the collected data originate from distinct sources with different formats and are most of the time ambiguous and full of insignificant information for the diagnosis of faults. Therefore, an efficient storing and extraction methods are necessary to prepare data before starting the localization process.

4 LUMEN Framework

In this section, we discuss the scope of our proposed LUMEN framework and describe the functions of each plane.

Table 1: LUMEN proposed solutions for the NVEs issues and challenges

Issues	LUMEN Solutions
Number and type of managed objects	- Data filtering before storing to reduce insignificant data. -Efficient storing engine and clustering methods to face the scalability. (Plane 1, 2)
Dynamic network topology	- Network topology changes can be extracted from collected data. (Plane 1, 2, 3)
Lack of network visibility	- The centralization of logs from different distributed infrastructures. (Plane 2)
Multi-tenant fault isolation.	-Additional fields (e.g. tenant ID). -Efficient Self-diagnosis to notify clients. (Plane 1 ,2,3 and 4)
Ambiguity and consistency of data	- Unification of data format. (Plane 1, 2 and 3)

Table 2: A qualitative comparison of fault management frameworks

	Environments	Data filtering	Data type	Fault Management steps	Decision	Dynamic network topology	Multi-tenancy
LUMEN	NVEs	yes	- logs, - metrics, - topology	Fault management	Bayesian networks	yes	Centralization of tenants data
ISF Framework	End-to-end Ethernet services	yes	- logs, - metrics, - topology	Fault management and Service-Level Monitoring	Petri nets	no	No resources sharing
Vitrage OpenStack	OpenStack environments	yes	- logs, - metrics, - topology	RCA : detection and localization	Templates	Yes	OpenStack tenants

