Doctoral theses at NTNU, 2021:248

Besmir Tola

Dependability Modeling, Analysis, and Provisioning of NFV-Supported Services

Doctoral thesis

Besmir Tola

ISBN 978-82-326-6410-8 (printed ver.)
ISBN 978-82-326-6186-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2021:248

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical
Engineering
Dept. of Information Security and
Communication Technology

NTNU
Norwegian University of
Science and Technology

Besmir Tola

Dependability Modeling, Analysis, and Provisioning of NFV-Supported Services

Thesis for the Degree of Philosophiae Doctor

Trondheim, July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ISBN 978-82-326-6410-8 (printed ver.)
ISBN 978-82-326-6186-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2021:248

Printed by NTNU Grafisk senter

Abstract

The increasing network traffic demands, stemming from an ever increasing number of devices connected to the network, have gradually accentuated the limits of current Internet networks, also known as Internet ossification, and innovation or further development has become almost impossible. One important cause of this limitation is the ubiquitous deployment of middleboxes (or network functions) that hamper the network capability to be flexible, scalable and innovative to an extent that new and specialized services cannot be easily introduced in the network.

Network virtualization promises to overcome the current ossified state of Internet network and is anticipated to revolutionize the design and operation of today's network infrastructures. Network Function Virtualization (NFV) is acknowledged as a crucial enabler of this transformation which promises to develop a more flexible, agile, and programmable networking paradigm that will help reduce both CAPEX and OPEX costs, and time to introduce new services. Inspired by the success of server virtualization and cloud computing, top major telecom providers conceived the NFV paradigm for enabling a major transformation of modern telecommunication networks, such as 5G.

NFV provides the ability to execute virtual instances of networking devices on top of a common physical network substrate. It utilizes virtualization technology to reduce dependency on underlying hardware by moving data processing tasks from proprietary hardware middleboxes to virtualized entities that can run on commodity hardware. NFV simplifies network infrastructure by exploiting standardized and commodity hardware for both compute and networking; introducing the benefits of agility, flexibility, and scalability of data centers to network infrastructures. Together with Service Function Chaining, it enables the replacement of traditional network hardware appliances by softwarized Virtualized Network Function (VNF)s chains. However, this major transformation brings additional challenges and one is them is the ability to ensure high availability, as an important dependability attribute, of carrier-grade services provided by NFV-enabled networks. This challenge is further exacerbated by the extreme availability demands that 5G use cases demand, i.e., 99.999% or higher availability figures. This thesis work targets this challenge by addressing the problem of how to assess and quantify the availability of NFV-supported network services, and how to provision highly available NFV services by means of fault-tolerant mechanisms.

First, this thesis contributes with the design and evaluation of a set of stochastic availability models that can abstract the functional behavior of the system components that are involved in the provisioning of NFV services. The models represent the virtualized network functions, the underlying hardware infrastructure, the chaining of several VNFs, and their management and orchestration (MANO) plane. Moreover, distinct models implement different fault-tolerance mechanisms, enhanced with specific recovery strategies, allowing to estimate the behavior of the availability metric for each redundancy configuration. The analysis result shows that VNF software can be a critical element and sufficient redundancy needs to be allocated if carrier-grade availability figures are to be expected. Moreover, a less robust MANO can significantly degrade the service availability but on the contrary, a highly redundant MANO does not bring additional benefits.

Second, network flexibility, as one of the main benefits introduced by the adoption of the NFV networking paradigm, regards the capability to deploy on-the-fly and run VNFs anywhere in the network substrate. Thus, service function chains, i.e., an order set of functions, can be composed of VNFs which in turn can be distributed in the network. Therefore, it is required that traffic flows are to be steered through all the VNFs that make up a specific function chain. As a result, the availability evaluation and assessment of an end-to-end network service shall involve also networking elements that are not necessarily part of the NFV infrastructure but vital to the VNF interconnection, such as routers, switches and network links. These elements are often disregarded in the related literature and this thesis develops a methodology for modeling and quantifying the availability of end-to-end network services by integrating all the engaged elements. Moreover, the models also integrate a Software-defined Networking (SDN) approach, as an NFV complementary technology. Extensive sensitivity analysis helped to identify availability bottlenecks for both traditional and SDN-integrated NFV network services. Results show that IP routers can represent a threatening availability bottleneck despite VNFs are enriched with redundancy.

Another aspect that is covered in this work is the provisioning of redundant resources for guaranteeing service availability demands under different system constraints such as limited resource capacity, heterogeneous equipment, or service request requirements. In addition to the adequate redundancy level, ensuring that service availability demands are met requires also the knowledge of a set of policies that ultimately decide where, how many, and what type of redundant function instances shall be allocated in the network infrastructure. This is referred to as the availability-aware NFV resource allocation problem and this work formulates it as an Integer Linear Programming (ILP) optimization problem aiming at minimizing resource utilization while still satisfying service availability and performance requirements. Two distinct ILP problems are developed, namely *AllOne* and *AllAny*, which optimally place redundant functions and perform routing of traffic flows. Given the $\mathcal{NP}$-hard nature of the problem, although the two formulations give optimal solutions, they do not scale well for large problem instances. To address this limitation, this work also proposes a scalable heuristic algorithm which can provide near-optimal solution in polynomial time also for large problem instances. The

algorithm, coined *CoShare*, decides the required number of backup instances, and efficiently places them by avoiding the simultaneous unavailability of working and backup service chains, which can happen due to network structural dependencies. In addition, *CoShare* exploits a shared reservation principle, in which instance capacity is shared among multiple flows for redundancy purpose. The numeric evaluation shows that the algorithm can achieve better resource efficiency, i.e., lower additional amount of redundant resources, compared to previous literature while at the same time satisfy flow's availability demands.

To summarize, this thesis contributes with models that enable the assessment and evaluation of the availability of end-to-end NFV-supported network services, performs extensive analysis aiming at identifying critical components and advisable redundancy configurations, and proposes a set of algorithms that efficiently provide and orchestrate network resources by allocating redundant functions aiming at fulfilling availability demands of service requests in NFV-enabled networks.

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at NTNU-Norwegian University of Science and Technology. The presented work was carried out at the Department of Information Security and Communication Technology (IIK), Trondheim, under the supervision of Professor Yuming Jiang and the co-supervision of Professor Bjarne E. Helvik. The PhD position has been partially funded by the EU FP7 Marie Curie Actions of the EC Seventh Framework Programme (FP7/2007-2013) under the Grant Agreement No. 607584 (The CleanSky ITN Project).

