Dependability of the NFV Orchestrator: State of the Art and Research Challenges

Andres J. Gonzalez, Gianfranco Nencioni, Andrzej Kamisiński, Bjarne E. Helvik, and Poul E. Heegaard

***Abstract*—The introduction of Network Function Virtualisation (NFV) represents a significant change in networking technology, which may create new opportunities in terms of cost efficiency, operations, and service provisioning. Although not explicitly stated as an objective, the dependability of the services provided using this technology should be at least as good as conventional solutions. Logical centralisation, off-the-shelf computing platforms, and increased system complexity represent new dependability challenges relative to the state of the art. The core function of the network, with respect to failure and service management, is orchestration. The failure and misoperation of the NFV Orchestrator (NFVO) will have huge network-wide consequences. At the same time, NFVO is vulnerable to overload and design faults.**

Thus, the objective of this paper is to give a tutorial on the dependability challenges of the NFVO, and to give insight into the required future research. This paper provides necessary background information, reviews the available literature, outlines the proposed solutions, and identifies some design and research problems that must be addressed.

***Index Terms*—NFV, NFV Orchestrator, NFV MANO, Fault Tolerance, MANO Resilience, NFV Reliability, NFV Dependability, Cloud Computing.**

I. Introduction

Network Functions Virtualisation (NFV) represents a change of generation in networking and in the provisioning of network-based services. NFV is part of the "softwarisation" trend that includes Software-Defined Networking (SDN). It is a networking paradigm based on the idea of programmable network devices where the forwarding plane is decoupled from a logically-centralised control plane. NFV is expected to yield significant benefits, such as increased flexibility, faster introduction of new and revised functionalities, better utilisation of network resources, the ability to tailor network functionality and QoS to specific application requirements more effectively, as well as reduced capital expenses (CAPEX) and operational expenses (OPEX) [1]–[3]. However, there are significant challenges associated with this change. One of the most significant challenges is the dependability level of NFV in comparison to current networking technology [4]–[7].

A. J. Gonzalez is with Telenor ASA, Tromsø, Norway (e-mail: andres.gonzalez@telenor.com).

G. Nencioni was with NTNU – Norwegian University of Science and Technology, Trondheim, Norway. He is now with the University of Stavanger, Stavanger, Norway (e-mail: gianfranco.nencioni@ntnu.no; gianfranco.nencioni@uis.no).

A. Kamisiński is with AGH University of Science and Technology, Kraków, Poland (e-mail: andrzejk@agh.edu.pl).

B. E. Helvik and P.E. Heegaard are with NTNU – Norwegian University of Science and Technology, Trondheim, Norway (e-mail: {bjarne.e.helvik, poul.heegaard}@ntnu.no).

Manuscript received XXXXX; revised YYYYY.

A core component in this context is the NFV Orchestrator (NFVO), as defined in the general architecture specification proposed by ETSI [1]. The NFVO maintains a global view of the state of the network and provided services, and it manages the available resources to provide optimal service. Hence, it may be used to deal with failures of network elements efficiently, and thereby improve service dependability. On the other hand, since the NFVO maintains a global view of the NFV system, i.e., it is a logically-centralised entity, it may affect the entire network as a result of a misoperation, while errors due to physical and logical faults may have severe impact on the provisioning of network services. Similarly, the VIM is a locally-centralised entity and its failures may have an extensive impact on the NFV system.

Dependability is fundamental for making NFV a reality; however, the NFVO is a potential dependability bottleneck if it is not well planned and designed. The objective of this paper is to provide a tutorial on the dependability challenges of the NFVO, and to give a clear insight into further research concerning the key related issues that must be addressed. To achieve this goal, we survey the relevant papers and standards specifically related to this area. To provide the necessary context for the reader, we also include the selected additional references and we discuss the issues related to the dependability of the entire NFV system, but we do not cover them in the same detail. The steps toward the objective are as follows: (i) identify in detail the key dependability challenges, (ii) present the state of the art of the NFVO from the dependability point of view, and (iii) highlight the issues that need to be addressed to make NFV sufficiently dependable.

The next section gives an introduction to the necessary background concepts and definitions associated with the NFV architecture proposed by ETSI [1]. The related dependability taxonomy is presented in Section III, while Section IV identifies overall challenges related to having a dependable NFV system. The relevant monitoring and failure recovery concepts are presented in Section V. Section VI provides a more detailed discussion of the functionalities, requirements, and architecture of the NFVO. In Section VII, we detail the impact of the NFVO on service dependability. Different possible options to make the NFVO fault-tolerant are proposed in Section VIII, before we conclude the paper by summarising the identified challenges in Section IX.

II. NFV-MANO Concepts and Definitions

NFV Management and Orchestration (NFV-MANO or, in short, MANO) has been defined by the NFV ETSI Industry

TABLE I
LIST OF ACRONYMS IN THIS PAPER

BSS	Business Support System
CP	Connection Point
DPDK	Data Plane Development Kit
EMS	Element Management System
ISG	Industry Specification Group
KPI	Key Performance Indicator
NCT	Network Connection Topology
NFV	Network Functions Virtualisation
NFV-MANO	NFV Management and Orchestration
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NS	Network Service
OSS	Operations Support System
PNF	Physical Network Function
SDN	Software-Defined Networking
VIM	Virtualised Infrastructure Manager
VL	Virtual Link
VNF	Virtualised Network Function
VNFFG	VNF Forwarding Graph
VNFM	VNF Manager

Specification Group (ISG) in [8] as a key element in the effective provisioning and management of Network Services (NSs), Virtualised Network Functions (VNFs), and the underlying infrastructure. This section provides the description of the relevant NFV and MANO concepts. First, in Section II-A, we introduce the fundamental NFV concepts. Second, we identify and describe the key components of the MANO in Section II-B. Finally, Section II-C summarises the main functionalities assigned to each particular MANO component.

A. NFV-ETSI General Architectural Concepts

Based on [8], Figure 1 presents a high-level representation of NFV architecture, emphasising its key components and their interconnection scheme.

According to the ETSI proposal, every NFV system requires access to hardware resources, such as computing, storage, and the network. The virtualisation layer and its respective virtualised resources are catalogued and aggregated by the NFV Infrastructure (NFVI) subsystem to enable them to be dynamically assigned to VNFs, according to specific demands. Management of the entire infrastructure is performed by the Virtualised Infrastructure Manager (VIM) component. Careful planning of interactions between the virtualised environment and Physical Network Functions (PNFs) is required, and the PNFs must be accessible to the NSs that depend on them.

The VNF (see Figure 1) represents the software implementation of all offered network functions that are decoupled from the hardware resources they use. They gain access to the respective resource pools via the NFVI subsystem, as shown in Figure 1. VNFs are managed locally by the Element Management System (EMS) and globally by the VNF Manager (VNFM) component.

Dynamic management of the system relies on automation enforced by policies with a set of specific conditions and their corresponding execution. For this reason, the related management and orchestration actions are also needed. In the ETSI-NFV architecture shown in Figure 1, dynamic management is represented by the NFV-MANO subsystem. Its

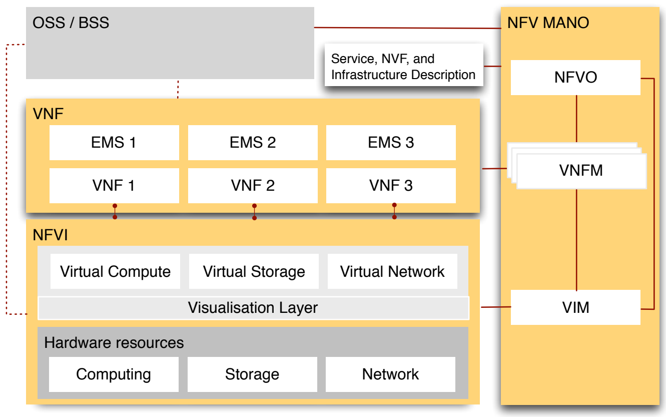

Fig. 1. ETSI NFV ISG architectural framework [1]

respective components and functionalities will be explained in the following sections.

NFV deployments are expected to involve multi-vendor ecosystems and different lifecycles. Thus, to make flexible management and integration with existing systems possible, standardised and consistent interfaces are required. In addition, the components should support monitoring, testing, and fault management actions, as identified in [8]. For instance, monitoring tasks may include the estimation of Key Performance Indicators (KPIs) of NSs and the detection and localization of faults, while fault management may involve such actions as fault correlation, root-cause analysis, isolation, and recovery.

B. MANO Components

The NFV-MANO subsystem, as proposed in [9], is composed of three main components: the NFVO, the VNFM, and the VIM.

The NFVO is at the highest hierarchical level of the NFV-MANO and it is responsible for the creation and lifecycle management of NSs, as well as the respective validation, authorisation and management of the necessary infrastructure. The VNFM configures and supervises the lifecycle of VNFs and performs the respective coordination and adaptation role for configuration and event reporting between the VIM and the EMS [9]. Finally, the VIM controls and manages the NFVI computation-, storage-, and network-related resources to create and assign the virtual resources needed by specific functions.

MANO components are critical to the operation of an NFV system. They are interconnected and work together to provide the adequate functionalities and life-cycle management of both physical and virtualised resources.

C. MANO Functionalities

The MANO functionalities required by an NFV system are introduced in the ETSI NFV-MANO specification [8] for different managed targets.

The first managed target is the NFVI. The main MANO functionalities in this case are focused on the accessibility, availability, allocation, and release of virtualised resources,

TABLE II
SUMMARY OF MANO COMPONENTS AND FUNCTIONALITIES

Managed Target	MANO Functionalities	Responsible
NFVI	(i) Manage the accessibility, availability, allocation, and release of virtualised resources (ii) Fault and performance management of virtualised resources and hardware	VIM
VNF	(i) VNFs Lifecycle Management (ii) Fault, Performance, Security Management (iii) Configuration and Accounting Management	VNFM
NS	(i) Registering, instantiating, scaling, updating, terminating NSs (ii) Creating, deleting, querying, and updating the corresponding VNFFGs	NFVO
General	(i) Fault, Performance, and Policy Management (ii) Performance, operational, and functional testing of NSs	VIM, VNFM, NFVO

as well as fault and performance management of virtualised resources and respective hardware.

VNFs represent the second managed target of MANO functionalities. The actual VNFs are decoupled from the physical infrastructure. Thus, it is required that additional features be deployed in MANO to enable logically-centralised management. The main VNF-related MANO functionalities are the configuration, fault, performance, security, lifecycle, and accounting management of VNFs.

NSs are managed targets that demand additional MANO functionalities. Usually, NSs are planned to be executed by the orchestration subsystem. In particular, the related MANO functionalities in this case are as follows: create, register, scale up/down, update, and terminate NSs, as well as create, delete, query, and update the corresponding VNF Forwarding Graphs (VNFFGs).

Finally, MANO performs general management functionalities on different levels in the context of virtualised resources, VNFs, and NSs, such as fault, performance, and policy management.

Table II presents a summary of the components and functionalities described in this section. For a detailed description of particular functionalities, the reader is referred to [8].

III. RELATED DEPENDABILITY TAXONOMY

We use the term *dependability* as defined in [10] as the overall property of NFV that is addressed in this paper.

> *Dependability* is the *trustworthiness of a system such that reliance can justifiably be placed on the service it delivers.*

This term is a general term that does not specifically focus on the system's ability to withstand or recover from failures. Other terms commonly used to describe an overall property are more specific, including *resilience* (ability to *recover* back to normal operation) and *robustness* (ability to *tolerate* misbehaviour and failures). For a more elaborate discussion,

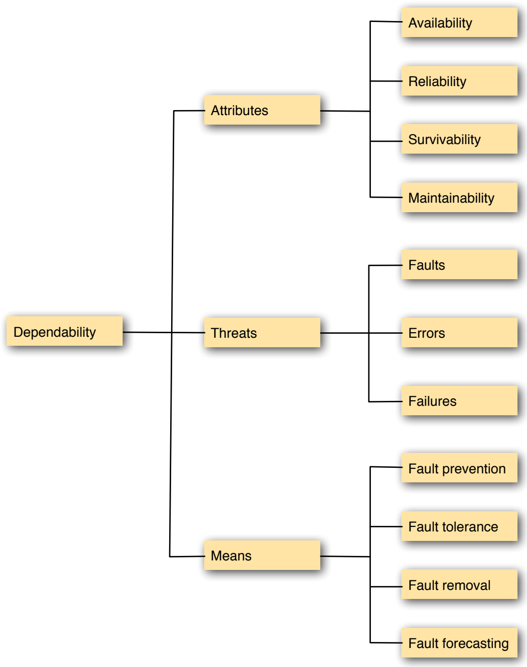

Fig. 2. The dependability tree [10] (revised to include the attributes considered as the most important for the study in this paper).

visit [11]. The taxonomy used to specify the dependability tree is given in Figure 2, revised to include the attributes that are considered the most important.

A. Attributes

Specific attributes are necessary to reflect the properties that are important to characterize and measure of specific NFV services, including the following:

- *Availability* — readiness for correct service [10], i.e., delivery of service in compliance with the service specification; measured as the *probability of the readiness to provide service compliant with the requirements*, for example, service new demand;
- *Partial availabilty* — delivery of service in compliance with a subset of the requirements, or to a subset of users;
- *Reliability* — continuity of correct service [10], i.e., continuity of service in compliance with the service specification; measured as the *probability of the continuity of service compliant with the requirements*, for example, providing service for the required duration and then terminating;
- *Survivability* — system's ability to continuously deliver services in compliance with the given requirements in the presence of failures and other undesired events; several measures related to the recovery phase are defined [12];
- *Maintainability* — ability to undergo modifications and repairs [10], i.e., the ease with which maintenance of a functional unit can be performed in accordance with prescribed requirements (the definition includes both proactive and reaction actions).

In this paper we focus on NFV network services. A service is regarded as degraded (partially down) if x% of the n users do

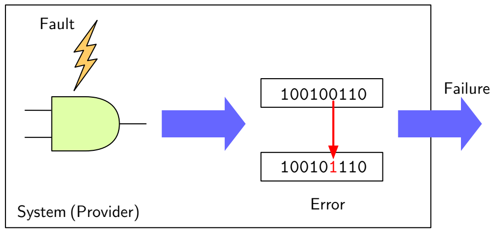

Fig. 3. Relation between fault, error, and failure.

not receive a service that is compliant with its specification. When $x < \theta$ (θ is a service dependent threshold), then the service is down. Partially available means that the service is x%-available.