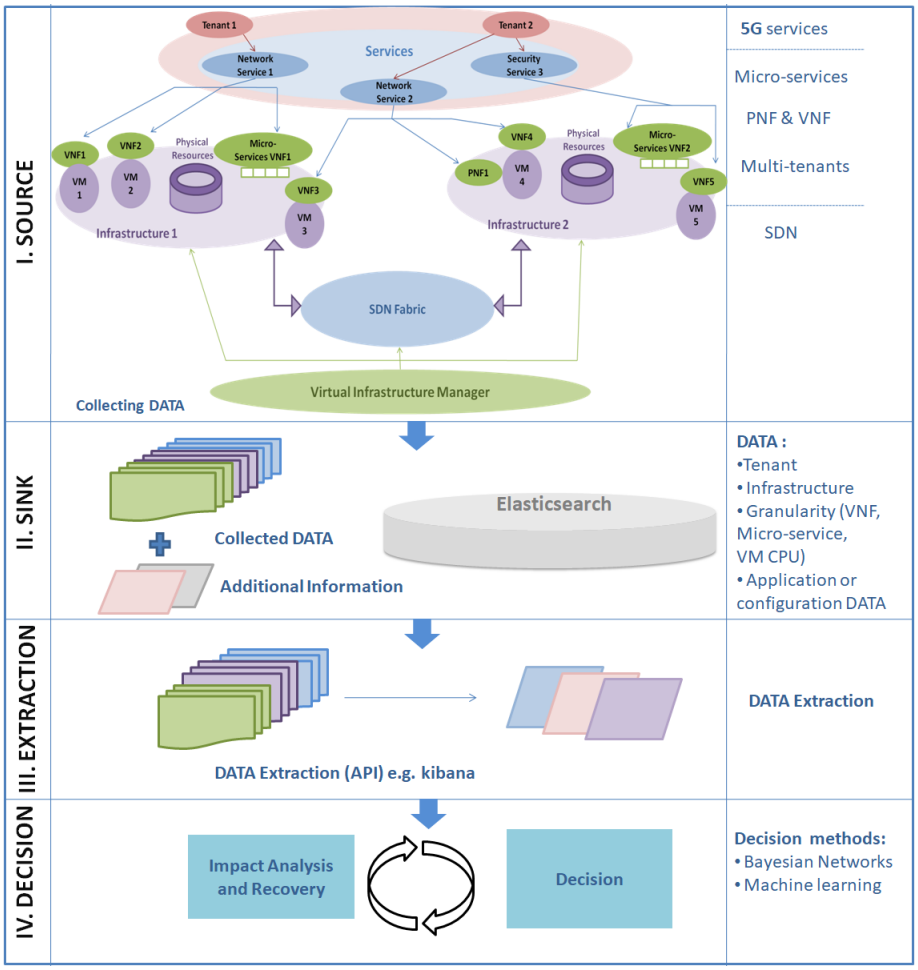

Figure 4: LUMEN Framework

4.1 Overview of the LUMEN Framework

In order to address fault management NVE challenges, we propose LUMEN: a Global Fault Management Framework. LUMEN (Figure 4) is a four step architecture where each plane summarizes the methods that should be deployed to address the different NVEs challenges of Table 1.

The LUMEN framework leverages an open source tool, namely the Elastic Stack [4], to address fault management challenges. The LUMEN framework can also integrate different tools at each layer if necessary. The Elastic Stack ensures a real time data collection, storage, search, analysis, and visualization. The Elastic Stack allows the centralization of logs collected from different locations. This feature is well suited for a multi-tenant environment, where resources of the same tenant can be located on different infrastructures. The Elastic Stack is composed of three tools: Logstash/Beats, Elasticsearch and Kibana. Beats and Logstash are used for collecting data. Beats are lightweight data shippers that can be installed as dedicated agents on managed entities to send specific types of operational data. Logstash allows a more large data collection and enables filtering, enriching, and transforming data from a variety of sources. Elasticsearch is a search engine and analytics NoSQL database designed for storing efficiently the gathered data. Finally, Kibana is used for the data extraction and visualization.

The way LUMEN was designed shows the importance of data in software networks [11]. Information about the network health and the topology changes are gathered in LUMEN. However,