Acknowledgements

First of all, with deep sense of gratitude, I thank my supervisor Professor Yuming Jiang, for the opportunity that he has given to me in pursuing a PhD career. He has been a great source of inspiration and his dedicated support and guidance has helped me in all the time of research and writing of this thesis. I am extremely grateful to my co-supervisor Professor Bjarne E. Helvik, whose expertise has been invaluable in the topics that this work covers. His insightful feedback pushed me to sharpen my thinking and brought my research work to a higher level. Sincere thanks are also due to my co-authors, Asc. Prof. Gianfranco Nencioni and Prof. K.K. Ramakrishnan for the fruitful discussions and collaborations. In particular, I sincerely thank my office mate and partner in crime Dr. Yordanos T. Woldeyohannes for invaluable academic and personal growth.

A special thank goes also to all the colleagues at IIK. Randi, Mona and Laurent for all the help with administrative tasks. Pål Sæther, a.k.a. Mr. Wolf – the problem solver, for helping me solve technical problems and if not, a good beer(s) with him would make me forget them. And of course, thanks to Katina, Danilo, Michele, David, Marija, Ruxandra, Peach, Kalpanie, Mattia and Faiga, who made my time spent at the department joyous and something I will always treasure. My heartfelt gratitude goes to Romina, Ergys, and Endri, great friends whose friendship and support has made this journey full of wonderful memories.

I would like to express my thanks towards colleagues and members of the CleanSky ITN group, especially Alessio, Nitinder, David, Mayutan, Abhi and Peter (Pengyuan), who made me look forward to the research monotony breaking travels that were periodically organized within the CleanSky group. Many thanks goes also to my hosts during my two research visits at Nokia Bell Labs, Stuttgart, Dr. Volker Hilt, for being an excellent host, and at UNINETT, Dr. Otto J. Wittner, for the great time I had at UNINETT, inspiring talks about research, and for the opportunity given to me in teaching his course.

I cannot begin to express my thanks to Marta. You have always been so patient and incredibly supportive to me in every aspect of the PhD journey. I deeply thank you for your love and belief in me, and for always being there for me. Gracias, Juez más guapa del mundo!

Finally, I would like to thank my parents, my sister and brothers for their love, help and much-valuable support throughout my PhD. They selflessly encouraged me to explore new directions in life and seek my own destiny. This journey would not have been possible if not for them, and I dedicate this milestone to them.

Table of Content

List of Figures

List of Tables

List of Acronyms

CAPEX/OPEX	Capital Expenditure/Operating Expenditure
COA	Capacity Oriented Availability
COTS	Commercial-off-the-shelf
D/CTMC	Discrete/Continuous Time Markov Chain
ETSI	European Telecommunications Standards Institute
FT	Fault Tree
ICT	Information and Communication Technologies
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
ISG	Industry Specification Group
ITU	International Telecommunication Union
LB	Load Balancer
MRM	Markov Reward Model
MTTF	Mean Time to Fail
MTTR	Mean Time to Repair
MUT	Mean Uptime
MUT	Mean Downtime
M&O	Management And Orchestration
NAT	Network Address Translation
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFV-MANO	NFV Management and Orchestration
NFVO	Network Function Virtualization Orchestrator
PoC	Proof of Concept
RBD	Reliability Block Diagram
RG	Reliability Graph
SAN	Stochastic Activity Network
SFC	Service Function Chaining
SLA	Service Level Agreement
SPN	Stochastic Petri Net
SRN	Stochastic Reward Network
TSP	Telecommunication Service Provider
URLLC	Ultra-Reliable Low-Latency Communications

vEPC	virtualized Evolved Packet Core
vIMS	virtualized IP Multimedia Subsystem
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VMM	Virtual Machine Manager
VNE	Virtual Network Embedding
VNF	Virtual Network Function
VNFM	Virtual Network Function Managers
VNF-FG	Virtual Network Function Forwarding Graph

Part I

Summary

Chapter 1
Introduction

1.1 Thesis Outline

The format of the present thesis is a collection of articles, which is in line with NTNU rules for the doctoral degree, and its content is divided into two main parts:

- **Part I: Summary**
- **Part II: Included Articles**

Part I presents a comprehensive summary of the thesis. It consists of the following chapters:

- The *Introduction* chapter (Chapter 1) illustrates the motivation for the research work and the focus of this thesis. In addition, it highlights the research questions and objectives together with the applied research methodology.
- The *Background* chapter (Chapter 2) gives the essential background for understanding the research scope and the contributions of the thesis. It also introduces the availability modeling approach and the context of availability-aware resource provisioning in NFV-enabled networks.
- The *Related Work* chapter (Chapter 3) reviews the state-of-the-art literature and works related to the challenges and problems that the thesis contributions tackle. Such challenges, which are tightly coupled with the research objectives of the thesis, are listed in the remainder of this chapter.
- The *Contributions and Concluding Remarks* chapter (Chapter 4) presents the paper contributions obtained during the PhD investigation period and summarizes the concluding remarks followed by suggestion for future work.

Part II consists of 6 papers, which represent the contribution of the thesis work, where 5 are published and 1 is currently submitted for peer-reviewed publication.

In addition to the first two parts, there is also **Part III** which illustrates a brief summary of secondary publications that are not included as contribution to this thesis.

1.2 Motivation and Focus

Today's communication networks include a plethora of network appliances, also called middleboxes, for providing different kinds of network functions in relation to security, performance, and/or other specialized policies within a network infrastructure [1]–[3]. Examples of such middleboxes include firewalls, Intrusion Detection Systems (IDSs), Network Address Translators (NATs), and Load Balancers (LBs). The number of employed middleboxes in modern communication networks is almost as high as the number of standard router devices [2], [4]. Although middleboxes have become an integral part of communication network infrastructures, they are typically expensive both in terms of investment and operation [4], i.e., Capital Expenditure (CAPEX) and Operational Expenditure (OPEX). Moreover, they are usually closed systems with little or no possibilities to enable innovation. Each middlebox typically performs a narrow specialized function and is designed for a particular choice of hardware platform, which makes it difficult and challenging for network operators to introduce and deploy new services. Frequently, network operators are obliged to purchase new hardware or consider new vendors in case they need to extend network capability or add new functionality to an existing middlebox. This may require changes in the deployment strategy, assessment of new hardware, and may lead to an increase of cost and time required to introduce new services.