B. Threats

All attributes and metrics above seek to characterise the dependability of the system and the services it provides. The properties are affected by many factors, including environmental, fault-tolerant design (or lack of), quality of equipment and design, and operation of maintenance. In order to understand the root causes, lack if consistency or logic in the inner workings of the system, and the visible consequences, we will use the following "pathology" [10]:

- *Fault* — adjudged or hypothesized cause of an error;
- *Error* — part of the system state which is liable to lead to a failure;
- *Failure* — deviation of the delivered service from compliance with the specification; transition from correct service to incorrect service (e.g., the service becomes unavailable).

The relation between these concepts is illustrated in Figure 3 where an external fault flips a bit in the memory causing an error that manifests as a failure when that partition of the memory is accessed. This pathology might be used in a recursive manner, meaning that a failure at one level (or in a subsystem) might be a fault on the next level (or for the system). In the NFV context where a service consists of a chain of service components (see Section VII-A), a failure in a service component is a fault of the services that rely on this component.

C. Faults

Different root causes (faults) of a failure exists, and the following classification is used in this paper:

- *Physical faults* — "classical" hardware faults;
- *Transient faults* — faults that are present only for a short period of time showing no physical change in the system;
- *Intermittent (sporadic) faults* — faults that come and go;
- *Design (logical) faults* — human-made faults during specification, design, and implementation;
- *Interaction or operational faults* — accidental faults made by humans that are operating or maintaining a system;
- *Faults caused by the environment* — faults from outside the system boundary;
- *Excessive load* — faults due to loads above the system capacity that last for a short or long period;
- *Malicious attack and intrusion* — faults that are caused by planned, malicious activity to harm data, systems or services.

D. Failure semantics

When a failure occurs in a distributed processing system such as NFV, the consequences on the services provided by a functional unit can be classified by the following commonly accepted *failure semantics*:

- *Omission failure* (crash failure) — does not respond to an input;
- *Value failure* — responds to an input within the correct time, but with incorrect value;
- *Timing failure* — responds to an input with the correct value, but not within the correct time;
- *Arbitrary failure* (Byzantine failure) — does not respond to an input within the correct time or with incorrect value (or both).

The same classification also applies to the network services provided by the system that consists of a set of sub-systems or functional units.

E. Recovery

After a failure in the system, services must *recover*, which means *return (component/system/service) to its original condition*. In this paper we distinguish between:

- *Repair* — fix or mend a component that is suffering from damage or a fault (including restart, reboot, upgrade of software);
- *Replacement* — change a failed component with a new (working) one (including reinstallation of software).

It is also important that recovery occurs in several consecutive stages, and in this paper we use:

1) *Detection* — a failure that has occurred must be detected, so that further actions can be taken; depends on proper monitoring of the system;
2) *Localization* — determine where the failure is;
3) *Isolation* — if necessary: isolate the failure to prevent it from propagating or escalating;
4) *Repair/Replace* — decide a proper action to restore the service by either repair or replacement.

These stages can be quantified by survivability, and are related to the maintainability.

IV. NFV Dependability Challenges

The virtualisation of network functions represents a generational change in network technology, which also represents a significant change in the related challenges of providing highly dependable services. Several real-world experiments are needed to measure, improve, and understand better the consequences and implications of such scenarios. In addition, an underlying change in threat scenarios is expected following the shift in technology, which imposes a different, and maybe less acceptable, risk profile for NFV-based networks. These issues are discussed in the following subsections. To help the reader understand some of the challenges concerning fault tollerance in NFV, a brief comparison of the dependency structure in a conventional network architecture and an NFV architecture is made.

A. Strengths and Vulnerabilities of NFV

In the current network, network functions are provided by proprietary vendor-specific implementations integrated with function specific hardware in dedicated network elements. This design has proved to be reasonably robust, and the relatively loose coupling between network elements has limited the network-wide outages. In the virtualised network, the functions will be provided by a logical centralised control and management. This has the potential to improve network dependability, but also introduces new threats and augments existing threats. Before we look at these, let's summarise some of the expected benefits of the virtualisation of network functions:

- The global overview of the state of the network and all its components combined with an increased control of resources may provide a better handling of faults of dedicated equipment.
- In current networks, functions are provided by dedicated components. When the control in these components fails, even when it is provided by a fault-tolerant platform, a manual intervention may be required to rectify the failure. This may incur a significant time. In a virtualised system, the VNF may be restarted in another virtual machine with minor delay.
- Better monitoring and alarm handling is expected to provide faster identification and handling of faults. However, a potential challenge is alarm storms triggered by a failure of a component causing an excessive number of alarms in the network. New monitoring techniques will provide smarter techniques for identifying fault patterns which can serve as early warnings of a potentially more severe failure.
- In general, a unified and centralised management of all resources in the network by the VIM is expected to yield improved management and increased dependability.

As evident in the above items, NFV has the potential to deal more efficiently with a number of failures in the network. However, the potential dependability-related weaknesses of NFV have received less attention. The list below does not claim to be exhaustive, but it aims to illustrate some fundamental challenges that the introduction of NFV imposes.

- By moving all network functions onto NFVI, control and management become centralised. The NFVI is highly likely to be implemented as a distributed system, but control and management will be logically centralised. This means that the system is far more vulnerable to error propagation between different software elements and across previously isolated functions. Misoperation of a software module may affect other related functionality that depends on shared information. Furthermore, the system becomes far more vulnerable to common cause failures in design and operation in spite of the distributed hardware platform. Hence, the probability of severe network-wide outages is likely to increase.
- For dependability and capacity reasons, the network functions and their management are likely to be executed on a distributed platform, which may cover more physical sites to achieve robustness to environmental failures. The sites may be dedicated computing entities or centres shared with other applications, e.g., the control and management functionality of SDN, and communication application. It is an open issue whether these platforms may provide a dependability similar to that of dedicated legacy network providing systems.
- The flexibility and adaptivity of NFV increase the complexity and chances of fault in design, implementation, configuration, and operation.
- Most distributed computing platforms are designed for a crash failure semantics, i.e., a failure causes the failed component to stop responding. The same behaviour is typically also made for control and management software. With the potential detrimental consequences of a network outage, it should be questioned whether this presupposition should be taken for NFV; especially for the VIM and the NFVO, i.e., whether the system should be designed for a wider range of failure semantics, (see Subsection III-D).
- The shift into VNF will also imply a shift in the context of setting up a network. The "NFV marketplace" will be an ecosystem of network functions, management software, and platform software, some proprietary and some available in the public domain. There is a large number of options for setting up a system, which will introduce a rage of compatibility and interworking issues that may be critical to dependability if they emerge during abnormal situations. As opposed to the "turn key functions" provided by a vendor, there will be no entity that has full insight into the entire software. Also, it is important to keep in mind that providing and maintaining a highly dependable configuration will be the responsibility of the network operator, which in a transition phase may constitute a challenge with respect to the available competence.
- The need to meet the performance requirements in the data plane has pushed the introduction of acceleration modules, such as DPDK, which provide more autonomy in the user space and reduce the load of the kernel space [13], [14]. At the same time, these improvements introduced new dependability dimensions that need to be considered. For instance, DPDK software releases have been undergoing bug-fixing and improvement procedures in the recent years, which increased the implementation awareness on stable and longtime support releases. In addition, challenges such as the respective VNF validation and the live migration implementation when using DPDK still need to be investigated. Currently, there are

several studies with a strong focus on the performance implications of such acceleration modules that need to be complemented with deeper studies on their dependability impact.

B. NFV Experiments on Dependability and Orchestration

Real-world experiments involving NFV increase the awareness of the related limitations and capabilities. There are numerous experiments carried out by different participating actors (operators, vendors, universities, independent researchers, and others), and multiple aspects need to be evaluated. For the sake of illustration, in this section we discuss the selected experiments related to the scope of this paper.

The study in [15] presents a fault injection prototype to analyse the fault resilience of OpenStack. The study resulted in 23 uncovered bugs in two OpenStack versions, and it presented a methodology to perform such analysis. Furthermore, it provided important design principles to build a fault-resilient cloud-management stack. In [16], an experimental analysis of a virtualised IP Multimedia Subsystem (IMS) using the VMware ESXi hypervisor was employed to evaluate and benchmark performance and reliability. Using fault injection, this work studies the impact of faults on VNFs in terms of performance degradation and service unavailability, pointing out the dependability bottlenecks in the NFVI and providing the dependability design policies. A recent study to detect early potential SLA violations due to anomalous virtual machine behaviour is presented in [17]. The authors provide tools to enable service providers to proactively plan for appropriate recovery strategies, using supervised machine learning algorithms and fault injection tools relying on the VMware vSphere 5.1 virtualised platform, and Clearwater, an open source implementation of an IMS for cloud platforms.

The ETSI-NFV [1] has encouraged the development of Proof of Concepts (PoCs) in order to increase the industrial awareness and confidence in NFV, and to help the development of a diverse and open NFV ecosystem, providing feedback on interoperability and other technical challenges. A full list of the current PoCs can be found in [18]. The most relevant PoCs concerning NFV orchestration are presented in [19]–[21], and the PoCs focused on the dependability aspects are documented in [22], [23]. Currently, there are several open projects developing orchestration solutions [24]–[26], and their current status, new features, and evaluation results are presented periodically. A more detailed explanation of the architectural features of those projects will be presented in VI-D.

C. Shift in Risk Profile

The previous section shows the potential of a change in the risk profile, as illustrated in Figure 4. The NFV has the potential to reduce the consequences of "everyday failures" in the network and thereby improve dependability. On the other hand, the system will be more centralised, which limits the inherent robustness towards network-wide outages. As mentioned in previous section, the increased complexity of the control software is expected to increase the likelihood for design, configuration, and operational failures, which in some cases may have severe consequences.

Even if it is feasible to maintain "carrier grade of service" in terms of average service availability towards individual customers, the potential shift in profile may have severe societal consequences.

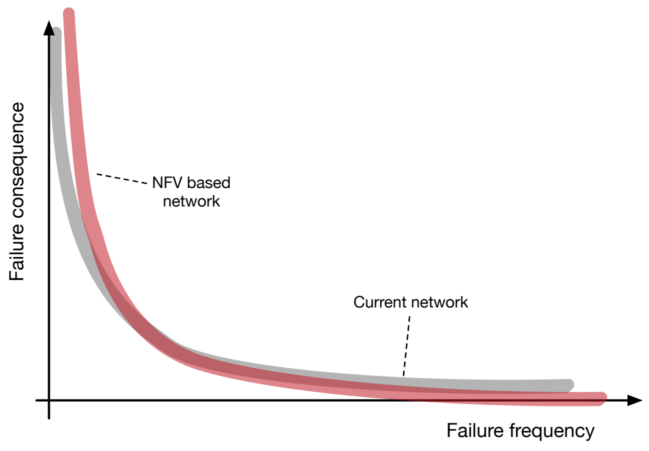

Fig. 4. Shift in risk profile with the introduction of NFV.

This shift in threat scenario and risk profile is most important during the definition of the NFV architecture and the design of systems. The relatively detailed NFV architectural framework [1] is necessary to establish the foundation for a sustainable NFV eco-system. The standardisation work on dependability in NFV ([27], [28], [29]) is tightly coupled to this reference architecture. However, no overall dependability strategy for the systems based on this architecture is addressed. To establish a basis for such a discussion, focusing on the NFVO, a relationship between the different elements of the architecture is established in the next section.

D. Depends-on relations of the NFV architecture

The NFV architectural framework [1] presents the functions in the system and their interworking. The objective of the system is to provide end-user services. The *depends-upon* graphs, introduced by Flavin Cristian [30], are useful means to study relationships between functional elements to understand how failure semantics and fault tolerance may be built into the system while providing end-to-end services. For a well designed system it must be a unidirectional loop-free graph, like for instance a Bayesian network graph. Note, however, that the purpose of these graphs is different; the depends-upon graph is aimed at revealing the structural relations between functions, while the Bayesian Network represents the probabilistic dependencies among events, see for instance [31].

This subsection presents a depends-upon graph for the NFV architectural framework, where the notation is slightly extended as in Figure 5. Network functions represent internal system services, and they are provided from system components, i.e., internal servers.

To illustrate the use of the depends-upon graph and the corresponding symbols, a simple example based on a conventional

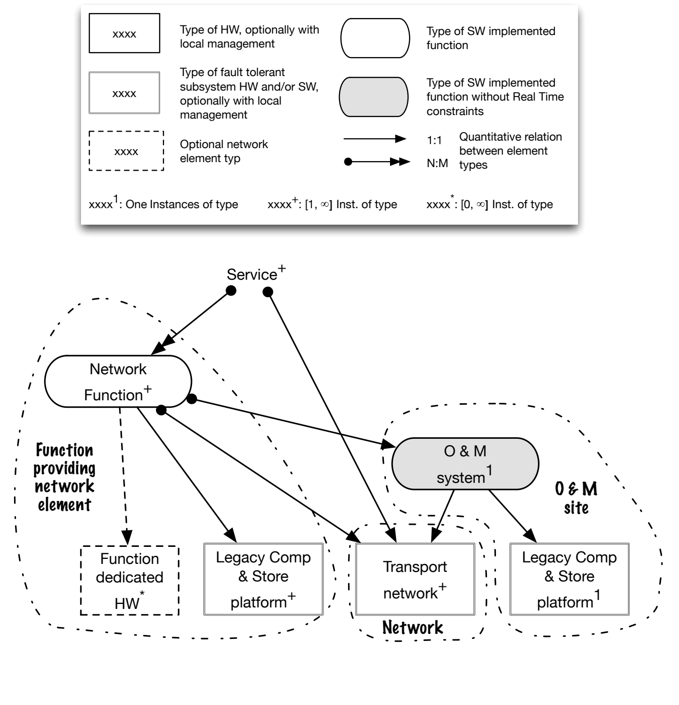
xxxx
Type of HW, optionally with local management
xxxx
Type of fault tolerant subsystem HW and/or SW, optionally with local management
xxxx
Optional network element typ
Type of SW implemented function
Type of SW implemented function without Real Time constraints
1:1
N:M
Quantitative relation between element types
xxxx1: One Instances of type
xxxx+: [1, ∞] Inst. of type
xxxx*: [0, ∞] Inst. of type
Service+
Network Function+
Function providing network element
O & M system1
O & M site
Function dedicated HW*
Legacy Comp & Store platform+
Transport network+
Network
Legacy Comp & Store platform1

Fig. 5. Notation and Example; the depends-upon graph of service provisioning in a conventional (non-NFV) network.