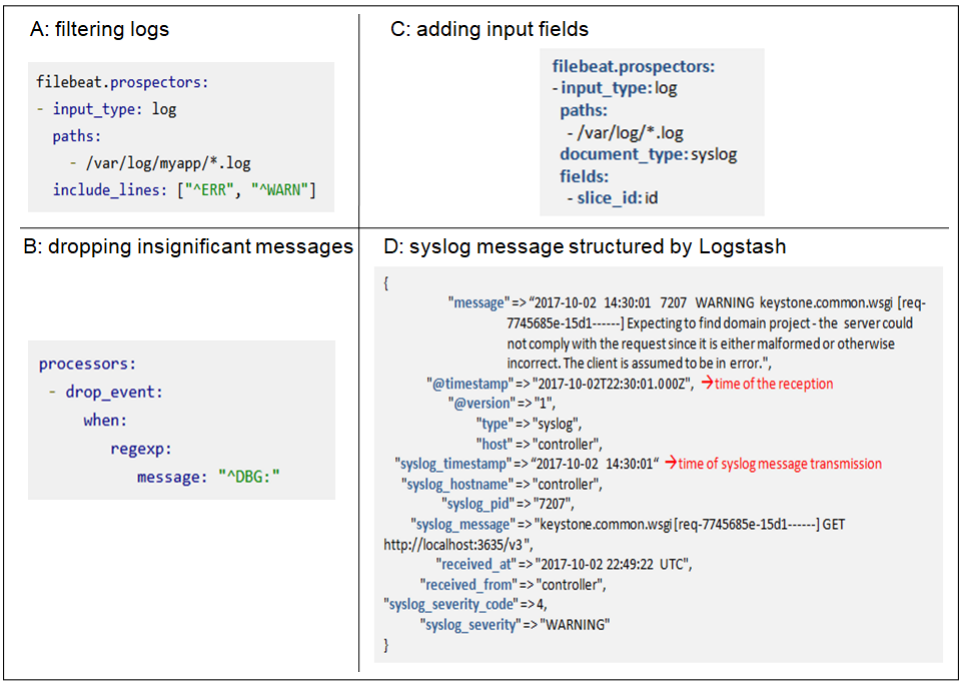

Figure 5: Elastic Stack Logs transformation

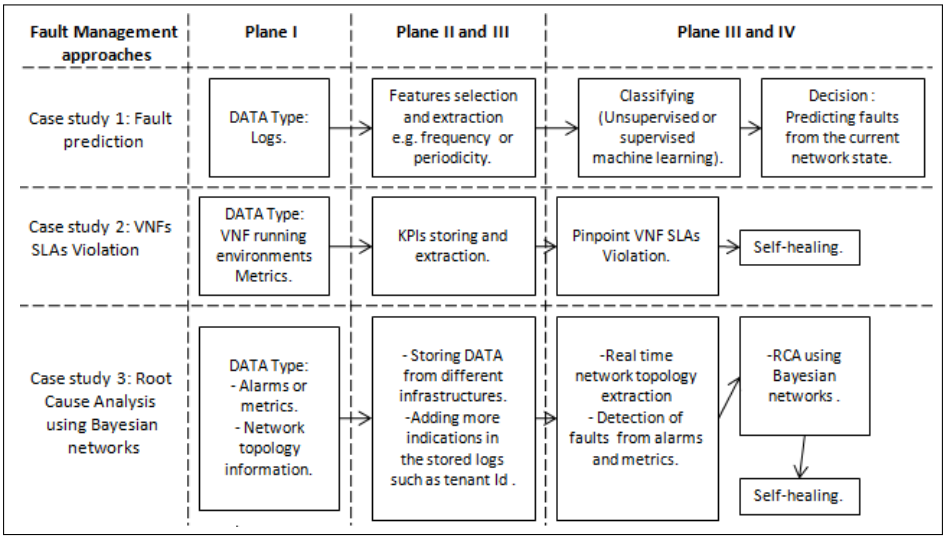

Figure 6: Case studies applied to the LUMEN Framework

this data is frequently mixed with noise from insignificant messages and unrelated events for the fault diagnosis process. For instance, only syslog messages that have severity from 0 to 4 (0: emergency, 1: alert, 2: critical, 3: error and 4: warning) are considered for the RCA process. LUMEN is a four-layer framework that presents efficient techniques to organize and prepare data for the deductions methods.

4.2 LUMEN Planes

The functions of the LUMEN framework planes are described in the following.

Source Plane: The first step of our framework consists of gathering all types of data from distinct entities and distributed locations. As an example, the Beats agents of the Elastic Stack can be deployed in every network entity to send real time alarms and metrics to Elasticsearch. The gathered data depend on the deduction method and the process that will be deployed in the decision plane. For instance, one way to answer the problem of dynamic topology for the model-based technique is to model the network entities dependencies using the real time network topology information extracted from data. The network topology information can be found in SDN controllers or Virtual Infrastructure Manager (VIMs) network modules like OpenStack Neutron [2]. Monitoring data such as logs and alarms are also an important source of inputs. They generally contain relevant hidden information about the network state, faults and root causes.

The collected data can originate from different management levels with distinct types and formats and depend on:

- the management entity: if the collected data concern a specific tenant or the whole infrastructure.
- the granularity: if the collected data concern virtual or physical entities, VMs or containers or sub-components (VM CPU or Network interface).
- the type of data: if the collected data is syslog alarms, metrics or topology information; application or configuration logs.
- the decision method: if the collected data is for RCA or SLA or fault prediction.

Sink Plane: Since the collected data originate from different sources, entities and technologies, the data will have different formats (e.g. syslog , JSON or CSV). Therefore, a unification of the data format and the organization of data is important to facilitate the next steps of fault management. Moreover, the data should be filtered and the insignificant messages should be dropped.