Another limitation of the traditional deployment of middleboxes arises from specialized network services that require traffic steering among several middleboxes. It is common that traffic flows may be required to go through a chain of network functions (i.e., middleboxes) like a firewall, an IDS, and finally through a proxy [5]. This mechanism is referred to as service function chaining (SFC) and the Internet Engineering Task Force (IETF) specifies it as "the definition and instantiation of an ordered set of service functions and subsequent steering of traffic through them" [6]. In current network settings, traffic flow routes are manually set up for some desired sequence of middleboxes [5], which can be cumbersome and error-prone in large scale infrastructures. In addition, middleboxes are deployed in fixed positions which limits traffic routing paths from an efficient utilization of the available network resources, hence making the middleboxes potential bottlenecks in the network.

A fast-emerging and prominent solution that promises to alleviate these limitations is Network Function Virtualization (NFV) [7], [8]. In late 2012, under the common efforts of seven of the leading Telecommunication Companies (telecoms), the European Telecommunications Standards Institute (ETSI) established an industry specification group for defining and developing NFV. The basic idea was that by exploiting server virtualization, a technology that makes the fortune of cloud-computing, NFV performs the decoupling of the network appliance software from purpose-built hardware and runs it in virtualized environments, which can be deployed on a range of industry standard server hardware, otherwise called commercial-off-the-shelf (COTS) servers. This way, virtualizing network functions (VNFs) offers many benefits such as reduced equipment costs, through consolidation and exploitation of COTS hardware, and introduces greater flexibility in deploying and operating network functions. The deployment

of VNFs allows sharing of physical resources across many services and customer bases. VNFs can be created on-the-fly and dynamically chained together to provide service chains for innovative and more advanced services. Additionally, VNFs can be deployed anywhere on the network and an operator can optimize the location of VNFs, so that network resources are efficiently utilized and service level agreements (SLA) [9] can still be satisfied. As a result, NFV can help, among others, increase flexibility in the provisioning of network services, simplify network management, increase scalability, and reduce CAPEX and OPEX costs, together with the reduced time to introduce new services. However, the "softwarization" of hardware-specific middleboxes poses several challenges and service *dependability* – as the ability to deliver service that can justifiably be trusted [10], represents a major concern that can undermine the success of NFV adoption [11]–[14]. For the widespread adoption of NFV, it is important that service providers can guarantee at least the same level of dependability compared to traditional specialized hardware-based appliances, which have, through years of development, grown to mature and dependable technologies.

Dependability is often announced as a unifying term integrating attributes like availability, reliability, safety, integrity, and maintainability [10], [15]. Alternatively, it is also defined as the ability to avoid service failures that are more frequent and more severe than acceptable. Although there is no unique definition of dependability, it is commonly agreed that it consists of the above set of attributes, which are subject to different threats, i.e., faults, errors, and failures, and can exploit various means for achieving the attributes [10], [15], [16]. Some of these attributes are quantitative (e.g., availability and reliability) while some are qualitative (e.g., safety). The importance of one dependability attribute over another depends on the application under consideration. Focusing on communication networks and the services provided by them, the availability attribute is of utmost importance. The International Telecommunication Union (ITU) framework for service level agreements (SLA) identifies service availability as the most important dependability attribute for end users, which has to be clearly defined in an SLA [17], [18]. In addition, it is more common that end-users are mostly interested in a running service, i.e. available, when they want to make use of it. Moreover, availability is a common attribute to the different network performance concepts such as dependability, security, survivability, and fault-tolerance [16].

There are several concerns that make availability a critical design factor in NFV. An important concern raises from the fact that legacy network appliances, enriched with built-in fault management mechanisms that reach “5-nines” standards, are replaced by COTS data-center hardware whose failure intensities are potentially higher than traditional purpose-built hardware [12], [13], [19]. Also, software code developed for implementing virtualized network functions is still at their infancy and may be less robust and more error-prone [13]. In addition, utilizing a virtualization layer comes at the cost of increased system dynamics caused by the introduction of virtual resources and the lack of direct control over the underlying physical hardware [20]. The benefit of efficient resource utilization relies on services sharing a common physical infrastructure and thus any eventual abnormal execution of applications, e.g., resource overload, may lead to availability issues for third party services [11]. Moreover, low-level failures, i.e., storage

or compute components, affect services not only regarding the respective layer, but also services that have been deployed above them [14]. Furthermore, in an NFV deployment, the virtualisation layer, which is realized through a virtual machine monitor (VMM) (also called hypervisor) [21], introduces an additional failure source. The hypervisor itself may be prone to software failures, which may affect a large part of the software infrastructure [22], [23].

In addition to the challenges associated with the technological shift, also the level of availability expectation of NFV-empowered services exacerbates the challenge of management and provisioning of highly available NFV services. A multitude of NFV envisioned use cases, which are expected to revolutionize the telecom industry, involve *carrier-grade* services that require their network being "always on" (i.e., 5-nines) [13], [24]. Also the imminent 5G cellular system, for which NFV represents an essential enabling technology [25], [26], envisions very demanding usage scenarios such as Ultra Reliable and Low Latency Communications (URLLC). Services like e-health applications, autonomous driving, or tactile Internet expect that the underlying infrastructure, e.g., the one supported by NFV, is able to provide even beyond 5-nines availability, being translated into less than 5 minutes of downtime per year. Ensuring such highly-demanding availability levels for NFV-based services is extremely difficult also because most virtualized data centers are designed to offer virtualized instances, which can usually achieve up to 99.9% uptime (three 9s) [27], [28], hence limiting the capability to provide highly available virtualized networked services.

The high expectation, in addition to the foreseen challenges of a complex infrastructure that relies on virtualization, software, and hardware resources that are not yet mature enough, makes availability a serious factor that may endanger the NFV transformation. To this end, ETSI has provided several guidelines with regard to availability and reliability requirements, models, and capabilities for end-to-end NFV-enabled services [11], [29], [30]. However, the included reliability and availability models, and their estimations, are derived from simple and basic models, which do not capture the failure and recovery process dynamics, and the inter-dependencies between the different components involved in the end-to-end service delivery such as VNFs, virtualization layers, compute, storage, and internetworking infrastructure (e.g., routers, links, switches), see for example [29]. Consequently, it becomes important to evaluate and quantify the availability of NFV-enabled services through more realistic models that are able to capture the system behavior and include all the involved service elements. Assessing dependability attributes will help identify critical elements within the NFV architecture and provide useful feedback to service providers on how to deploy, operate, and manage network services and the underlying infrastructure, for providing robust and highly dependable services. Therefore, for NFV-based services, the availability has to be considered all the way from the physical layer up to the virtualization and service layer, and resilience mechanisms need to be integrated into the software and service provisioning design.