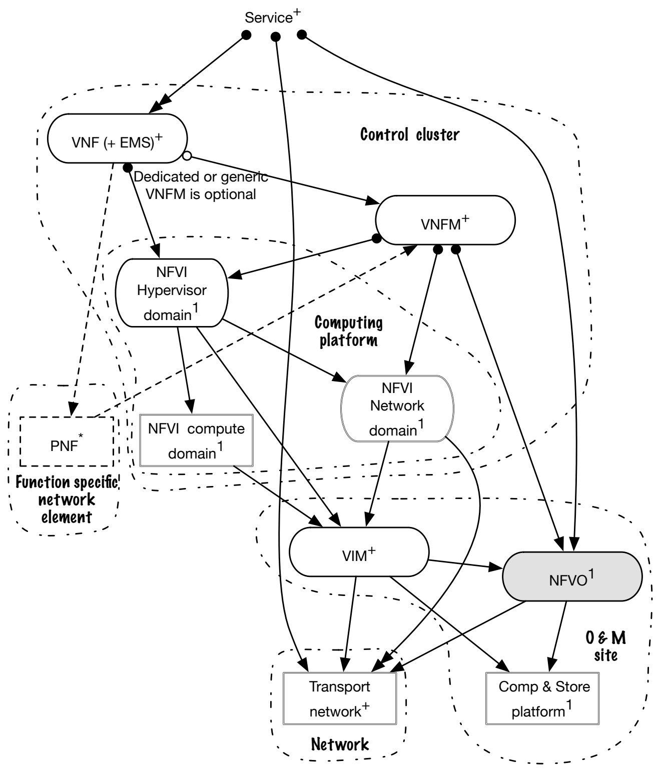

Fig. 6. Draft depends-upon graph based on the ETSI NFV architectural framework and a set of additional assumptions about the implementation.

(non-NFV) network is provided in Figure 5, where also the type of network subsystem (service) is indicated. Usually, we consider a network function as subordinate to operation and maintenance (O&M). However, in this context, the network function depends on proper O&M, although not in real time. In conventional networks, the function orchestration is integrated into the O&M block shown in Figure 5. The orchestration is not a separate function and hence, it is not shown in the Figure.

Establishing a depends-upon graph for a network with virtualised function may not be done directly from the architecture for the following reasons:

- The ETSI reference architecture is presented in various degrees of detail, see [1], [8], and the correct level of abstraction for the graph is not obvious.
- The description of the functionality of all architectural elements is not yet described in sufficient detail.
- The VNFs will be executed on the NFVI. However, it is not defined how the NFV O&M is supported.
- The realisation of the fault tolerance will depend on implementation choices, which is not a part of the standardisation work.

The prime objective in establishing the graph in Figure 6 is to discuss the fault tolerance provisioning. Nevertheless, from the above discussion it is necessary to make some assumptions in order to establish the graph.

- With respect to realisation, it is necessary to split the NFVI into a network domain and a compute and store domain. This is also in accordance with the approach in [32], [33].
- The NFVO and VIM are located on a separate platform. At this level of model granularity, this is necessary to avoid loops in the depends-on graph, e.g., the infrastructure manger depends on the operation of the infrastructure it should manage in order to work. This may not be necessary, but requires more detailed information about actual implementation to avoid mutual interdependencies.
- Note that in this context, the operation of the transport network is assumed to be independent of the NFVO. If it includes SDN, further challenges need to be addressed [5]. For a discussion on an SDN-enabled NFV architecture, see for instance [34] and Section 5.6 of [8].

Under a different assumption it might be necessary to revise the graph in this paper, or to make alternative graphs for other realisations.

Figure 6 shows that it is not an objective to make the physical and virtual network functions (PNFs and VNFs) inherently fault tolerant. The fault tolerance of the functions is to a large degree provided through system internal services by the management system, i.e. the VNFMs, the VIMs, and the NFVO. From the documents available, [28], [29], [32], [33], [35], [36], it is seen that a consistent architecture with respect to provision of fault tolerance in the NFV architecture is under development. However, for the moment, only fragments are described, and it is hard to get an overview.

The dependability of the MANO, especially the VIM and NFVO, is crucial, since all system functionality depends on it. In principle, an extreme availability may (likely) not be required for the NFVO, as real-time requirements are low. It means that short stops will not cause significant impact unless they take place during network reconfiguration. However, as will be explained in Section VII-B, this is a risky approach that may be reconsidered, and, based on the reconfiguration demands, limits on the maximum allowed NFVO downtime must be defined. In addition, it is crucial that the VIM and NFVO are designed for fail-omission/fail-stop semantics for all kinds of failures, cf. Section IV-A, as mis-operations may have catastrophic consequences.

Example: To demonstrate the depends-on graph and its adaptation to a specific real-world case, an example is considered. One of the NFV-related use cases that has been widely addressed by network operators in several Proofs of Concepts (PoCs) is the virtualisation of the Evolved Packet Core (EPC) [19], [37]–[39]. See [40] for a generic EPC description. This paper refers to one of the PoCs run at Telenor. One of the evaluated services was Voice over LTE (VoLTE). The corresponding depends-upon graph is presented in Figure 7. It is an instantiation of the generic Figure 6. The service-provisioning system consists of the VNFs from four different vendors, providing the following virtualised functions: BaseBand Unit (BBU), Mobility Management Entity (MME), Packet data network Gateway (PGw), Serving Gateway (SGw), Policy and Charging Rules Function (PCRF), Home Subscriber Server (HSS), and IP Multimedia Subsystem (IMS). The graph is adapted to the current PoC implementation. It was put together in a local data centre where all the compute nodes are located in the same room, while the traditional layer 2 switches supporting VLAN features are used for physical network interconnection reflected by the transport network in Figure 7. The VIM is based on the OpenStack platform, whereas OpenStack Neutron is responsible for the NFVI network domain.

In the case of the implemented PoC, all the vendors provided their own VNFMs. The orchestrator functions were carried out by manual operations. Therefore, some automation was missing, and it was not possible to observe all agility properties resulting from a fully orchestrated system. However, for illustration purposes, this paper considers a scenario where the mentioned virtualised EPC is fully orchestrated by the NFVO. In particular, to ensure dependability, the core management functionality, including VIM and NFVO, is placed on a separate platform.

In comparison with Figure 6, in the considered PoC system, each function is explicitly represented and has only one instance. Hence, there are only one-to-one relations 1:1 represented by plain arrows. The current VNFMs are implemented without real-time constraints and the boxes are made grey. Finally, there is no function specific network element and hence, the optional PNF in Figure 6 is removed from this diagram.

We considered the depends-upon graph for other specific use cases with considerable end-user differences from VoLTE. In all cases the results look structurally similar, since all they follow the architecture presented in Figure 6, with mayor differences only in the specific set of VFNs used. For instance, the use of the ETSI NFV ISG architectural framework for designing a content delivery network (CDN) is presented in [41]. A depends-on graph based on the global

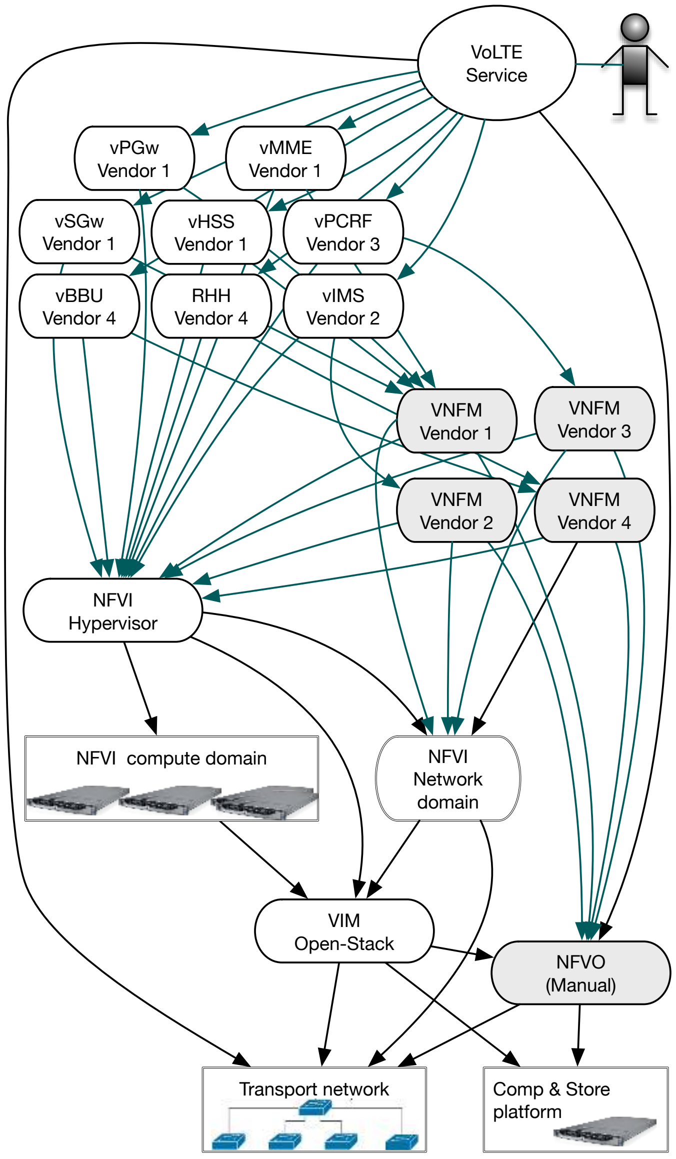

Fig. 7. The depends-upon graph for a VoLTE service in a virtualised EPC system.

virtual CDN architecture outlined in Section III.B of [41], will follow the structure presented in Figure 6, with VNFs such as: Request Routing, CacheInstance (vCache), Authentication Authorisation Accounting Instance (vAAA), Streaming server Instance (vStream) and Origin-server (OS) Instance. For a more detailed description see [41].

V. Monitoring and Failure Recovery

Monitoring and failure-recovery mechanisms are required to guarantee the dependability of an NFV system. In this section, we will discuss related monitoring and failure recovery alternatives that may be applied in the context of the NFVO.

A. Monitoring

There are several monitoring aspects to be considered during the design of an orchestrator solution. In this section, we present a classification of different relevant factors. The first

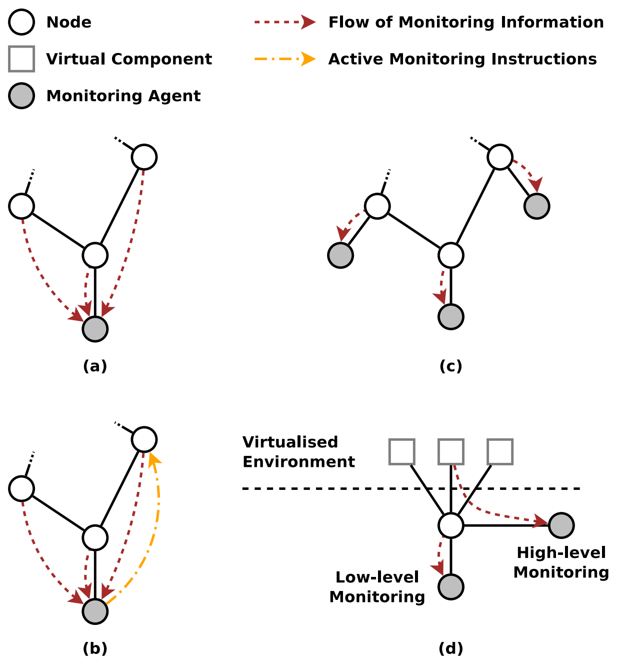

Fig. 8. An illustration of the ideas behind different monitoring schemes: (a) passive/centralised, (b) active/centralised, (c) distributed, and (d) low-level/high-level monitoring.

factor considers the ways in which different monitoring agents request and gather information, where monitoring may either be classified as passive or active (Figures 8(a) and 8(b), respectively). Further, the allocation scheme of monitoring units which collect and analyse data determines if the monitoring is distributed (Figure 8(c)) or centralised (Figures 8(a)-(b)). Finally, if the data is acquired from the layers above or below the virtualisation layer, the monitoring may be classified as high-level or low-level, respectively (Figure 8(d)).

According to the related specification recently proposed by ETSI [36], monitoring tasks may be passive, active, or a hybrid of both. Passive monitoring usually assumes that the monitoring server does not act proactively in order to gather extra information, but just collects unmodified data as it is reported by the agents allocated on the monitored elements. Passive monitoring for instance, may be based on analysing user traffic in real-time and assuming that the results of the measurements can only be collected at specific locations and processed off-line [36], potentially creating significant delays between events and corresponding actions. On the other hand, active monitoring is expected to enable proactive fault detection, where the server not only waits for information from monitoring agents, but it also performs autonomous tasks that may even require the generation of additional flows and tests needed in a specific monitoring situation. Further, using active monitoring techniques, it is possible to follow an iterative approach to analyse particular VNFs or NFVI resources without involving user traffic. In NFV, the design- or configuration-related decision about when and where to apply active and/or passive monitoring is fundamental for the dependability of the system.

The second important aspect is to determine whether the monitoring subsystem will be centralised or distributed. Centralised monitoring is based on the idea that the health and performance of a group of system components is monitored by a single unit which is also able to trigger further actions to respond to the observed anomalies. An example representing this approach might be the monitoring of KPIs of particular VNFs by a central VNFM subsystem to collect information needed for auto-scaling operations [8]. An important advantage of this strategy is that the response to the detected abnormal behaviour is coordinated by a single entity which maintains a complete view of the current state of the monitored subsystems. Thus, the corresponding action may be selected in a broader context of the entire group of similar subsystems. On the other hand, the monitoring unit may have to deal with alarm storms, which might slow down the expected response. To address this critical issue, distributed monitoring mechanisms may be deployed in the system. Distributed monitoring is based on the assumption that some components may monitor either their own operation (e.g., on-demand scaling of VNFs, in which an explicit request is sent to the VNFM based on the results of local measurements [8]) or the operation of a group of other subsystems. More importantly, a hierarchy of monitoring units may be built, effectively solving or limiting the overall impact of an alarm storm. An example of a solution that meets this requirement is the widely-used open-source Zabbix monitoring platform [42]. In this case, monitoring nodes can form a tree-like structure in which every node reports only to its master node [45]. In addition, alarm storms can also be suppressed through the use of trigger dependencies introduced in Zabbix. Trigger dependencies define specific conditions that must be satisfied before triggering an alarm. For example, if the only gateway router of a remote subnetwork fails, then it is expected that all monitored devices behind this gateway will not be reachable, and the corresponding alarms can be suppressed using the trigger dependencies-based mechanism. Furthermore, Zabbix has been selected as one of the fundamental components of the OpenBaton implementation of MANO [26]. It manages the *VirtualisedResourceFaultManagement* and *VirtualisedResourcePerformanceManagement* interfaces shared with VIM [8], [53] and it communicates with MANO through a plugin.