Figure 5-D presents an example of a controller syslog message structured by Logstash in a JSON format. The example illustrates how Logstash can unify messages from disparate sources and normalize the structure of the message before storing it in the Sink. While Figure 5-A and 5-B exposes how Filebeat configuration files enable message filtering and dropping irrelevant information, respectively. Adding supplementary indications before the storing process is also crucial to localize the tenants and the network slices in further fault management steps. Filebeat enables transforming the data with adding new fields (Figure 5-C).

After the data transformation process is completed, the collected data can be stored in a Sink, Elasticsearch in our case. Elasticsearch is a robust search engine to centralize data and enable efficient extraction. Elasticsearch can also be distributed in different infrastructures.

Extraction Plane: This step is highly dependent of the decision process. In fact, the extraction of data depends on the inference engine used in the diagnosis process and the decision methods. For instance, some KPIs (VM cpu or disk load) can be mined to calculate VNF SLAs violation. Application Programming Interfaces (APIs) of the Sink plane can be used for the data extraction process.

Decision Plane: The last step of LUMEN is the decision plane which is the most important step in our fault management framework. In this step we create an educated guess through one or more deduction methods and diagnosis approaches such as Bayesian network and machine learning techniques. This step is enabled by the first three planes that provide the necessary information in a unified and organized way for a real time, rapid and efficient decision process.

5 Discussion and Analysis

We here discuss the use of the LUMEN, ISF [33], and Vitrage OpenStack [3] frameworks applied to three case studies and give a qualitative analysis.

LUMEN enables an autonomic collection and organization of data to prepare the fault management decision process. Plane 1, 2 and 3 represent the detection step in fault management, while Plane 4 represents the localization and recovery step. Further, Self-X frameworks such as the Self-healing framework in [28] can be integrated in LUMEN. Table 1 presents how LUMEN planes addresses the issues of NVEs discussed in Section 3. Figure 6 illustrates how LUMEN can be customized to respond to fault management needs, in this figure three case studies are presented:

- case study 1: fault prediction using machine learning in logs in a time slot.
- case study 2: detection of SLA violation with collecting real time VNFs metrics.
- case study 3: RCA of dynamic multi-tenant environment using Bayesian networks.

5.1 Qualitative Comparison of Fault Management Frameworks

Table 2 presents a qualitative comparison of the three fault management frameworks: LUMEN, ISF [33] and Vitrage [3]. The three architectures are compared with different parameters related to the chosen environment and the methods and steps used in fault management. We focused on the LUMEN case study 3 in this analysis.

The conception of LUMEN and the choice of the opensource tools was made to accept distinct physical and virtual resources distributed in different infrastructures. Vitrage is designed for OpenStack environments, while the ISF framework addresses only Ethernet networks.

The three architectures filter data before processing, but only LUMEN proposes to augment the consistency of data with additional indications such as slice ID. Both LUMEN and Vitrage consider multiple tenants, while in ISF there is no concept of vitalization and network sharing in the considered environment.

The three frameworks consider the information about network topology and entities dependencies. A topology database was defined in ISF framework that stores three types of information: the node address, the node functionality and a list of connecting nodes. However, no topology builder was defined in the ISF framework since the end-to-end Ethernet services has a fixed topology, i.e. only new added entities to the network can be detected but not real time changes such as in the case of an on-demand NFV chain service. In Vitrage an entity graph builds the network topology using the information collected locally in the OpenStack modules. While in LUMEN we consider two sources to collect topology information: the VIM modules and SDN controllers.

The decision methods should be adapted to the scalability and dynamicity of NVEs. Templates or rule based techniques are used for static networks with the presence of expert knowledge . This knowledge is then encoded in rules. Acquiring this knowledge in a dynamic network such as NFV is difficult. Petri nets are dynamic systems that can only encode sequences of events. To progress towards a more ambitious diagnosis, one must adopt more appropriate methods such as Bayesian networks. The later model the network using constraints and statistical dependencies. They provide explanations about failure propagation, deal with multiple failures and capture false or lost alarms. Still, the definition of the model in a scalable and dynamic network such as NVEs is a difficult step.