Fault-tolerance is the basic resilience principle that helps systems achieve high availability even in the presence of faults and it is commonly accomplished by using extra resources in addition to those necessary for the system to provide its services. These extra resources are called redundancy and are used to protect a system from failures of

primary resources [10]. In [29], ETSI introduces the required mechanisms for supporting and enabling resilience management and assurance. Through model-driven analysis, an operator can estimate and assess availability figures that services can achieve under specific redundancy levels such as single, double redundancy and so forth. However, an operator also needs to plan for availability by orchestrating NFV resources such that the allocation of redundant units provides effective protection against failures, service availability demands are fulfilled, network resources are efficiently utilized, and business profit is maximized. In general, an NFV resource allocation is a challenging problem that involves a set of decisions on where to place, how much to allocate, and how to concatenate VNF instances such that system constraints are optimized [8], [31]. The allocation of redundant resources is a resource allocation problem with a particular focus on the satisfaction of service availability requirements [32]–[35].

From a resiliency perspective, the basic methods highlighted in [29] emphasize that VNF placement constraints should adhere to anti-affinity rules, which specify the placement constraints with respect to common failure modes in the hosting infrastructure. Anti-affinity rules form the basic mechanisms for enforcing placement of redundant units such that redundancy is effective against failures. However, while such rules are key placement constraints from the resiliency point of view, there are other important constraints that are required in the placement decisions, including, without limitation, node resource capacity constraints, performance related constraints, and other service optimization constraints, e.g., path routing through a predefined sequence of VNFs [8]. In addition, it is not sufficient that redundant instances are placed at separate hosting machines as correlated failures that impact both primary and redundant function may arise due to network structural dependencies [13], [36], [37]. Moreover, redundancy can be costly, especially when high availability levels are demanded, and unless planned carefully it may significantly limit the network resource efficiency. Therefore, smart resource allocation decisions are necessary for optimizing the benefits that NFV embrace.

Accordingly, the overarching theme of this work is the availability of NFV-driven network services, which is defined as the probability that the service will be provided when needed [10]. In particular, the focus of the thesis is on the methods and tools to abstract, estimate, and analyze availability of end-to-end NFV-driven services for identifying dependability flaws, effective redundant mechanisms, and critical system elements that pose threats to service resilience. Furthermore, the thesis research focus is further extended on the orchestration of redundant NFV resources such that the provisioning of highly available services can be achieved by allocating resources in an efficient and scalable way.

1.3 Research Questions and Objectives

Research Questions

Although network operators monitor service properties after deployment, traditionally, they also employ models to estimate properties such as performance or availability [38]–

[40]. Availability modeling is a widely used technique for evaluating and analyzing service availability of computing and communication systems [39], [40]. It is common that system designers use availability modeling for performance prediction since the early stages of system lifecycle [39], [41]. The nature of the system under analysis drives the choice of the modeling formalism but also vice versa — the formalism may limit the level of details that can be included in the abstracted system model [40]. The design of detailed availability models requires to identify the most significant failure modes concerning the system components, which are involved in the delivery of services. Once these modes have been identified, the interplay between the elements and their inherent dependencies needs to be factored in the overall availability model. Moreover, the choice of appropriate modeling techniques, which allow to mirror realistic dynamics of failure and repair processes, may play a significant role. Accordingly, this poses the first research question (RQ):

RQ1 - How to design analytic models that allow to characterize in detail and assess the availability of NFV-based services?

A key improvement of NFV-enabled networks is the flexibility to deploy and run virtualized network functions potentially anywhere in the network. This advantage allows an operator to instantiate VNFs and optimally distribute them in distinct parts of the network. For example, an IDS needs to be placed behind a firewall on the edge of the network. However, from the service availability perspective, this distributed deployment imposes connectivity requirements such that an end-to-end service can be deemed available. The service is available only if in addition to the VNFs also the network devices interconnecting the VNFs are available. Thus, the evaluation of the service availability should regard also these elements and the next goal is to ensure that the models, which are used to represent the behavior of the overall NFV service, will incorporate all the elements involved in the service delivery. As highlighted by ETSI [29], a correct evaluation of the availability of end-to-end services needs to take into account also the connectivity requirements, which are imposed by the network interconnecting the geo-distributed VNFs composing a service chain. Correspondingly, it comes naturally to ask:

RQ2 - How to define availability models that feature connectivity requirements among the involved elements providing and supporting end-to-end NFV services?

Analytic availability models are valuable tools to quantify and predict the availability of NFV-driven services. They can enable a modeler to construct various settings that simulate practical fault-tolerant configurations, in the form of redundant resources, which can be further assessed and compared. However, an operator needs to know not only the most suitable fault-tolerant setups but also how to provision these redundant resources in the network such that target service availability values can be achieved. Allocating redundant resources requires knowledge about the optimal placement of the resources in the network substrate and their assignment to service requests such that availability, performance, and other optimization constraints are satisfied. In

addition, from an operator's perspective, it is also required that redundancy allocation approaches present key important features such as scalability and resources-efficiency. Consequently, it becomes imperative to question the following:

RQ3 - How to construct optimal (or near-optimal) redundancy allocation schemes that are scalable, cost-efficient, and provide adequate protection against failures?

Research Objectives

Modeling complex systems can be cumbersome and tedious. Nevertheless, it is important that the model design is sufficiently able to capture the main system features and behavior, which influence measures of interest. For example, the combinatorial models presented in [29] conceptualize the service at a high level and are useful tools to estimate service availability. However, they cannot be used for evaluating system outputs related to failure and repair process dynamics of service components such as VNFs, links, or networking devices because they do not capture the interaction and dependencies among them [40], [42]. As a result, more powerful models are needed to evaluate and assess service availability. To this end, one of the objectives of this thesis is to develop availability models of end-to-end NFV-supported services by employing an abstract representation, which is able to exhibit also system (and component) dynamics in terms of failure and repair processes, capture dependencies among components, and characterize various redundancy mechanisms such that also carrier-grade availability can be achieved (*OB1*).

An end-to-end NFV-enabled service, where both ends are customers, is a composition of several functional blocks, which are connected in series or parallel, to construct a network service chain [29]. These functional blocks include not only the VNFs and the supporting infrastructure (e.g., virtualized hardware) but also networking and interworking equipment. This is particularly important since a key characteristic of NFV-enabled networks is the ability to flexibly and dynamically deploy VNFs anywhere in the network, and an operator can interconnect them for realizing specialized network services in the form of service chains [20]. However, from an availability perspective, this flexibility imposes connectivity requirements among elements since the service will be available only if all the functional elements are available. Therefore, the availability of a network service has to be estimated based on the availability of all these functional blocks. Although several research efforts have performed model-based quantitative evaluation of NFV service availability, both prior and while this thesis work was being developed (see for example [43]–[45]), none of them has considered the effects of the underlying physical network and its intrinsic topological dependencies emerging from the network connectivity requirements. To address this gap, another objective of this work is to propose a comprehensive methodology to characterize the availability of end-to-end NFV-deployed services, which integrates all the service functional elements (*OB2*).