Finally, there are several components below the virtualisation layer related to the infrastructure and hardware resources, as well as components above the virtualisation layer that are related to VNFs and NSs [1]. Therefore, monitoring systems should be planned to operate on different levels, as the scope of their operation may differ significantly. In particular, low-level monitoring focuses primarily on the availability and performance of the physical components of the entire infrastructure, such as servers, storage arrays, and network equipment. The functionality and resources delivered by physical devices are aggregated and managed on higher layers. The corresponding monitoring mechanisms are supposed to detect abnormal operation conditions related to middleware and particular cloud-powered applications. Modern high-level monitoring solutions include capabilities such as advanced event filtering and aggregation mechanisms, as well as various alerting policies. To simplify management, the existing monitoring solutions

TABLE III
MONITORING AND FAILURE RECOVERY — CLASSIFICATION AND SUMMARY OF THE SELECTED PROPOSALS RELATED TO NFV (*italic font*), CLOUD/GENERIC SYSTEMS (NORMAL FONT), OR BOTH (**BOLD FONT**).

	Passive	Active	Centralised operation	Distributed operation	Low level	High level
Detection Localization	**[42]**, [43], [44]	**[42]**, [43], [44]	*[8]: VNFM*, **[42]** *[26]*, [43], [44]	*[8]: VNF*, **[45]**	**[42]**, [43], [44]	**[42]**, [43] [44]
Isolation Repair	[46]	[47]	*[24], [26], [48]*	[46], [47], [49] [50]–[52]		

often merge input data from all monitoring levels, process and correlate particular events, and present the results in a consistent way using a graphical user interface [43], [44], [54], [55].

The design of an effective monitoring subsystem requires that the following factors be taken into account:

- type of the monitored components and the involved performance metrics (ability to detect different kinds of fault);
- number of monitored components and their hierarchy (scalability, data aggregation and filtering);
- capability of the system to react to particular events in real time (alerting and self-healing policies, ensuring dependability of system components);
- dependability of the monitoring subsystem itself (redundant data sources, data processing units, connections, dealing with excessive amounts of data);
- management and deployment-related issues (integration with the monitored system, flexibility of configuration).

Although the existing technology allows to cover all of the listed factors, it may not be clear how to combine and configure particular components to work reliably in a specific deployment scenario. Thus, each deployment requires technical expertise in this area, especially with respect to proper dimensioning, dependability of the monitoring subsystem itself, and the ability to predict potential consequences of different solutions.

B. Failure Recovery

Whenever a failure is detected using any of the monitoring techniques previously described, further recovery mechanisms must be in place to bring the affected NSs back to their original condition. In NFV, the recovery can either be performed locally with the assistance of the corresponding EMS using pre-planned mechanisms, or it can rely on the global recovery provided at different scales by the VIM, VNFM, or NFVO.

Local recovery has been extensively used in cloud computing. One of its main advantages is the short recovery time offered. Current cloud computing technologies make use of different fault tolerance techniques to maintain high availability [56], [57], [58]. The specific solutions may differ in the way the image of a virtual machine replica is handled. For instance, there are two common techniques referred to as *active-hot replication* (e.g., [47]) and *passive-hot replication* (e.g., [46]). Active replication can obtain the best recovery time, since each process is performed at the same time on every replica which is actively running, and hence theoretically, any of them will be ready to take responsibility at any time in case of failure. Passive replication on the other hand requires each request to be processed on a single replica before the results are transferred. Due to the passive condition of the replicas they need extra time to take responsibility in case of failure recovery. Finally, hybrid techniques such as the one presented in [59] can combine different recovery mechanisms that make use of a given specific approach on a given time, depending on the current needs and status of the systems.

The general intention is to make the failures that affect a specific virtual machine as transparent as possible for the end user, by keeping the reaction time short due to the advantage offered by the local properties. The solutions used to restore virtual machines represent a hot research area and several related approaches have been proposed [49]–[52]. What is common to all these approaches is that a running system can transparently continue its operation on an alternative physical host in case of failure, which is also desired in an NFV system. Finding the way to integrate and take advantage of the existing cloud computing fault tolerance techniques in the NFV solutions is still an open issue that should be explored.

On the other hand, given the openness of the NFV specification on recovery related implementations, most of the current NFVO proposals include global mechanisms that take corrective actions when failures are detected in different components of the NFV system [24], [26], [48]. Having a global entity in charge of the recovery procedures allows centralised coordination for identification of the system-wide optimal solution, as well as effective troubleshooting, given a more complete context related to the failure events.

Regardless of the scope of recovery (local or global), it is important to keep in mind that the NFVO should have a complete and consistent view of the entire NFV system. In particular, it should be able to track available resources, as well as current locations of VNFs and NSs across the entire system. As local recovery processes may modify the allocation scheme on demand, it is crucial that the NFVO be notified of such changes. In the case of global recovery, since the related procedures may be started by the NFVO directly, it is relatively easy to maintain a consistent view of the network. At the same time, when local recovery routines are executed, it is important that the NFVO receive agile updates to make sure that the following decisions are made in the appropriate context. Finally, discovering the ways in which local and global recovery should work together in NFV, the specific NFV implementations, and trade-off/balance between these two approaches represent interesting open issues in NFV dependability.

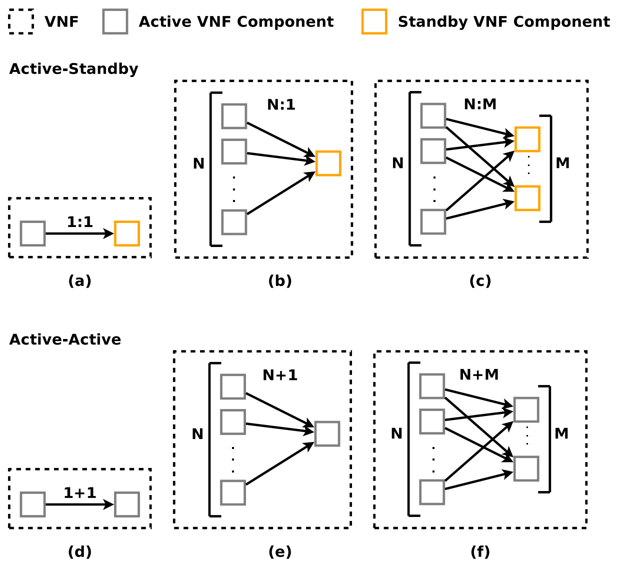

Fig. 9. An illustration of the ideas behind different VNF redundancy schemes — Active-Standby: (a) $1 : 1$, (b) $N : 1$, and (c) $N : M$; Active-Active: (d) $1 + 1$, (e) $N + 1$, and (f) $N + M$.

Redundancy planning is an important concept for the successful recovery of VNFs. Figure 9 presents different VNF redundancy schemes studied by the ETSI NFV-REL working group for two primary protection scenarios: *Active-Standby* and *Active-Active*. We provide below a brief summary of considerations given in [27], [35]. In the case of the Active-Standby redundancy schemes (see Figures 9(a)-(c)), to avoid common-mode failures and reduce the probability of correlated failures, the standby and active instances should be placed on different hardware nodes. Further, failures may cause a noticeable disruption of service during the failover, due to reduced processing capacity of the active instances. As standby instances do not perform application load processing, additional mechanisms may be required to confirm that the standby instances are ready to handle the load after failure, and network reconfiguration is required to direct traffic to standby components. When the current state of the VNF is important, the standby instances should have sufficient memory capacity to store the state information corresponding to each of the N active instances. In the case of the Active-Active redundancy schemes (see Figures 9(d)-(f)), load distribution functions are needed in front of the pool of active resources. However, the load distribution mechanism itself should also be protected against failures, which may involve both redundancy and state replication. For a discussion of different examples of Active-Standby and Active-Active redundancy schemes, the reader is referred to [35].

The state of the VNF is also a key consideration for its failure recovery. Operations of stateless VNFs assume that in the case of failure, the new VNF only needs to provide the same functionalities of the failed one, without considering the VNF state. Thus, the related advantage is that no additional delay is imposed as a result of state synchronisation procedures, and there is no risk that the recovery will be interrupted by a mistake in the related process. At the same time, in the

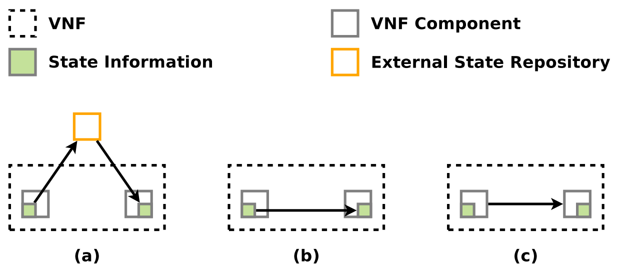

Fig. 10. An illustration of the ideas behind different VNF state synchronisation schemes: (a) external VNF state replication, (b) direct partial VNF state replication, and (c) full VNF state replication.

case of stateful VNFs, the redundancy schemes need to be coupled with appropriate state synchronisation mechanisms. The selected three general strategies discussed in [35] are presented in Figures 10(a)-(c). The first strategy relies on an external state repository (see Figure 10(a)) which maintains a copy of the internal state of the active VNF component. Once a failure is detected, the NFVI and NFV-MANO localise the failure and disable the affected VNF components, which may also involve network reconfiguration actions. It is required that the standby component be brought to the state consistent with the state stored in the external state repository. Once the state is synchronised, a VM failover is performed to use the standby VNF component as the new active component. Finally, a new standby instance is assigned to the VNF by the NFV-MANO from the resource pool. The second strategy is based on direct partial VNF state replication that occurs between the active VNF component and the corresponding standby VNF component (see Figure 10(b)). Finally, the third strategy assumes full VNF state replication, which also includes full VM execution state replication. While each of the three presented strategies provides a way to transfer state information between VNFs, they do not detail the underlying synchronisation mechanism. As it will be explained in Section VII-C, this is still an open challenge with huge research interest.

Table III summarises the most relevant proposals of the monitoring and failure recovery classification presented in this section. In the following sections, the selected implementations and challenges of monitoring and failure recovery will be discussed in the context of NFVO.

VI. NFVO Functionalities, Requirements and Architecture

In this section, we discuss dependability challenges related to NFVO. First, we identify the main functionalities of NFVO. Then, we summarise the dependability requirements specified by NFV ETSI ISG and we present the latest 5G Standardisation Work of 3GPP. Finally, we discuss the current architectural solutions with respect to the NFVO subsystem, referring to particular NFVO functionalities.

A. NFVO Functionalities

As already presented in Section II-C, the functionality of the NFV-MANO is defined in the corresponding specification released by ETSI [8]. The NFVO functionalities can be divided

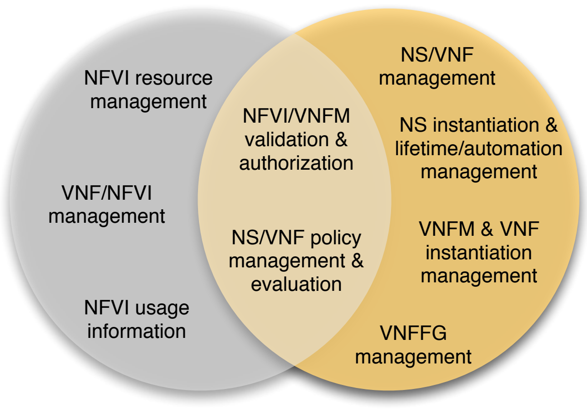
NFVI resource management
VNF/NFVI management
NFVI usage information
NFVI/VNFM validation & authorization
NS/VNF policy management & evaluation
NS/VNF management
NS instantiation & lifetime/automation management
VNFM & VNF instantiation management
VNFFG management

Fig. 11. The NFVO functionalities split between Resource orchestration (grey circle) and NS orchestration (yellow circle).

in the following two categories, as described in [8] and depicted in Figure 11:

- *Resource Orchestration* — the orchestration of the NFVI resources across multiple VIMs;
- *Network Service Orchestration* — the lifecycle management of NSs.

According to the proposed division scheme, some functionalities belong to both categories but have different prospectives, i.e., the NFVI resources or the impact on NS. Moreover, the resource and NS orchestrations mainly focus on NFVI resources and NS instances, respectively; however they also consider the relationship with the corresponding VNF instances.

Furthermore, ETSI has specified the functional requirements of MANO, which are defined in [60]. Based on this specification, key operational function categories can be identified. The first branch of operational functions is related to the separated management of NSs and VNFs: *information management*, *lifecycle management*, and *fault management*.

VNF *information management* includes the management of VNF package and VNF instance information. NS *information management* comprises the management of the NS deployment template, NS instance information, and NS performance. In general, information management includes verification and validation of integrity and authenticity, as well as retrieving and collecting information and performance status.

Lifecycle management of VNFs and NSs includes instantiation, scaling, updating, and terminating VNFs and NSs, respectively.

Fault management includes collecting alarm notifications, providing fault information, requesting healing, and preforming automated or on-demand healing.

Virtual resource management consists of managing the association between NS/VNF and the NFVI resources through resource commitment models (reservation model, quota model and on-demand). VNF-related virtual resources include compute and storage resources necessary for VNF components, as well as networking resources needed to ensure intra-VNF connectivity. NS-related virtual resources comprise networks, subnets, ports, addresses, links and forwarding rules, and are used for the purpose of ensuring inter-VNF connectivity. The management of virtualised resources includes allocation, update, scaling, and termination, but also the respective failure and information management. NVFO manages the resources belonging to the virtual infrastructure by cooperating with one or more VIMs.

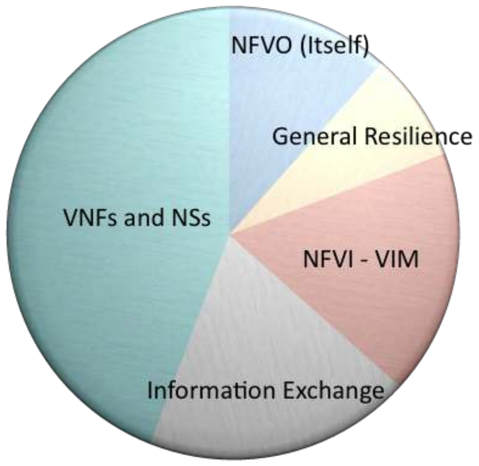

Fig. 12. A standardization-based classification of the resilience requirements of the NFVO.