The Vitrage project was developed to detect the root cause of problems in an OpenStack running environment and notify clients. The defined architecture focuses on the detection and localization of faults. The ISF framework addresses the different management steps with an integration of a service-level monitoring module. The module measures the service performance and compare these measurements with the requirements set in the SLA.

In the case of fault management, LUMEN considers all the steps and is open to the different NVEs. However, the decision module needs further investigation which is one of our future goals.

6 Conclusion and future work

The 5G deployment consists on a variety of NVEs: SDN, NFV, VNF chains and multi-tenant environments. The coexistence of this distinct and complementary NVEs models will open up new fault management challenges and issues where classical fault management methods are limited.

In this paper, we presented significant issues to be considered in the fault management of NVEs. Such as the dynamic network topology and the large number and variety of managed entities. These issues weaken diagnosis solutions that were designed for limited and fixed network topologies. We referenced relevant studies illustrating this fact.

We highlighted the necessity to provide automation to the canonical fault management steps and proposed LUMEN: a Global Fault Management Framework. LUMEN summarizes the fault management steps in four planes (Source, Sink, Extraction and Decision). The first three planes compose the detection step. In these three phases, the data is remodeled to prepare the decision plane where deduction methods can be deployed. In our future work we will focus in the Decision Plane. Some of the planned activities relate to the third example of Figure 6.

To implement LUMEN, we plan on using OpenStack Ocata and the Docker-engine as a Virtual Infrastructure Manager (VIM), The Clearwater virtual IMS with docker and virtual machines [7] for the virtual environment and R or Python language for the RCA inference engine.

Our next steps consist on remodeling the Bayesian network technique to pinpoint faulty network components in NVEs. Still, some open issues need to be investigated for our future contribution. We will focus on the following points: the problem of scalability of Bayesian networks that was addressed by [19] for IMS networks, introduce VNF chains in the managed entities plus the SDN model that was addressed by [27] with a finer granularity. This case study should not affect the diagnosis reliability, address in more details the problem of tenants fault isolation and propose a real time Self-diagnosis engine for NVEs.

Acknowledgment

Authors would like to thank their colleague, Ayoub Bousselmi, for his early review of the paper.

References

[1] Telecom infra project (tip), https://telecominfraproject.com. Accessed on: 18/09/2017.

[2] Openstack neutron, https://wiki.openstack.org/wiki/Neutron. Accessed on: 03/09/2017.

[3] Openstack Vitrage, https://wiki.openstack.org/wiki/Vitrage. Accessed on: 02/10/2017.

[4] Elastic stack, https://www.elastic.co/products/elasticsearch. Accessed on: 03/09/2017.

[5] Open network automation platform (onap), https://www.onap.org/. Accessed on: 18/09/2017.

[6] Openstack USER SURVEY, https://www.openstack.org/assets/survey/April2017SurveyReport.pdf. Accessed on: 17/09/2017.

[7] Project clearwater, http://www.projectclearwater.org/. Accessed on: 18/09/2017.

[8] R. Abhishek, S. Zhao, S. Song, B. Y. Choi, H. Zhu, and D. Medhi. Buddi: Bug detection, debugging, and isolation middlebox for software-defined network controllers. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 307–311, Oct 2016.

[9] L. Bennacer, L. Ciavaglia, A. Chibani, Y. Amirat, and A. Mellouk. Optimization of fault diagnosis based on the combination of bayesian networks and case-based reasoning. In *2012 IEEE Network Operations and Management Symposium*, pages 619–622, April 2012.

[10] Jeffrey D Case, Mark Fedor, Martin L Schoffstall, and James Davin. Simple network management protocol (snmp). Technical report, 1990.

[11] 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*, 30(1):58–65, January 2016.

[12] R. P. Esteves, L. Z. Granville, and R. Boutaba. On the management of virtual networks. *IEEE Communications Magazine*, 51(7):80–88, 2013.