In general, a model-based evaluation process consists of two phases: a *modeling* phase and a *solution* phase. The ultimate goal of the modeling process is to facilitate a detailed evaluation of the system availability characteristics. This is achieved by solving

the model through appropriate analytic or simulative approaches to compute measures of interest. Solving the model will enable the users to carry out assessment and analysis of the most influential factors/components, study tradeoffs for redundancy alternatives, identify availability bottlenecks, and understand the impact that parameter uncertainties have on the system output. Accomplishing this phase serves as the next objective of this thesis (*OB3*).

While model-based analysis can help a user identify, among others, adequate redundancy levels for reaching target availabilities, the provisioning of highly available NFV services requires also that the operator carry out a set of tasks in which redundant resources are optimally allocated to the virtualized functions composing the services [46]–[49]. In particular, various studies have shown that simply deploying primary instances for network services, i.e., primary VNFs, is not enough for satisfying stringent availability demands [50], [51]. The allocation of redundant resources, also called backup resources, is a variant of the NFV resource allocation problem with an emphasis on guaranteeing service availability demands. It is typically formulated as a Integer Linear Programming (ILP) mathematical optimization problem, which consists of a set of decisions that ultimately define the placement of backups within the network substrate, how the backup instances are chained together, and the traffic route steering. In addition to these decisions, there are other system constraints that influence the decision making including, without limitation, node resource capacity constraints, performance related constraints (e.g., latency), and other service optimization constraints [8], [31]. Henceforth, an operator needs to adopt schemes that optimally place and assign VNF backup instances while satisfying service availability and performance requirements. To this end, an additional goal of this thesis work is to develop optimized redundancy allocation strategies that enable highly available NFV-based network services (*OB4*).

The NFV resource allocation problem, and its availability-aware variant, is widely acknowledged as a challenging and not trivial problem. Moreover, many studies consider it as closely related to the well-known $\mathcal{NP}$-hard virtual network embedding problem (see for example [31], [52] and references therein). As a result, also the NFV redundancy allocation problem is $\mathcal{NP}$-hard [50], [53], [54]. Although an optimized scheme provides an optimal solution, given the nature of the problem, its applicability is limited to small-scale problem instances. If the problem scale increases, e.g., a higher number of service requests or a larger network topology, the computation efforts becomes unsustainable. A common workaround to this limitation is to propose ad-hoc heuristics, which are able to scale well to medium- and large-problem instances and at the same time obtain near-optimal solutions.

An important drawback of redundancy is that it can be costly in terms of additional resources [50], [55]. This can be particularly critical in case services require high availability, e.g., 5-nines or 6-nines, as more additional resources are required to satisfy such demands. This may result in a resource exhaustion situation and thus inhibit the network ability to accommodate new flows. Henceforth, unless carefully planned, redundancy may come at an increased cost and resource allocation schemes should be able to achieve a balance between multiple objectives that can also be in conflict with each other. To tackle this challenge, the final objective of this thesis is to propose a heuristic

algorithm that performs near-optimal, scalable, and resource-efficient NFV redundancy allocation (*OB5*).

To summarize, in order to answer to the research questions, the contributions presented in this thesis have been directed towards multiple goals in regard to constructing availability models, assessing the availability of end-to-end NFV-enabled services, and proposing and evaluating optimized, resource-efficient, and scalable redundancy allocation strategies for supporting high-availability levels. The objectives of this thesis can be outlined as follows:

OB1 - Design availability models that characterize failure dynamics of involved service elements and incorporate different failure mitigation mechanisms (Paper A, Paper B, and Paper C);

OB2 - Develop a comprehensive availability mode that takes into account not only NFV system elements but also network connectivity requirements imposed by NFV deployment schemes (Paper D);

OB3 - Perform quantitative model-driven assessment and analysis of the service availability aiming at identifying critical failure parameters, service elements, and redundancy techniques for ensuring highly available services (Paper A, Paper B, Paper C, and Paper D);

OB4 - Design and formulate optimized redundancy allocation schemes for enabling high-availability levels for NFV-based services (Paper E);

OB5 - Propose a novel approach for VNF redundant placement and allocation of service chains, which is both scalable and resource efficient (Paper F);

1.4 Research Methodology

This section briefly describes the research methodology adopted to achieve the research goals previously described. It follows the well-established scientific research process [56], and Figure 1.1 outlines the logical view.

The research effort started with a general research scope definition which subsequently was narrowed down through many discussions with my supervisors and colleagues having expertise in the technological area. This was followed by a literature review of the NFV architecture and the associated dependability challenges. In particular, this step was capital in understanding the background and the related literature, as well as identifying open challenges and defining the research questions. Following that, the system model and underlying working hypothesis are defined. In particular, the working hypothesis (or assumptions) eased the system model definition by providing helping simplifications yet, without loss of generality. Subsequently, the research process is divided into two separate tracks, consisting of the design of the availability models and the mathematical optimization/heuristic models, in regard to the objectives

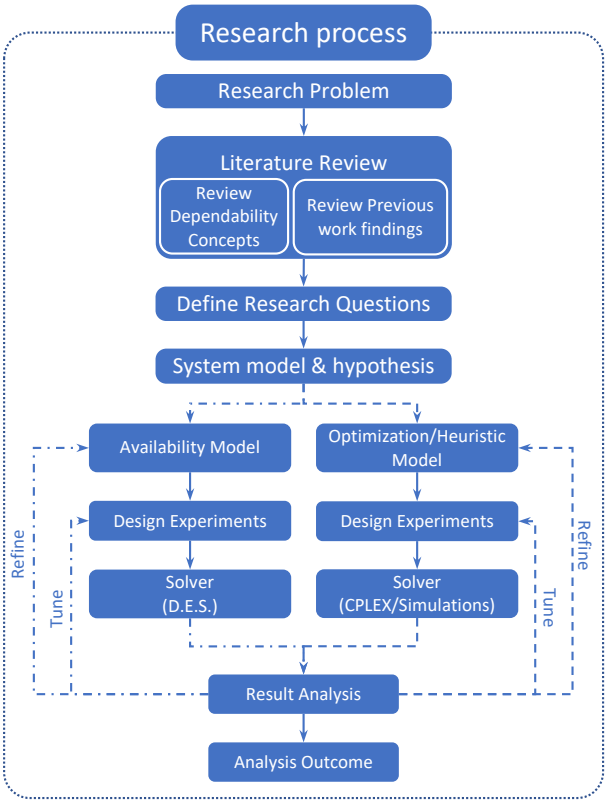


FIGURE 1.1: Research Methodology.

of model-based availability evaluation and the optimized redundancy allocation problems. Afterwards, targeted experiments, aiming at retrieving metrics of interest, are carried out and solved via either simulations or exact solvers like CPLEX. Then, result analysis is performed and in many ways this step helped both refine and tune the system models and experiments, respectively, by providing useful feedback. Finally, produced outcomes are reported in the form of contributions presented in this thesis.