The operational functions of the other elements of the MANO can also belong to some of the above categories. In particular, virtual (infrastructure) resource management is carried out by VIM, while VNFM is also taking care of virtual resource management and information/lifecycle/fault management in the context of VNFs.

In conclusion, we can summarise NFVO functionalities as the (direct and/or indirect) management of NSs, VNFs, and NFVI, including the relationships among them, their various instances, and their different types, through coordination with the other two major components of MANO (VNFM and VIM).

B. NFVO Dependability Requirements According to ETSI Standards

After getting an overview of NFVO functionality, having a perspective on its dependability requirements is also important. The ETSI-NFV reliability, availability and assurance working group, *NFV-REL*, has documented the general reliability requirements for the overall NFV architecture in [28].

Having a strong focus on the dependability of NFV-MANO is important, not only due to its fundamental role for the entire NFV system, but also because it has been identified as one of the less mature parts of NFV (especially with respect to the NFVO). Therefore, in 2016, the ETSI NFV-REL group started to work specifically on the resilience requirements and capabilities of the NFV-MANO [27]. Additionally [27] has included compiling the NFVO related requirements from the previously mentioned ETSI-NFV-REL-001 [28] standard. Here, we follow the same approach, but we use our own classification (see Figure 12) to provide a clearer overview of the dependability requirements that NFVO has from the standardisation point of view.

First, we have the "general resilience" requirements (8%), which refer to avoiding single points of failure, providing resilience mechanisms that are vendor-agnostic and supporting different availability levels. Second, there are some references requiring the NFVO itself to have a resilient design (12%) with redundant resources and support for geographically distributed deployments. Third, some requirements involve the interaction with between NFVI and VIM (17%), involving topics such as redundancy assurance, hardware failure detection and remediation, and monitoring of overall infrastructure utilisation and performance in order to detect potentially dangerous infrastructure behaviour. Fourth, information exchange requirements (19%) refer to the capabilities of the different descriptors in order to provide sufficient and explicit data and procedures that enable the efficient implementation of resilience mechanisms that can be measured and verified. Finally, VNFs and NS represent for end-users the most tangible aspect of NFV, and they have received major attention. NFVO requirements in this direction represent approximately 44% of the total NFVO requirements, including issues such as providing automatic restoration capabilities, monitoring, migration, and escalation of VNFs and NSs to prevent failures; fast and agile restoration mechanisms aligned with Service Level Agreement (SLA) requirements; replication and load distribution of VNF and NS clusters; and smart mechanisms for the efficient and fast identification and response of reported alarms. To conclude, the requirements also consider the implementation of proactive routines to avoid potential failures due to predictive mechanisms.

In summary, the NFVO must fulfil the general-systems prevention, tolerance, removal, and forecasting dependability requirements [10] in the interaction with the NFVI, VNFs, NSs, and specially on its own operation. The NFVO must provide the mechanisms to prevent faults by having a robust and well-planned design, including adequate monitoring and information exchange tools; additionally, the NFVO must provide fault tolerance and removal tools by guaranteeing the redundancy needed in all the physical and logical levels, complemented by intelligent and efficient recovery and remediation mechanisms. Finally, the NFVO provides fault forecasting tools that enable it to act in a proactive way via intelligent mechanisms, and avoiding some potential problems and improving preparedness to mitigate them.

Since the NFVO is an influential power modifier for the entire NFV system, its operation correctness, high performance, and high quality of service become fundamental requirements that must be guaranteed. An extended and detailed list of the standardisation requirements can be found in [28] and [27].

C. Latest 5G Standardisation Work of 3GPP

During the recent years, the research community and telecommunication industry have made massive efforts to advance the ongoing standardisation work on the fifth generation (5G) cellular networks. 5G will be a major innovation step in wireless communications by integrating various wireless technologies to be able to offer high performance for a broad variety of use cases, such as broadband access in dense areas, increased user mobility, massive Internet of Things (IoT), real-time communication, lifeline communication, ultra-reliable communications, and broadcast-like services [61].

To achieve this ambitious objective, NFV will play a key role within the 5G network architecture, enabling Cloud Radio Access Network (C-RAN), Mobile Edge Computing (MEC), multi-domain/multi-provider orchestration, network programmability, and network slicing [62].

In 3GPP, and especially in the Service and System Aspects (SA) Working Group 5, significant effort has been made to provide technical specification for the design of the 5G system [63]. In [64] is presented the telecommunication management of mobile networks that include VNFs, and it explains the relationship between the 3GPP management architecture and the ETSI ISG NFV management and orchestration. The specification covers such aspects as fault management, configuration management, performance management, and lifecycle management. Further, it indicates the corresponding requirements with particular reference to the NFV-MANO. In [65], the requirements for fault management are presented in more detail. The specification highlights a major role of the VNFM with respect to alarm correlation and reporting, VNF healing, and virtualisation-specific failure detection and correlation. In this context, the main functionality of the NFVO is the NFVI maintenance coordination in order to avoid unwanted impact of NFVI maintenance on VNF applications.

D. Architectural Solutions

As presented in the beginning of this section, ETSI specification defines the functionalities and the respective requirements of each building block of the MANO framework, but it does not specify actual implementation and deployment. With respect to implementation and deployment, both academic and industrial research communities have been active in proposing solutions to implement the NFV MANO. Table IV depicts NFVO functionalities as presented in Section VI-A, versus current architectural solutions. We have categorised the following architectural approaches:

- *Specific module* — there is a dedicated module for NFVO functionality;
- *Aggregate module* — a subset of functionalities is aggregated in a single module;
- *Modified MANO* — the MANO architecture is different from the ETSI specification, therefore the functionalities are included in different subsystems;
- *Including SDN* — the NVF is integrated with Software-Defined Networking (SDN).

As follows, we will describe the table findings row by row, presenting current architecture solutions for each NFVO functionality to provide a better understanding of the different architectural approaches.

1) Information management: The first row of Table IV refers to information management and includes two sub-rows to specify relationships to VNF or to NS. Some of the information management tasks are fundamental, and hence, transversal to any architectural approach, such as the verification of the authenticity, integrity, and standardised mandatory

TABLE IV
NFVO FUNCTIONALITIES VS ARCHITECTURAL SOLUTIONS

		Specific module	Aggregated module	Modified MANO	Including SDN
Information Management	VNF	[42], [66]	[48]		
	NS	[66]	[48]		
Lifecycle Management	VNF	[26], [48]	[24]	[66]	
	NS	[26], [48]	[24]	[66]	
Fault Management	VNF	[26], [48]		[66]	
	NS				
Virtual Resource Management			[24]	[48], [66]	
Network Controller					[25], [48]

information. As already stated, one of the main tasks of information management is to collect and retrieve the information across the NFV system and the performance status of the different NFV elements. To this target in the existing proposals of NFV-based architectures, the monitoring subsystem is an important element, which is included either as a separate module or as a part of a more general subsystem. For example, Verizon introduced monitoring within a Service Assurance component [48], which also performs other functions, such as Fault Management. In the OpenBaton implementation of MANO [26], a specific module called Monitoring Driver is included that interacts with the Zabbix monitoring platform [42] through a plugin. Further, AT&T proposed an architecture in which monitoring is performed within the Data Collection, Analytics, and Events (DCAE) subsystem [66].

2) Lifecycle management: The second row of Table IV refers to the lifecycle management of VNF and NS. In the architecture proposed by Verizon [48], there is a functional block called End-to-End Orchestration (EEO) that is responsible for lifecycle management of both VNF and NS (VNF orchestration is done by the NFVO, which in turn is a subcomponent of the EEO). OpenBaton has specific modules that take care of particular aspects of a the lifecycle management. Autoscaling Engine interacts with the monitoring system to dynamically scale VNFs in order to accommodate NS requirements.

In Open Source MANO (OSM) [24], the NFVO functionalities are split between two elements similar to categorisation in the ETSI specification: Resource Orchestration (RO) and Service Orchestration (SO). Lifecyle management should be part of the SO.

AT&T defined a Software Platform for Enhanced Control, Orchestration, Management and Policy (ECOMP) [66]. Compared to ETSI MANO, instead of NFVO, VNFM, and VIM, there are DCAE, a VNF manager, and an Infrastructure manager. The VNF manager and Infrastructure manager are also composed of two components: an Orchestrator and one or multiple Controllers. The lifecycle management is included in the VNF manager.

3) Fault management: The third row of Table IV refers to fault management. In NFV, fault management is based on monitoring health and performance indicators, followed by actions that aim to rectify the problems detected. According to the recent specification released by the ETSI-NFV [8], failures should be investigated and resolved without unnecessary delay, preferably by the subsystem that has access to all necessary information to identify the root cause of a failure, and to take the appropriate countermeasures. This specification [8] leaves open the possibility of performing fault management in a centralised or distributed way by stating *"fault management can be centralised or distributed, and no assumption is made here regarding either"*. For instance, the *NFV fault management flow* in Annex B of [8] presents the deployment of information-collection, fault-correlation, and fault-recovery points across different NFV management entities (VIM, VNFM, NFVO and EM). It represents the flexibility from the standardisation point of view in order to implement different recovery techniques, including global approaches where handling goes to the NFVO or solutions via local management.

From all the available NFVO proposals, OpenBaton [26] provides the best level of detail on failure recovery techniques by proposing a Fault Management module that includes retrieving the policy contained in the different VFN and NS descriptors, creating rules for detecting faults on the monitoring system, registering triggers, and finally every time a trigger is received, executing an action which implies the healing of the affected parts. The architecture proposed by Verizon [48] also proposes specific and separate fault management module. AT&T considers failure management and healing mechanisms as part of its modified MANO proposal ECOMP architecture [66]. Current architecture proposes only NS fault management, the respective subset of VNF fault management a main focus. However, having a generic approach to guarantee the availability of end-to-end NSs is still an open issue.

4) Virtual resource management: The last NFVO functionality in the table is virtual resource management. In the Verizon architecture, this functionality has been delegated to the VIM. Similar to lifecycle management, OpenBaton has specific modules that take care of particular aspects of virtual resource management. For example, there is a module called Network Slicing Engine, which enables the deployment of multiple virtual NSs in parallel by managing the virtual infrastructure resources.

In OSM, virtual resource management is part of the Resource Orchestration. In the AT&T ECOMP, the virtual resource management is included in the infrastructure manager.

5) Network controller: The last row of the table refers to the network controller, which is not one of the already presented functionalities of the NFVO, but it is an optional element of the NFV. The scope of the network controller is the orchestration of connectivity between the components of a given VNF, connectivity among various VNFs composing a NS, and connectivity to the PNFs. In the Verizon architecture [48], the network controller is a SDN controller. In the proposed architecture, there are three SDN controllers that are related to different network domains: data centre, Wide Area Network (WAN), and wireline or wireless access network. Open-O [25] also includes SDN orchestration as part of its global service orchestration module, setting it up at the same level as the NFVO module.

The specific architectural solutions selected to implement the NFVO will have a strong influence on the dependability of the NFV system. Based on that, in Section VII, we will present the considerations needed for NFVO to deploy dependable NSs, and in Section VIII, we will analyse the dependability principles that the different architectural features should follow, in order to produce a robust NFVO.

VII. Impact of NFVO on NS Dependability

In this section, we discuss how NFVO could affect the provision of a dependable NS. Firstly, we introduce the definition of NS in the NFV. Secondly, given the NFVO functionalities presented in the previous section, we assess the dependency of the NS on NFVO by investigating the impact of NFVO unavailability. Finally, referring to the presented current architectures of NFVO, we discuss the threats that can be present in providing a dependable NS.

A. Network Service

According to the ETSI specifications, the NS can be defined as the subset of the end-to-end service formed by VNFs and associated Virtual Links (VLs) instantiated on the NFVI (see Figure 13). The procedure for providing a NS by using NFV is similar to the one specified by the Service Function Chaining (SFC), which is “the definition and instantiation of an ordered set of service functions and subsequent steering of traffic through them” [67]. In the following, we explain in detail the composition of a NS in NFV, where SFC is a part.

The NS is composed of two graphs (see Figure 14). The first is a Network Connection Topology (NCT) graph that specifies the VNF nodes that compose the global service and the connection between them using the VL concept. Each VL is connected to a VNF through the Connection Point (CP), which represents the VNF interface. VLs are also used to possibly interconnect the VNFs to PNFs. The second type of graph is the VNFFG, established on top of the NCT. The VNFFG is composed of Network Forwarding Paths (NFPs) that are ordered lists of CPs forming a VNF chain, also known as Service (Function) Chain.

On providing the NS, the NVFO plays a key role: it is the single point of access for all requests from the Operations Support System (OSS) to simplify the interfacing; it handles the lifecycle of NSs and VNFFGs and it has the end-to-end view of the resources being allocated across NS and VNFs by VNFMs (which handle VNFs lifecycle from an application point of view) so all requests for resource allocation transit

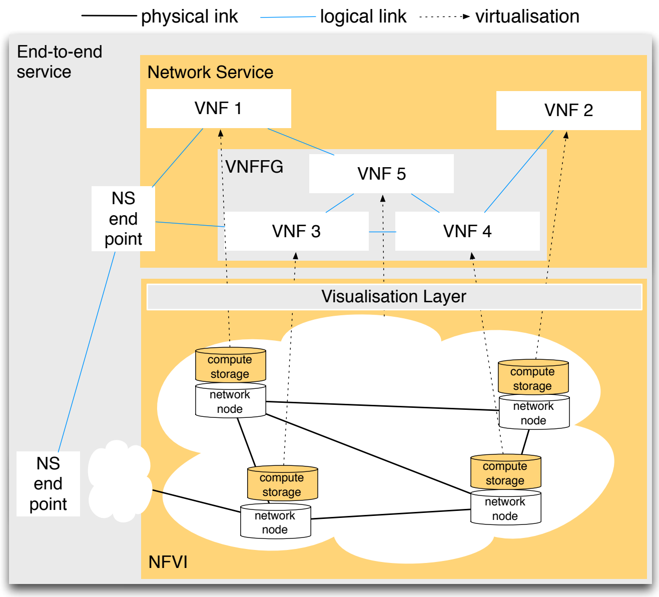

Fig. 13. Example of a Network Service as part of an end-to-end service with VNFs and a forwarding graph [8].