[13] GSNFV ETSI. Network functions virtualisation (nfv); use cases. *V1*, 1:2013–10, 2013.

[14] NFVGS ETSI. Network functions virtualisation (nfv); management and orchestration. *NFV-MAN*, 1:v0, 2014.

[15] Lv Feng, Li Xiang, and Wang Xiu-qing. A survey of intelligent network fault diagnosis technology. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 4874–4879. IEEE, 2013.

[16] Rainer Gerhards. The Syslog Protocol. RFC 5424, March 2009.

[17] W Haeffner, J Napper, M Stiemerling, D Lopez, and J Uttaro. Service function chaining use cases in mobile networks. *Internet Engineering Task Force*, 2016.

[18] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, Feb 2015.

[19] C. Hounkonnou and E. Fabre. Empowering self-diagnosis with self-modeling. In *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, pages 364–370, Oct 2012.

[20] Tatsuaki Kimura, Akio Watanabe, Tsuyoshi Toyono, and Keisuke Ishibashi. Proactive failure detection learning generation patterns of large-scale network logs. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 8–14. IEEE, 2015.

[21] Ma lgorzata Steinder and Adarshpal S. Sethi. A survey of fault localization techniques in computer networks. *Science of Computer Programming*, 53(2):165 – 194, 2004. Topics in System Administration.

[22] ITU-T Recommendation M.3010. Principles for a telecommunications management network, 2000.

[23] Antonio Manzalini, Cagatay Buyukkoc, Prosper Chemouil, Slawomir Kuklinski, Franco Callegati, Alex Galis, Marie Paule Odini, Chih-Lin I, Noel Crespi, Eileen Healy, and Stuart Sharrock. Towards 5g software-defined ecosystems. 07 2016.

[24] V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle. A survey of network isolation solutions for multi-tenant data centers. *IEEE Communications Surveys Tutorials*, 18(4):2787–2821, Fourthquarter 2016.

[25] The Linux Foundation Projects. Open platform for nfv (opnfv), 2017.

[26] El Hattachi Rachid, Erfanian Javan, and 5G Initiative Team. Ngmn alliance 5g white paper; reliability; report on models and features for end-to-end reliability, 2015.

[27] J. M. Sanchez, I. Grida Ben Yahia, and N. Crespi. Self-modeling based diagnosis of software-defined networks. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–6, April 2015.

[28] José Manuel Sanchez Vilchez, Imen Grida Ben Yahia, Noël Crespi, Tinku Rasheed, and Domenico Siracusa. Softwarized 5G Networks Resiliency with Self-Healing. In *5GU - 1st International Conference on 5G for Ubiquitous Connectivity*, Levi, Finland, Finland, November 2014.

[29] C. Sauvanaud, K. Lazri, M. Kaâniche, and K. Kanoun. Anomaly detection and root cause localization in virtual network functions. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 196–206, Oct 2016.

[30] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. A link failure recovery algorithm for virtual network function chaining. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 213–221. IEEE, 2017.

[31] M. Steinder and A. S. Sethi. Probabilistic fault localization in communication systems using belief networks. *IEEE/ACM Transactions on Networking*, 12(5):809–822, Oct 2004.

[32] S. R. Tembo, J. L. Courant, and S. Vaton. A 3-layered self-reconfigurable generic model for self-diagnosis of telecommunication networks. In *2015 SAI Intelligent Systems Conference (IntelliSys)*, pages 25–34, Nov 2015.

[33] 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*, 4(1):28–38, June 2007.

[34] J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308 – 320, 2013.

[35] G. Zhaojun and W. Chao. Statistic and analysis for host-based syslog. In *2010 Second International Workshop on Education Technology and Computer Science*, volume 2, pages 277–280, March 2010.

[36] Z. Zheng, Z. Lan, B. H. Park, and A. Geist. System log pre-processing to improve failure prediction. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pages 572–577, 2009.