Chapter 2
Background

In this Chapter, the background of the present thesis is introduced, and the related study literature is reviewed. First, the NFV architecture and the associated dependability requirements and challenges are introduced. Then, the basic principles of service dependability and the different methods for modeling dependability attributes are presented. Additionally, the NFV resource allocation problem and the availability-aware variant are illustrated. Finally, the related works in the research area are reviewed together with discussion about open challenges identified from the revision of the current state of art.

2.1 NFV Architecture

Today's modern networks are composed of diverse network functions deployed in specialized proprietary hardware, commonly called network appliances or middleboxes. These network appliances perform important network functionalities and despite they represent a vital part in today's networks, they are associated with several problems that can be identified, among others, in reduced flexibility, high operational and capital expenditure, and highly demanding innovation procedures [4].

Network Function Virtualization is an emerging solution that promises to alleviate the numerous disadvantages brought by traditional network appliances. NFV aims to radically transform the way network operators architect, operate, and manage networks by leveraging server virtualization technology for consolidating network appliances onto standard high volume servers, switches, and storage equipment, which can be deployed in datacenters, network nodes, or end user promises. NFV envisions the implementation of network functions as software running in virtualized environments, which is decoupled from the underlying hardware and can be instantiated in different locations without the need for installation of new vendor equipment.

Applying NFV brings many benefits to network operators hence, contributing to a radical change in telecommunications industry. Some of the foreseen benefits include lower capital expenditures, by eliminating the need to purchase costly specialized network appliances, reduced operating costs as through a centralization of the network management a reduction of staff time to maintain networks is foreseen, and greater flexibility and scalability since it will require much less time and work to add new capabilities in the network [57].

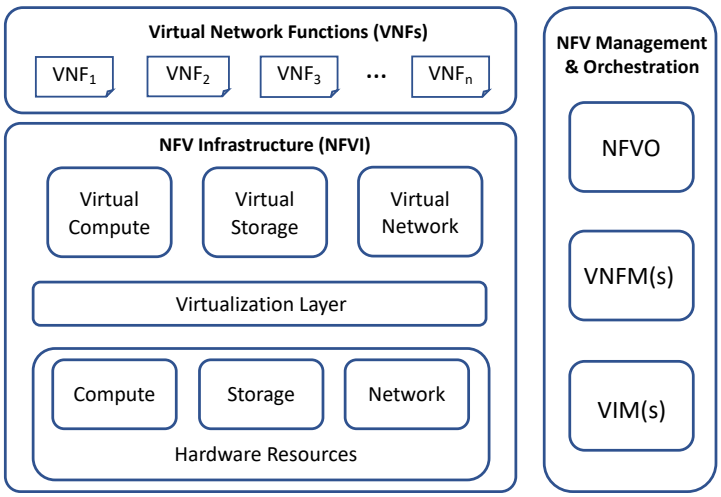


FIGURE 2.1: NFV high-level architecture (adapted from [20]).

The NFV concept, conceived in late 2012, started as an international collaboration among some of the leading Telecom organizations with the intention of accelerating the development and adoption of agile, open, and interoperable solutions for the telecommunication landscape, primarly based on high volume industry standard servers [7]. Its conception triggered an industry movement where more that 300 companies, evolved into the ETSI NFV Industry Specification Group (ISG), are leading a large-scale innovation in the telecommunication domain. From member's feedback, field-trial experiences, and proof of concepts, the ETSI NFV ISG has published more that 100 publications specifying and recommending standardized guidelines and requirements for the NFV ecosystem.

NFV envisages the implementation of network functions (NFs) as software-based entities that run over a virtualized infrastructure constituted by compute, storage, and networking resources. Figure 2.1 depicts the high-level NFV reference architecture, which consists of three main working domains [20]: the NFV Infrastructure (NFVI), the VNFs, and the NFV Management and Orchestration (MANO).

NFV Infrastructure (NFVI)

The NFVI is the set of hardware and software resources that constitute the environment where VNFs are executed. The physical resources include high volume industry standard equipment providing computing, storage, and network hardware resources.

Virtual resources are abstracted counterpart of computing, storage, and network resources. This abstraction is achieved using a virtualization layer, which decouples the virtual resources from the underlying physical resources. Typical virtualization technologies, where VNF can be executed, can be based on a hypervisor or containerized

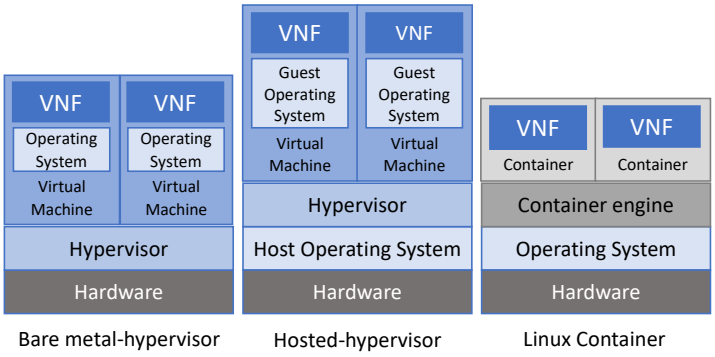


FIGURE 2.2: Most common virtualization technologies.

infrastructure. Figure 2.2 depicts these two most common virtualization technologies, i.e., virtual machines (through either bare metal- or hosted-hypervisor virtualization) and containers [58]. A hypervisor is a software allowing the emulation of hardware resources. The emulated resources, referred to as virtual resources, abstract physical resources and enable running different operating systems on top of common shared hardware resources. This way, the hypervisor enables the operation of multiple machines (virtual) within the same hosting computer. Each machine is associated with virtual resources, i.e., virtual CPU, memory, disc, and represents a "closed" environment where applications, e.g., VNFs, can be deployed and operated. It is possible to differentiate two types of hypervisors, *Type 1* hypervisor running directly on hardware (hence also called bare metal) not requiring an operating system and *Type 2* hypervisor running on the operating system of the host machine. Differently, container-based virtualization utilized the kernel features to create isolated environments, a.k.a. containers, for processes. Container virtualization does not emulate an entire computer rather create environments where software can directly communicate with the host kernel for utilizing hardware resources [58].