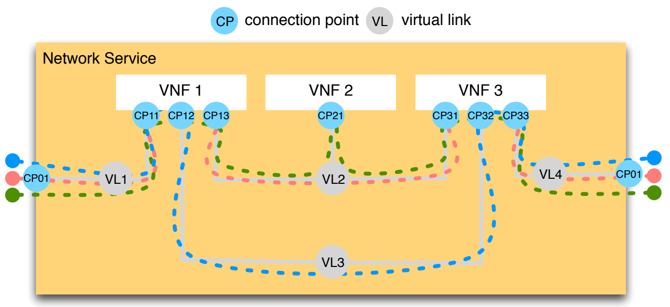

Fig. 14. Network Service with two VNFFGs with different NFP [8]

through the NFVO. Hence, handling of all services depends directly on the NFVO, as illustrated in Figure IV-D, which makes correct operation vital for the system. This will be further discussed in Subsection VII-C5.

B. Dependency of NS on the NFVO

Dependability of the NFVO has not received the attention that it deserves, in some part because it is common to argue that if it fails, there are no downtime implications for the deployed VNFs and NSs. This argument is only true to some extent. This calls for a design where NFVO and the managed targets are allocated in independent failure domains to avoid simultaneous downtime resulting from the correlation of failure propagation.

We want to analyse more carefully the implications of the unavailability of NFVO. This may be estimated from its architecture and specific functions, as indicated in Section VI-A. The unavailability of NFVO results in the inability of the system to perform any service or resource orchestration. In this sense, the original argument that the absence of an NFVO does not have downtime implications for the running functions is partially true for some of the already established functions. However, it is important to be aware of the alarming vulnerability state of the entire NFV system whenever the NFVO becomes unavailable. In particular, the risk of not being able of perform further services and resource orchestration can be categorised as the inability of the system to:

- Deploy new services and features;
- React to any event that requires modifications of the current configuration.

The severity of the first case depends on the importance and urgency of the new services that need to be deployed. However, the second point described poses an even more serious threat to the entire NFV system, since under normal operations, the existence of events that demand urgent reactions may be common and frequent.

The severity of possible consequences increases rapidly with increasing NFVO downtime. For example, a short downtime may result in the inability of the system to scale NSs according to changing demands, causing transient delays. Further, local fault management may be affected considerably due to degraded fail-over capabilities of particular subsystems, which might increase the impact of failures on overall system performance. In extreme situations related to long unavailability periods of the NFVO, the entire NFV system would be unable to handle an excessive load, perform service and resource orchestration, or deal with accumulating failures effectively. In such a case, the risk of further outages increases significantly, which may create a domino effect with serious consequences. Thus, the reaction capacity is a fundamental requirement that every NFVO system should meet.

C. Dependability Threats and Challenges

In this section, we present key orchestration features that the NFVO must have, but that may bring a series of threats and challenges that need to be addressed in order to provide dependable NSs.

1) Monitoring all components and system layers: The adequate provisioning of NSs demands the regular monitoring of their availability and performance in order to detect abnormal conditions and take appropriate countermeasures. In an NFV system, NSs may involve multiple VNFs and PNFs, which may include several layers such as application, virtualisation, and infrastructure. Abnormal situations, such as failure of system components or degraded performance measures according to the predefined KPIs, may trigger the corresponding alarms to start a recovery procedure. In some cases, an event can trigger several related alarms, due to the multiple layers and to the complexity of the system structure, leading to an alarm storm which consumes significant system resources and delays the expected response. Thus, dealing with alarm storms effectively becomes an important requirement that must be met by a well-performing and reliable monitoring subsystem, as pointed out in Section V. The potential solutions to this issue include intelligent filtering of triggered events, both at the source and on the receiver's side, as well as aggregation of closely related events. In particular, distributed monitoring based on a tree-like hierarchy is one of the currently-considered approaches (see Section V for the related discussion).

2) Failure management: An NFVO must have failure management features. The design and operation principles of an NFV system introduce both advantages and disadvantages with respect to the capability of the system to handle failures of VNFs and NSs. The primary advantages are scalability and flexibility in deploying redundant components, backup instances, and recovery mechanisms in a cloud-like architecture [68], [69]. Compared to the physical systems, recovery actions in the cloud can benefit from the resource pooling flexibility, increasing recovery possibilities that may improve the efficiency of the process [35]. On the other hand, NFV related monitoring (detection) and troubleshooting (localising and repair) may be more complex. Querying the subsequent system components for diagnostic data may consume a significant amount of time, introducing additional delays before the system can initiate replacement or repair. As NSs may be delivered by several chained VNFs and PNFs, and the recovery procedures are different in complexity and scope, the recovery may either finish successfully or fail. In particular, when the active and backup instances of a VNF reside on the same failure domain, all stages of the recovery process must be well designed and highly reliable, and the placement of backup VNF instances must be well managed.

Finally, as mentioned in Section V, the existing techniques to handle failures of NSs may be categorised with respect to their scope as either local or global. Currently, the criteria to decide which of the two strategies is better in a given specific scenario are not clear, and some of the solutions are still under study [8]. VNF developers provide mechanisms to perform local recovery in a fast and efficient way. On the other hand, different NFVO projects include as an integral part of their solutions a fault management module that is able to perform global recovery. In this context, there are four challenges that still need to be addressed, including he following:

- Mechanisms must be provided that go beyond the fault management of individual VNFs in order to guarantee the availability of end-to-end NSs.
- Local recovery has been tested in environments with few VNFs with specific and well delimited features. However, it is still not clear how to design generic local recovery mechanisms in complex service chains.
- A flexible architecture that enables the selection of local or/and global recovery must be designed. In particular, there is a need to design a hierarchy that organises the use of the different levels of the existing techniques, so that it is possible to use local recovery as a backup of global recovery duties and vice versa.
- Whenever local recovery techniques are used, there must be a guarantee that the NFVO never looses track of the current state and location of the different resources, VNFs, and services.

3) Management of Stateful NSs and VNFs: Some NSs require the reliable state synchronisation of stateful VNF instances. The state synchronisation is usually specific to VNFs, and thus it must be performed internally as a part of the repair process [35]. The state protection phase, as defined in [35], may either involve full Virtual Machine (VM) state replication

TABLE V
SUMMARY OF RELEVANT NFVO FEATURES THAT MAY BRING SPECIAL THREATS AND CHALLENGES

NFVO Feature	Threats	Challenges
Monitoring all components and layers	- Alarm Storms	- Efficient alarm aggregation. - Efficient alarm correlator.
Failure Management	- Inefficient and long recovery time - Unsuccessful / Disregarded recovery. - Inconsistencies when performing local and global recovery.	- Guarantee fast recovery time. - Guarantee the solution of a reported problem. - Successful coordination of local and global recovery.
Management of Stateful NSs and VNFs	- VNFs and NSs inconsistencies after recovery. - VNFs and NSs extended downtime due to synchronisation problems.	- Provide mechanisms to guarantee the correct state synchronisation of VNFs and NSs. - Provide support and solution in case of detecting inconsistencies.
Interaction with heterogeneous VIMs	- Incompatible and restricted VIM interaction. - Failures in the interaction between the NFVO and some VIM solution.	- Generic management of heterogeneous VIMs. - Design standard VIM-interaction protocols and plan for their efficient troubleshooting routines.
NFVO Operations Correctness	- Timing failures when the NFVO respond to a request out of time, generating inconsistencies. - Value failures when the NFVO set the wrong value in the NFV system.	- Create methods to anticipate, detect, diagnose, and correct the misbehavior of the NFVO. - Identify the cause of misbehavior and find a solution to avoid it in future operations.

or partial state replication, whereas the state information can be stored at VNF instances themselves or on an external unit. Stateful NSs and VNFs represent services where each action taken may change the state of the system and the execution of the next command, since the current state may depend on the output obtained previously. Given this particular situation, those services are more difficult to manage, since in addition to planning redundancies and recovery techniques, these NSs and VNFs need a mechanism that guarantee consistency of the state. In ETSI-NFV Proof of Concept (PoC) 12 [70], a stateful SIP proxy distributed server was evaluated. The results demonstrated high reliability of the deployed VNF by ensuring service continuity substantiated and confirmed by the distributed processing middleware. However, how this design scaled for stateful NSs, which are composed of several VNFs is still an open issue. The NFVO should take a central role in solving this issue. Finally, having a consistent distributed system state image may be technically feasible, but will come with a cost of reduced performance due to the extra time needed for synchronisation [71]. Therefore, the selected solutions have to be planned according to the performance needed and the implied speed of the consistency-guard mechanisms.

4) Interaction with heterogeneous VIMs: To be in line with the flexibility vision of NFV, an NFVO cannot be locked to an specific VIM. Therefore, the reference Orchestration point *Or-Vi* is defined by ETSI-NFV [8]. *Or-Vi* is used for information exchange between the NFVO and the VIM, and supports functions such as resources reservation, allocation and release, VNF software image addition/deletion/update, and configuration and measurement related to the NFVI. The robustness of the NSs depends on how efficient the NFVO is in using appropriate resources. Managing heterogeneous VIMs is challenging; it is important to (i) have a generic and standardised Or-Vi reference point to guarantee interoperability between the NFVO and various VIM solutions, which then allows reliable and VIM-agnostic communications, and (ii) identify the technical differences on the potential VIMs to be used, especially regarding performance.

5) Correctness of Orchestration operations: In Section VII-B, we mentioned the dependency of NS on NFVO and failures having huge impact. If the NFVO crashes (omission failure), then the network is in a very vulnerable position, and the NSs might be affected. However, even when the NFVO has not crashed, it might provide incorrect responses due to a value and/or timing failure (see failure semantics introduced in Section III).

- *Timing failures* may be present, for instance, when an alarm storm is not well handled by the NFVO. We may assume that the time to filter and clear the alarm storm is long, and when the reaction is executed by the NFVO, the conditions have already changed, generating inconsistencies. In this example the NFVO responds with correct value, but not within the specified maximum time, and hence the response must be disregarded.
- *Value failures* represent a potential threat in the NFVO, especially due to the unavoidable existence of bugs in software, which lead to an incorrect valued NFVO response; e.g., the location of some infrastructure or services may be temporally misinterpreted, leading to wrong decisions on new deployments or modifications on current services, resulting in serious consequences.

The need for new methods to anticipate, detect, diagnose, and correct the potential misbehaviour of the NFVO is still an open issue that must be addressed. In addition, identifying the cause of misbehaviour and the mechanisms needed to prevent it are central and critical challenges that need to be solved.

Table V summarises the NVFO features described here, and the respective threats and challenges that result from them.

VIII. Fault-tolerant NFVO

As mentioned in Section VII-B, the unavailability of NFVO inhibits the deployment of new services and system features and the inability to react to several NFVO-dependent events that require modifications or corrective actions, putting the system in a very vulnerable state. Therefore, having a highly dependable NFVO is an important duty. This section will provide concepts and policies that are useful in order to better understand how to fulfill such obligation by describing this process in relation to platforms such as the VIM and the SDN controller and describing principles related to designing a reliable NFVO.

A. Fault Tolerance and Recovery in NFVO-Related Approaches

NFVOs must be robust. The general solution is having an NFVO that can operate under the presence of any kind of failure. For this, redundancy, fault detection, and diagnosis and fault-tolerant mechanisms must be rigorously planned. There are several background and best practices in related as well as more mature technologies, such as the VIM or the SDN Controllers, which can be used as a reference.

The basic SDN architecture depicts the controller as a potential single point of failure [71]. Therefore, control plane must be fault-tolerant. The straightforward solution is to have redundant controllers that can take responsibility in case of failures [81]–[83]. The balance between consistency and performance is one of the most important challenges to be considered, since the controller is a stateful and highly dynamic system, where the network image changes constantly with each new command. Such network images must be the same for all controller entities. Today, it is technically possible to have distributed and synchronised data-stores [84], [75], [86], but it takes some time on each transaction, which may negatively affect the performance. Therefore, it is important there to be balance between consistency and performance.

The ONIX proposal [84] provides a list of concepts and pillars to be considered in the implementation of a general fault-tolerant SDN controller platform, although the solution to some specific challenges is left open. A fault-tolerant controller platform may be implemented by distributing the load among separate controller units, which means that the responsibility has to be spread over several domains [84], [85]. This represents huge advantages in terms of redundancy and recovery efficiency, since there are several units available that can take responsibility immediately after a failure is detected, and the maximum capacity (handled flows within a time unit) of the controller as a whole is much bigger. However, the synchronisation of the network image shared by all controllers is more challenging. On the other hand, there is a simpler approach known as Master-Slave, where a single master controller is in charge of all decisions [81]–[83], and it is supported by backup controllers having a synchronised view of the network-image. Here, the challenge is that when the load is managed by a single master, the maximum capacity of the controller is constrained to lower values.

The SDN controller is a good example of a logically centralised unit that gives directives about policies and mechanisms executed in the network. Another logically centralised system that may have useful similarities with the NFVO is the cloud computing infrastructure controller, such as OpenStack [87]. The cloud computing infrastructure controller is composed by different types of nodes: controller, compute, storage, network, and utility. Controller nodes are responsible for running the management software services needed for the OpenStack environment to function. Each of those node-types need to be protected against failures. The fault tolerant strategies depend on the state of the services, i.e., if they are Stateless services (such as Nova, Neutron, Cinder, Glance) or Stateful (such as MySQL, Message queues). As mentioned in the previous section, if the controller node service is stateless, the fault tolerant mechanisms must take care of the deployment of multiple controller nodes with redundant HA-Proxies [88] and auto healing properties. In case of stateful services, it is important to provide appropriate distributed storage mechanisms. OpenStack has two main approaches to provide fault tolerance on the control nodes: *Pacemaker* [76] and *Keepalived* [89]. *Pacemaker* is a cluster manager that coordinates the actions of various services. It is defined as a distributed system capable of reliably monitoring and recovering all the OpenStack components across the entire cluster. *Pacemaker* manages Virtual IPs, Load Balancers, and Controller services in order to maximise the availability of OpenStack APIs. *Keepalived* is another fault tolerant approach optimised for Active/Active services that does not require any inter-machine coordination. This may be advantageous since in an active/passive approach the recovery time is usually longer, while in active/active scenarios, all controller nodes are running simultaneously. However, *Keepalived* is not supported by all services (e.g. Cinder, Neutron-lbaas-agent, l3-agent), and in case of network partitioning, there is the chance that two or more nodes running *Keepalived* claim to hold the same virtual IP address, which may lead to undesired behaviour. Therefore, it is less popular than *Pacemaker*.

Based on the SDN controller and VIM concepts previously presented, in the next section we will present the main principles that need to be followed in order to design a dependable NFVO.

B. Principles to Design a Dependable NFVO

In this section, we will present a list of design principles that need to be considered in order to have a dependable NFVO. Based on the general considerations used in the design of distributed systems [72]–[74], and more specifically in NFV systems [6], [16], [28], we want to increase the awareness of the importance of designing a dependable NFVO, by analysing those general principles and mapping them to specific NFVO scenarios, which is still unexplored. Table VI summarises the relevant references concerning each of the presented principles.