Virtual Network Functions (VNFs)

A virtual network function is the software implementation of a network function, e.g., firewall or deep packet inspection, which can be deployed in virtual resources such as virtual machines (VMs) or containers. A VNF can be decomposed into smaller functional modules for scalability, reusability, and/or faster response, or multiple VNFs can be composed together to reduce management and VNF traffic steering complexity. Decomposing a VNF is the process whereby a higher-level VNF is split into a set of lower-level VNFs. A single VNF may be deployed into a single VM or it may be composed of multiple components and thus it can be deployed over multiple VMs [20].

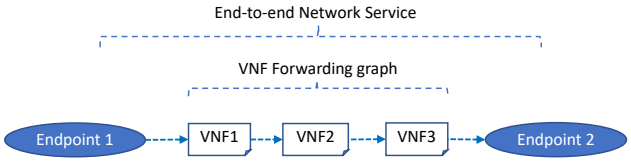


FIGURE 2.3: Illustration of an end-to-end network service.

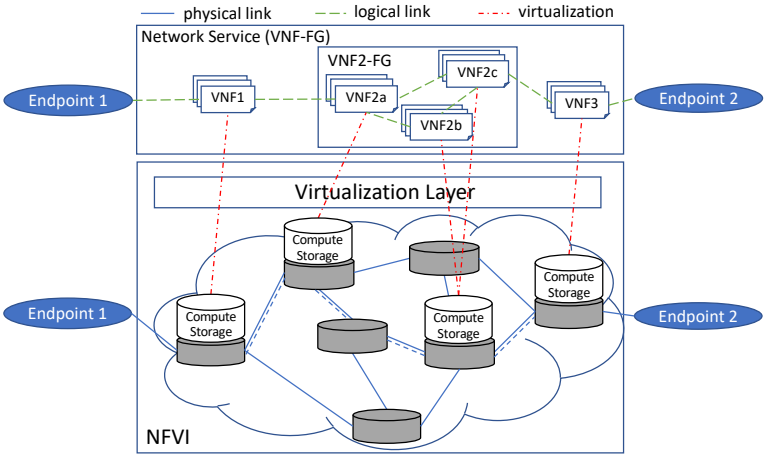


FIGURE 2.4: Deployment of an end-to-end network service with VNF forwarding graph.

NFV Management and Orchestration (NFV MANO)

The NFV Management and Orchestration (NFV MANO) is responsible for the orchestration and life-cycle management of the physical and software resources supporting the virtualized infrastructure, and the life-cycle management of VNFs providing the network service.

The NFV MANO entity, in the remainder referred to as simply the MANO, comprises three separate functional blocks, namely NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). The NFVO is the orchestrator of the architecture and is responsible for operations such as on-boarding, instantiation, or termination of network services and the orchestration of their corresponding resources. The VNFM is responsible for VNF lifecycle management including typical operations like VNF instantiation, update, query, scaling, or termination. Multiple VNF Managers may be deployed; a VNF Manager may be deployed for each VNF, or a VNF Manager may serve multiple VNFs. The VIM comprises the functionalities that are used to control and manage the interaction of a VNF with computing, storage, and network resources under its authority, as well as their virtualisation.

NFV Network Services

The constituent blocks of the NFV architecture interact with each other to provide end-to-end network services. An end-to-end network service can be described by a Forwarding Graph of interconnected NFs and end points [59]. Figure 2.3 shows a graphical representation of a VNF Forwarding Graph (VNF-FG), which defines the composition of VNFs providing an NFV-enabled service, and their relative sequence for traffic to traverse. This is similar to the definition that the Internet Engineering Task Force specifies as an SFC - "the definition and instantiation of an ordered set of service functions and subsequent steering of traffic through them" [6].

In the NFV context, both nomenclatures refer to the same thing, hence hereafter we will refer to an SFC as the composition of an ordered set of VNFs providing a network service. Thus, the deployment and delivery of an end-to-end service, illustrated in Figure 2.4, where both end points are customers of the NFV architecture, comprises several network functions, which are mutually connected in parallel or in series, to construct a network service graph in the form of an SFC. The service is implemented and operated through an interaction of the SFC, realizing the service, and the MANO, which acts as the manager of the service life-cycle.

2.2 Dependability Concepts

This section introduces a brief revisit of basic definitions, threats, and means for achieving dependable systems. The revision content is mainly based on notions and definitions taken from [10], [15]. In addition, the most widely used dependability modeling techniques and their relative capability, such that the concepts and relations from this work can be easily identified, are introduced.

System dependability is the defined as "*the ability to deliver a service that can justifiably be trusted*". This definition highlights the requirement of justifying the trust to be placed upon a system. An alternative definition, which imposes the criteria of whether a system is dependable, is "*the ability to avoid service failures that are more frequent and more severe than is acceptable*". Generally, it is referred to dependability as an umbrella term that integrates concepts including: *threats* to, *attributes* of, and *means* by which dependability is accomplished. Figure 2.5 shows the relation between these concepts in the diagram known as the dependability tree. In the following sections, they are introduced in more detail.

Dependability Attributes

Five principal attributes can be used for characterizing the dependability of a system: availability, reliability, safety, integrity, and maintainability [10]. Later, security, as a composite of integrity, availability, and confidentiality, was integrated with the other dependability attributes for establishing a dependable and secure computing taxonomy. One of the most well-known attribute is system availability, which refers to the ability of a system to deliver services at a given instant of time or within a specific time interval.

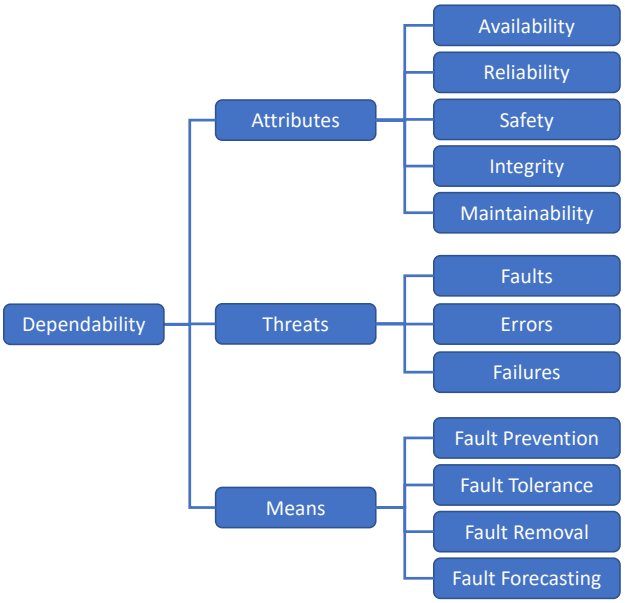


FIGURE 2.5: Dependability tree (source [10]).