1) *Avoid Single Point-of-Failure*: The most recognised and fundamental principle in dependability design is to avoid the failure of any part of the system [72]–[74], which

TABLE VI
SUMMARY OF RELEVANT REFERENCES FOR PRINCIPLES TO DESIGN A DEPENDABLE NFVO

Principle	References
Avoid Single Point-of-Failure	[6], [15], [16], [28], [72], [73] [74]
Stateless vs Stateful	[71], [75], [76], [77], [78]
Consistency vs Performance	[52], [71], [79], [80]
Distributed-Load vs Master/Slave	[81], [82], [83], [84], [85]
Failure-Independent Domains	[6], [28], [72], [73], [74]
Monitoring and Failure Detection in Controller and Orchestrator Systems	[15], [36], [43], [44], [45], [54], [55], [75]

could cause overall operation to fail. The same principle should be applied to the NFVO, and basically implies that the NFVO cannot be implemented as an individual element, but it needs to be deployed in such a robust way that a failure does not produce the unavailability of the orchestration functionalities. To this target, modularity and redundancy need to be considered. The following principles focus on different ways of achieving redundancy and modularity in relation to the architectural solutions presented in Section VI-D.

2) *Stateless vs Stateful*: As highlighted in Section VI-D, the NFVO is composed of several subsystems that are taking care of: a specific functionality (Specific module), a subset of functionalities (Aggregate module), functionalities that are associated to VIM or VNFM according to ETSI (Modified MANO), or functionalities that are integrated with other systems (Including SDN). Those subsystems/modules may have stateless or stateful properties, and consequently different strategies for providing dependability should be considered, as presented in Section V-B. For instance, there are computational operations, such as the collection of basic measurements, that do not modify the state of the NFV system. The reliability of these stateless functionalities can be guaranteed just by providing redundant computational instances and the mechanism to monitor and execute the fail-over when needed. On the other hand, there are actions like scaling in or out some VNFs or NSs. In this case, the state of the network and the respective NFVO catalogs of running services are modified. These kinds of changes need to be tracked and replicated in redundant systems that guarantee the consistency and availability of the information about the current network state at any time. The dependability of this kind of stateful subsystems/modules needs the implementation of reliable and distributed storage systems and its respective synchronisation mechanisms in addition to the redundant instances and fail-over mechanisms. In the literature there are several works that attempt to designs such mechanisms [71], [75], [76], [77], and [78]. However, as mentioned in Section V-B, this is still an open challenge.

3) *Consistency vs Performance*: In the case of a stateful NFVO subsystem/module, in order to guarantee a common view of the NFV system, dependable distributed storage systems need to be implemented. Currently, having a consistent distributed storage system is not a problem [79]. However, the processes needed for synchronising and having exactly the same image on all parts take time that may affect the performance of the NFVO system [80], e.g., the maximum number of commands executed per time unit, or the time needed to answer to and incoming requests. Depending on the frequency of the operations that change the NFV system state and the time needed for storage-image synchronisation, an appropriate balance of consistency and performance should be defined. In this context, it is important to have a clear dimension of the peak load of each of the NFVO subsystems/modules and to assess and specify a limit of the maximum allowed performance reduction.

4) *Distributed-Load vs Master/Slave*: There are two different ways to implement a redundant controlling system in order to achieve high robustness.
In the "Distributed-Load" approach (e.g [84], [85]), the cluster that constitutes the controlling system is composed of different units and each unit has the responsibility of a part of the overall system load, see Figure 15. In this approach, the fault tolerance may be provided by making the appropriate "load redistribution" in case of the failure of an unit. The advantages of this approach are the flexibility and the scalability, but since each unit acts independently, making consistent decisions based on a common global view is a challenge. This may pose a perfomance bottleneck as illustrated in Figure 15(a). A Distributed-Load approach can be implemented in two different ways. In the first, the controlled domain (e.g. managed targets) is divided in several pre-planned sub-domains, and each sub-domain is controlled by a different unit of the cluster. In this scenario, when a unit fails, the "load redistribution" is obtained by dividing the sub-domain belonging to the failed unit and assigning it to adjacent units. On the other hand, a cluster configuration can be done by using a proxy that automatically executes the load balancing according to the current system status. This kind of implementation demands special care since the availability and accessibility of the proxy as such needs to be guaranteed. For this, there are some proposals, such as Pacemaker [76], that try to address this issue.
The "Master/Slave" is a simpler approach where a single unit (master) is in charge of the controlling domain (e.g. [81]–[83]). The remaining units that compose the cluster

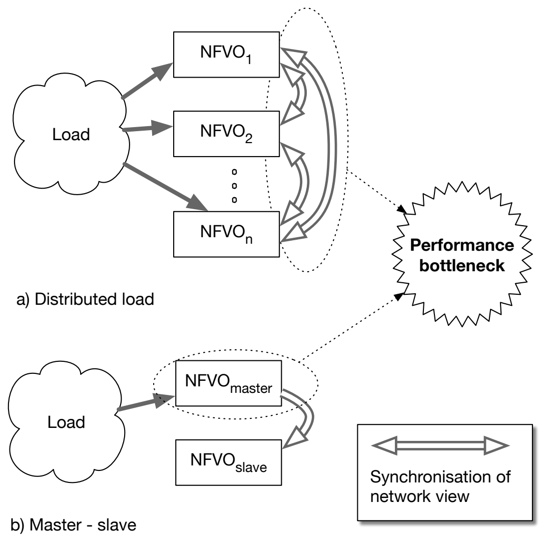

Fig. 15. Distributed-Load and Master/Slave approaches. Implementation options and corresponding performance bottlenecks.

(controlling system) are passive backup entities (slaves) with the same properties. Therefore, if the master fails, one of the slaves can immediately take responsibility of its functionalities. For the NFVO, the criteria for selecting a distributed-load or a master/slave approach will depend on the size of the domain that each of the modules need to orchestrate and to balance performance and consistency (as explained in the previous principle). Keeping the consistency in a master/slave approach is easier, since there is a single unit in charge of all the decisions. On the other hand, that single unit may not be able to handle all the load required when the size of the orchestrated domain increases, as illustrated in Figure 15(b). In this case, the use of a distributed-load approach is necessary, but at the cost of having more elaborate mechanisms that maintain consistency of the different autonomous parts of the cluster.

5) *Failure-Independent Domains*: As argued in the previous section, a failure in the NFVO ideally must not imply downtime on the operating network functions and vice versa. The traditional principle suggests that the NFVO cannot run on the same platform it manages, but under ideal circumstances, it should have a dedicated platform. However, in order to fulfil dependability requirements, the NFVO design needs to consider a wider view, by taking into account what is defined as an independent failure domain [72]–[74]. This demands complete physical and logical independence with the running functions. All the existing NFVO modules must be planned in order to avoid the propagation of their failures into the operational NFV functions, e.g., a crash failure in the module in charge of the NS lifecycle management should not affect any of the running NSs. In addition, failures external to the NFVO (NFVI, VNF, NS, etc) should not have the capacity to affect it, e.g., a group of alarm storms should not reduce the performance of the NFVO below an acceptable level.

6) *Monitoring and Failure Detection in Controller and Orchestrator Systems*: Finally, in order to take advantage of the previously presented principles, it is important to have monitoring tools for the NFVO. As explained in Section V, techniques used may include passive, active, centralised, or/and distributed features. Given the potential multi-module architecture of the NFVO, an alternative would be to include a distributed monitoring where each of the modules check among each other in an mechanisms similar to the used by ZooKeeper [75], complemented by some centralised monitoring features. In addition, Section VII-C shows the importance of having correctness on the orchestration operations which may incorporate smart methods that combine active and passive monitoring. The specific techniques used to monitor the NFVO will depend on its specific architecture, taking into account its modularity and operation correctness.

IX. Summary of Challenges

In this paper, the dependability requirements and challenges associated with the introduction of NFV are discussed with the main focus on orchestration. The corresponding ETSI standards and the related work are reviewed together with ongoing implementation efforts. In addition, specific design considerations related to fault-tolerant NFVO are presented and discussed. Our main observations can be summarised as follows:

- There is currently a lack of research on how to ensure the dependability of NFV-based services.
- While significant effort is directed towards ensuring the dependability of the network functions themselves, little attention is paid to the dependability of the MANO.
- Misoperation of the management system, especially the NFVO, may have severe or catastrophic consequences for the entire network and services it provides. Thus, the management system should be designed to have failure omission semantics, yet this is currently not on the research and standardisation agenda.
- As fault tolerance and fault management in general are 'orthogonal' to service provisioning, the mechanisms ensuring fault tolerance in the system should be well defined.
- NFV standardisation specifies a modular approach to independent development and provisioning of functional modules for NFV. Fault tolerance and fault management should support the modular approach and coherent failure handling simultaneously. At the same time, this might lead to a rigid platform, making it more challenging to embrace future resilience provisioning technologies.
- The NFVO is a critical element with respect to the dependability that may be provided by NFV. Great care should be taken in the fault-tolerant design of this element, and means not found in "off-the-shelf" fault-tolerant systems should be considered.

In addition to these specific issues, it is observed that due to flexibility and fine-grained control over resources, NFV-based systems have the capability to withstand frequent and simple failures better than traditional systems. However, higher complexity and logical centralisation of NFV-based systems may increase the likelihood and consequences of failures of the management system considerably. Thus, more effort should be made to prevent and deal with failures effectively. Finally, we expect this paper motivates the further research needed to find specific technical solutions to the identified challenges.

REFERENCES

[1] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "Network Function Virtualisation (NFV) - Network Operator Perspectives on Industry Progress," *Tech. Rep.*, Oct. 2013. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[3] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, "How will NFV/SDN Transform Service Provider OPEX?" *IEEE Network*, vol. 29, no. 3, pp. 60–67, 2015.

[4] A. Gonzalez, P. Grønsund, K. Mahmood, B. E. Helvik, P. Heegaard, and G. Nencioni, "Service Availability in the NFV Virtualized Evolved Packet Core," in *2015 IEEE Global Communications Conference: Communication QoS, Reliability and Modeling*, R. Rao, M. Sarkar, and M. Song, Eds. IEEE press, 6 - 10 December 2015.

[5] P. Heegaard, B. Helvik, and V. Mendiratta, "Achieving Dependability in Software-Defined Networking - A Perspective," in *2015 7th International Workshop on Reliable Networks Design and Modeling (RNDM)*, Oct 2015, pp. 63–70.

[6] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 152–157, 2017.

[7] V. G. Nguyen, A. Brunstrom, K. J. Grinnemo, and J. Taheri, "SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1567–1602, thirdquarter 2017.

[8] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "ETSI GS NFV-MAN 001 V1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration," *Tech. Rep.*, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsigs/NFV-MAN/001099/001/01.01.0160/gsNFV-MAN001v010101p.pdf

[9] ——, "ETSI GS NFV 002 V1.2.1: Network Functions Virtualisation (NFV); Architectural Framework," *Tech. Rep.*, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.02.01_60/gs_nfv002v010201p.pdf

[10] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[11] J. P. Sterbenz and D. Hutchison. (2017, Mar.) ResiliNets: Resilient and Survivable Networks. [Online]. Available: https://wiki.ittc.ku.edu/resilinets

[12] P. E. Heegaard and K. S. Trivedi, "Network Survivability Modeling," *Comput. Netw.*, vol. 53, no. 8, pp. 1215–1234, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.02.014

[13] G. Pongracz, L. Molnar, and Z. L. Kis, "Removing Roadblocks from SDN: OpenFlow Software Switch Performance on Intel DPDK," *2013 Second European Workshop on Software Defined Networks. EWSDN 2013*, Oct. 2013.

[14] M. Kourtis, G. Xilouris, V. Riccobene, M. J. McGrath, G. Petralia, H. Koumaras, G. Gardikis, and F. Liberal, "Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration," in *IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2015, San Francisco, CA, USA, November 18-21, 2015*, 2015, pp. 74–78.

[15] X. Ju, L. Soares, K. G. Shin, K. D. Ryu, and D. Da Silva, "On Fault Resilience of OpenStack," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SOCC '13, 2013, pp. 2:1–2:16.

[16] D. Cotroneo, L. D. Simone, A. K. Iannillo, A. Lanzaro, and R. Natella, "Dependability evaluation and benchmarking of network function virtualization infrastructures," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, April 2015, pp. 1–9.

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

[18] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG). NFV Proofs of Concept. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv/nfv-poc

[19] ——. (2014) Proof of Concept: E2E vEPC Orchestration in a multi-vendor open NFVI environment. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=E2E_vEPC_Orchestration_in_a_multi-vendor_open_NFVI_environment

[20] ——. (2014) Proof of Concept: Automated Network Orchestration. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Automated_Network_Orchestration

[21] ——. (2015) Proof of Concept: Service orchestration for virtual CDN service over distributed cloud management platform. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Service_orchestration_for_virtual_CDN_service_over_distributed_cloud_management_platform

[22] ——. (2014) Proof of Concept: Demonstration of High Reliability and Availability aspects in a Multivendor NFV Environment. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Demonstration_of_High_Reliability_and_Availability_aspects_in_a_Multivendor_NFV_Environment

[23] ——. (2015) Proof of Concept: Availability Management with Stateful Fault Tolerance. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Availability_Management_with_Stateful_Fault_Tolerance

[24] ETSI OSM. (2017, Mar.) An Open Source NFV Management and Orchestration Software Stack. [Online]. Available: https://osm.etsi.org

[25] The OPEN-O Project. (2017, Mar.) OPEN-O: Open Orchestrator Project. [Online]. Available: https://www.open-o.org

[26] G. Carella. (2017, Mar.) OpenBaton: The Open Source Network Function Virtualization Orchestrator (NFVO). [Online]. Available: http://openbaton.org

[27] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "Draft ETSI GR NFV-REL 007 V0.0.8: Network Functions Virtualisation (NFV); Report on the Resiliency of NFV-MANO Critical Capabilities; MANO Resiliency," *Preliminary; Work in progress*, Jul. 2017.

[28] ——, "ETSI GS NFV-REL 001 V1.1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," *Tech. Rep.*, Jan. 2015. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_nfv-rel001v010101p.pdf

[29] ——, "ETSI GS NFV-REL 002 V1.1.1: Network Functions Virtualisation (NFV); Reliability; Report on Scalable Architectures for Reliability Management," *Tech. Rep.*, Sep. 2015. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/002/01.01.01_60/gs_nfv-rel002v010101p.pdf

[30] F. Cristian, "Understanding Fault-Tolerant Distributed Systems," *Commun. ACM*, vol. 34, no. 2, pp. 56–78, Feb. 1991. [Online]. Available: http://doi.acm.org/10.1145/102792.102801

[31] T. D. Nielsen and F. V. Jensen, *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[32] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "ETSI GS NFV-INF 003 V1.1.1: Network Functions Virtualization (NFV); Infrastructure; Compute Domain," *Tech. Rep.*, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/003/01.01.01_60/gs_nfv-inf003v010101p.pdf

[33] ——, "ETSI GS NFV-INF 005 V1.1.1: Network Functions Virtualization (NFV); Infrastructure; Network Domain," *Tech. Rep.*, Dec. 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/005/01.01.01_60/gs_nfv-inf005v010101p.pdf

[34] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an SDN-enabled NFV architecture," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 187–193, April 2015.

[35] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "ETSI GS NFV-REL 003 V1.1.1: Network Functions Virtualization (NFV); Reliability; Report on Models and Features for End-to-End Reliability," *Tech. Rep.*, Apr. 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/003/01.01.01_60/gs_NFV-REL003v010101p.pdf

[36] ——, "ETSI GS NFV-REL 004 V1.1.1: Network Functions Virtualization (NFV); Assurance; Report on Active Monitoring and Failure Detection," *Tech. Rep.*, Apr. 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/004/01.01.01_60/gs_NFV-REL004v010101p.pdf

[37] ETSI. (2015) Demonstration high availability vEPC and SDN controlled service chain. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Demonstration_high_availability_vEPC_and_SDN_controlled_Service_Chain

[38] ——. (2015) Demonstration of virtual EPC (vEPC) applications and enhanced resource management. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=Demonstration_of_Virtual_EPC_%28vEPC%29_Applications_and_Enhanced_Resource_Management

[39] ——. (2015) VoLTE service based on vEPC and vIMS architecture. [Online]. Available: https://nfvwiki.etsi.org/index.php?title=VoLTE_Service_based_on_vEPC_and_vIMS_Architecture

[40] F. Firmin. The evolved packet core. 3GPP MCC. [Online]. Available: http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core

[41] H. Khedher, E. Abd-Elrahman, H. Afifi, and M. Marot, "Optimal and cost efficient algorithm for virtual CDN orchestration," in *2017 IEEE 42nd Conference on Local Computer Networks*, 2017, pp. 61 – 69.

[42] Zabbix. (2017, Mar.) The Zabbix Monitoring Platform. [Online]. Available: http://www.zabbix.com

[43] Nagios. (2017, Mar.) The Nagios Monitoring Software. [Online]. Available: https://www.nagios.com

[44] Zenoss. (2017, Mar.) Zenoss Monitoring Tools. [Online]. Available: https://www.zenoss.com

[45] Zabbix. (2017, Mar.) Hierarchical Distributed Monitoring Based on Zabbix. [Online]. Available: https://www.zabbix.com/documentation/1.8/manual/distributed_monitoring

[46] VMWare. (2017, Mar.) VMware HA: Deployment Best Practices. White Paper. [Online]. Available: http://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/vmw-server-wp-bestpractices-white-paper.pdf

[47] VMware. (2009) Protecting Mission-Critical Workloads with VMware Fault Tolerance. White Paper. [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/ft_virtualization_wp.pdf

[48] Verizon. (2017, Mar.) SDN-NFV Reference Architecture. [Online]. Available: http://innovation.verizon.com/content/dam/vic/PDF/Verizon_SDN-NFV_Reference_Architecture.pdf

[49] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, Oct. 2007.

[50] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: High Availability via Asynchronous Virtual Machine Replication," in *5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 161–174.

[51] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live Migration of Virtual Machine Based on Full System Trace and Replay," in *18th ACM International Symposium on High Performance Distributed Computing*. ACM, 2009, pp. 101–110.

[52] A. Singh, P. Fonseca, P. Kuznetsov, R. Rodrigues, and P. Maniatis, "Zeno: Eventually Consistent Byzantine-Fault Tolerance," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 169–184.

[53] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "ETSI GS NFV-IFA 005 V2.1.1: Network Functions Virtualization (NFV); Management and Orchestration; Or-Vi reference point - Interface and Information Model Specification," *Tech. Rep.*, Apr. 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/005/02.01.01_60/gs_NFV-IFA005v020101p.pdf

[54] VMware. (2017, Mar.) VMware vRealize Hyperic. [Online]. Available: http://www.vmware.com/products/vrealize-hyperic.html

[55] Google. (2017, Mar.) Stackdriver Monitoring. [Online]. Available: https://cloud.google.com/monitoring

[56] A. J. Gonzalez and B. E. Helvik, "System Management to Comply with SLA Availability Guarantees in Cloud Computing," in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, Dec 2012, pp. 325–332.

[57] ——, "Hybrid Cloud Management to Comply Efficiently with SLA Availability Guarantees," in *2013 12th IEEE International Symposium on Network Computing and Applications (NCA)*. IEEE, 2013, pp. 127–134.

[58] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers," *Journal of Network and Computer Applications*, vol. 52, pp. 11–25, 2015.

[59] A. J. Gonzalez, B. E. Helvik, P. Tiwari, D. M. Becker, and O. J. Wittner, "Gearshift: Guaranteeing availability requirements in slas using hybrid fault tolerance," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 1373–1381.

[60] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG), "ETSI GS NFV-IFA 010 V2.1.1: Network Functions Virtualisation (NFV); Management and Orchestration; Functional requirements specification," *Tech. Rep.*, Apr. 2016. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/010/02.01.01_60/gs_NFV-IFA010v020101p.pdf

[61] NGMN Alliance, "5G White Paper," *Next Generation Mobile Networks*, Feb. 2015. [Online]. Available: https://www.ngmn.org/5g-white-paper.html

[62] 5GPPP, "Vision on Software Networks and 5G," *White Paper - Sorfware Networks WG*, Feb. 2015. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP_SoftNets_WG_whitepaper_v20.pdf

[63] 3GPP. Service and System Aspects (SA) Working Group 5. [Online]. Available: http://www.3gpp.org/Specifications-groups/sa-plenary/56-sa5-telecom-management

[64] 3rd Generation Partnership Project (3GPP) Technical Specification Group (TSP) Services and System Aspects (SA), "3GPP TS 28.500 V14.1.0: Telecommunication management; Management concept, architecture and requirements for mobile networks that include virtualized network functions," *Tech. Spec.*, Mar. 2017.

[65] ——, "3GPP TS 28.515 V14.1.0: Telecommunication management; Fault Management (FM) for mobile networks that include virtualized network functions; Requirements," *Tech. Spec.*, Jun. 2017.

[66] AT&T Inc. (2017, Mar.) ECOMP (Enhanced Control, Orchestration, Management & Policy) Architecture White Paper. [Online]. Available: http://about.att.com/content/dam/snrdocs/ecomp.pdf

[67] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt

[68] W.-L. Yeow, C. Westphal, and U. C. Kozat, "Designing and Embedding Reliable Virtual Infrastructures," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 57–64, Apr. 2011.

[69] V. Medina and J. M. García, "A Survey of Migration Mechanisms of Virtual Machines," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 30:1–30:33, Jan. 2014.

[70] Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG). (2017, Mar.) Proof of Concept: Demonstration of Multi-Location, Scalable, Stateful Virtual Network Function. [Online]. Available: http://nfvwiki.etsi.org/index.php?title=Demonstration_of_multi-location,_scalable,_stateful_Virtual_Network_Function

[71] A. J. Gonzalez, G. Nencioni, B. E. Helvik, and A. Kamisiński, "A Fault-Tolerant and Consistent SDN Controller," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[72] F. Cristian, "Understanding fault-tolerant distributed systems," *Communications of the ACM*, vol. 34, no. 2, pp. 56–78, 1991.

[73] A. S. Tanenbaum and M. v. Steen, *Distributed Systems: Principles and Paradigms*. Prentice-Hall, Inc., 2006.

[74] E. Bauer and R. Adams, *Reliability and Availability of Cloud Computing*. Wiley-IEEE Press, 2012.

[75] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-Free Coordination for Internet-Scale Systems," in *USENIX Annual Technical Conference*, vol. 8, 2010.

[76] The OpenStack Project. (2017, Mar.) OpenStack: The Pacemaker Architecture. [Online]. Available: http://docs.openstack.org/ha-guide/intro-ha-arch-pacemaker.html

[77] T. D. Chandra, R. Griesemer, and J. Redstone, "Paxos made live: An engineering perspective," in *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '07, 2017, pp. 398–407.

[78] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner *et al.*, "F1: A distributed SQL database that scales," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1068–1079, 2013.

[79] A. S. Rawat, O. O. Koyluoglu, N. Silberstein, and S. Vishwanath, "Optimal locally repairable and secure codes for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 212–236, 2014.

[80] J. Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. a. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-recovery for middleboxes," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, ser. SIGCOMM '15. New York, NY, USA: ACM, 2015, pp. 227–240. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787501

[81] F. A. Botelho, A. N. Bessani, F. M. V. Ramos, and P. Ferreira, "SMaRtLight: A Practical Fault-Tolerant SDN Controller," *CoRR*, vol. abs/1407.6062, 2014. [Online]. Available: http://arxiv.org/abs/1407.6062

[82] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller Fault-Tolerance in Software-Defined Networking," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. ACM, 2015.

[83] V. Pashkov, A. Shalimov, and R. Smeliansky, "Controller Failover for SDN Enterprise Networks," in *Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), 2014 International*. IEEE, 2014.

[84] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "ONIX: A Distributed Control Platform for Large-scale Production Networks," in *OSDI*, vol. 10, 2010, pp. 1–6.

[85] A. Tootoonchian and Y. Ganjali, "HyperFlow: A Distributed Control Plane for OpenFlow," in *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, 2010.

[86] W. A. Lambeth, T. Koponen, and M. Casado, "Distributed Network Control System with One Master Controller per Logical Datapath Set," Jun 2016, US Patent 9,363,210.

[87] O. Sefraoui, M. Aissaoui, and M. Eleuldj, "OpenStack: Toward an Open-Source Solution for Cloud Computing," *International Journal of Computer Applications*, vol. 55, no. 3, 2012.

[88] The OpenStack Project. (2017, Mar.) OpenStack: HAProxy. [Online]. Available: http://docs.openstack.org/ha-guide/controller-ha-haproxy.html

[89] ——. (2017, Mar.) OpenStack: The Keepalived Architecture. [Online]. Available: http://docs.openstack.org/ha-guide/intro-ha-arch-keepalived.html

Andres Gonzalez is Research Scientist at Telenor Group. He received his PhD in Telecommunications from the Norwegian University of Science and Technology NTNU in 2013, and his Master in Telecommunications Engineering from The National University of Colombia and the Vienna University of Technology in 2008. Since 2013 he joined Telenor, working in Research projects and Proof of Concepts towards 5G with a main focus on NFV, SDN, and Network Slicing, which are his main current research interest, as well as the analysis of dependability issues in emerging network technologies, analysis and simulation of fault tolerant systems, dependability modeling, failure data processing and analysis, and reliable networks design.

Gianfranco Nencioni received the M.Sc. in telecommunication engineering and the Ph.D. in information engineering from the University of Pisa (Italy) in 2008 and 2012, respectively. In the fall of 2011, he was a visiting Ph.D. student with the Computer Laboratory, University of Cambridge (UK). From 2012 to 2015, he was a Postdoctoral Fellow at the University of Pisa. From 2015 to 2018, he was a Postdoctoral Fellow at NTNU (Norway). He is now an Associate Professor at the University of Stavanger (Norway). His past research activity has regarded energy-aware routing and design in both wired and wireless networks and dependability on SDN and NFV. His current research activity regards orchestration and control in Software-Defined 5G Radio Access and Mobile Core Networks.

Andrzej Kamisiński is an Assistant Professor in the Department of Telecommunications at the AGH University of Science and Technology in Kraków, Poland. He received his B.Sc., M.Sc. (with honors), and Ph.D. (with distinction) degrees in telecommunications from the same University in 2012, 2013, and 2017, respectively. In October 2015, he joined an international research team for four months as a visiting Ph.D. student in the Department of Information Security and Communication Technology (formerly Department of Telematics), Norwegian University of Science and Technology (NTNU), where he worked on dependability of Software-Defined Networks. Since April 2018, as a member of the Management Committee of the "Resilient Communication Services Protecting End-user Applications from Disaster-based Failures" European COST Action (European Cooperation in Science and Technology), he is involved in international research on dependability and security of computer and communication networks in the context of simultaneous failures of multiple network elements.

His primary research interests span dependability, programmability, and security of computer and communication networks. In particular, he is interested in fast network recovery strategies.

Bjarne E. Helvik (1952) received his Siv.ing. degree (MSc in technology) from the Norwegian Institute of Technology (NTH), Trondheim, Norway in 1975. He was awarded the degree Dr. Techn. from NTH in 1982. He has since 1997 been Professor at the Norwegian University of Science and Technology (NTNU) and is currently with the Department of Information Security and Communication Technology. In the period 2009 – 2017, he was Vice Dean with responsibility for research at the Faculty of Information Technology and Electrical Engineering at NTNU, and in 2003 – 2012 Principal Investigator at the Norwegian Centre of Excellence Q2S - the Centre for Quantifiable Quality of Service in Communication Systems. He has previously held various positions at ELAB and SINTEF Telecom and Informatics. In the period 1988-1997 he was appointed as Adjunct Professor at the Department of Computer Engineering and Telematics at NTH.

His field of interests includes QoS, dependability modelling, measurements, analysis and simulation, fault-tolerant computing systems and survivable networks, as well as related system architectural issues. His current research focus is on ensuring dependability in services provided by multi-domain, virtualised ICT systems.

Poul E. Heegaard (IEEE SM'14) is Professor at the Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU). Since 2006, Heegaard has been a faculty member at NTNU, and Head of Department in the period 2009-2013. From 1989-1999 he was Research Scientist and Senior Scientist at SINTEF, and from 1999 to 2009 a Senior Research Scientist at Telenor R&I. He is now Principal Investigator in the CINELDI (Centre for Intelligent Electricity Distribution), a Centre for Environment-friendly Energy Research (FME) where he is responsible for smart grid operation.

His research interests include performance, dependability and survivability assessment, with focus on communication networks (such as 5G, NFV, SDN) and services, and communication system as part of a digital ecosystem, including interaction with other critical infrastructures (such as Smart Grid).