Another important concept is reliability and it refers to the ability of a system to provide uninterrupted service. The third concept, safety, expresses the ability of a system to provide service without experiencing catastrophic failures. Integrity refers to the omission of improper system alterations. Finally, the maintainability is the ability of a system to undergo modifications and repairs.

The extent to which a system retains dependability attributes is to be considered in a probabilistic sense and not in an absolute, deterministic sense. Due to the unavoidable threats, which will be introduced in the following section, a system is never to be regarded as absolutely available, reliable, and so forth.

The importance of any of the attributes over another may be subject to the application service that is under consideration. Availability is typically the most common attribute for assessing dependability of communication networks [60]–[62]. This is because end-users are mostly interested in service readiness, i.e., being able to use the service whenever they want [63]. Moreover, availability is the most common attribute specified in SLAs for services provided by communication networks [17], [18]. This is also true for web-based services provided through virtualized infrastructures such as cloud computing [27], [28], where an important service level objective is the definition of the monthly uptime percentage, i.e., monthly service availability. These remarks serve as motivation for this work to focus on the service availability as a primary and noteworthy dependability attribute.

For quantifying system availability, some measures have to be introduced. Uptime

refers to the time period during which the service is correctly delivered. It is commonly quantified in terms of the Mean Up Time (MUT), which defines the mean interval of time from the moment the service is restored after a failure until the next service failure. Similarly, the Mean Down Time (MDT) measures the mean time duration from the instant a failure is experience until the service is restored. Usually, it is important to guarantee service availability in the long time range, hence one is more interested in the steady-state availability A, i.e., asymptotic availability. Such a metric quantifies the probability that the service can be correctly accessed at some point in the future and is defined as [62]:

$$A = \frac{MUT}{MUT + MDT}$$

Inversely, the probability that the service is not correctly delivered at some point in the future defines the asymptotic unavailability U, thus yielding $U = 1 - A$. In addition, if one is interested on the availability on a specific time interval τ, the interval availability ($A(\tau)$) is defined as the fraction of time in which the service is correctly delivered within the given interval. Note that some literature uses different but completely equivalent terms where MUT is referred as Mean Time To Failure (MTTF) and MDT refers to Mean Time To Repair (MTTR) [40]. Accordingly, the steady-state availability can be expressed as:

$$A = \frac{MTTF}{MTTF + MTTR}$$

where MTTF defines the average duration of time from the moment a service request is received, given that the service was up at that time, until the first service failure is experienced and MTTR defines the average time it takes for the service to be repaired.

Threats to Dependability

There are three *impairments* to dependability: *faults*, *errors*, and *failures* [10], [63]. According to the definitions, there is an intrinsic relationship between these dependability threats, shown in Figure 2.6, known as "chain of threats".

Faults are the "adjudged or hypothesized cause of an *error*". They can be physical defects, electromagnetic shocks, flaws in software, etc. A fault is active when it causes an error otherwise it is dormant. A dormant fault may be triggered within the system, leading to an active fault, which may in turn be observable as an error.

An error is "the part of the total state of the system that may lead to its subsequent service failure". It is important to note that not all errors will eventually lead to a service failure. Therefore, an error can be regarded as the deviation from system correctness, which by reaching the service interface leads to a service failure.

A failure is defined as "the event that occurs when the delivered service deviates from correct service" and the period of incorrect service delivery is defined as the service outage.

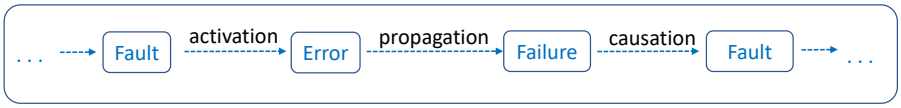


FIGURE 2.6: Fundamental chain of dependability threats (source [10]).

The arrows in the chain relate the causality between faults, errors, and failures, indicating that faults cause errors, which in turn cause failures. Such relationship should be interpreted generically and may be recursive if a system is part of another system: a failure in one system may cause a fault in another system, which in turn may cause an error and subsequently a failure. In the scope of this thesis, the contributions are mainly focused on failures and service outages rather than the sources of the lack of dependability, i.e., faults and errors. This is because the focus of this work is on the transition from a correct to incorrect service delivery rather than on the modeling of the fault, error, failure chain. Moreover, please note that in the following, the word failure will be mostly used for identifying both system failure and the specific causality type leading to the service outage.

Dependability Means

Faults are the source of dependability threats, hence the means to attain dependability focus on preventing, tolerating, removing, and forecasting faults [10].

Fault prevention is a part of a general development strategy aiming at avoiding the introduction of faults during the design and development phase. Prevention of development faults can be done both on the software level, e.g. using strong-type programming languages or modularization, and on the hardware level, e.g. by shielding the system from external threats.

Fault tolerance aims at delivering the specified service despite the existence and activation of fault within the system. The objective of fault tolerance is, as definition indicates, to tolerate faults but avoid service failures. With fault tolerance, an error is allowed to occur, but is prevented from causing a failure. The basic principle of achieving fault tolerance is the use of extra resources, in addition to those necessary for the system to deliver a service. Employing extra resources to attain fault tolerance is widely know as redundancy. Redundancy can be realized in many forms including: i) *hardware*, e.g., employing spare or parallel components, ii) *software*, e.g., enhancing the software with fault handling capabilities, iii) *information*, e.g., implementing error-correcting codes (FEC - Forward Error Control codes), or iv) *time*, e.g., enabling retransmission of erroneous/corrupted data packets.

With fault removal, the objective is to clear away faults both during the development phase and during the operational life of a system. Removing faults during the development phase of a system life-cycle consists of three steps: verification, diagnosis, and correction. Verification is the process of validating whether the system complies to given specifications. In case the verification results negative, a diagnoses of the fault(s) that

prevented the verification conditions from being fulfilled is carried out, and then the necessary corrections are implemented. After correction, the verification process should be repeated for validation that fault removal had no undesired consequences. Fault removal during the operational life is generally considered as corrective or preventive maintenance. Corrective maintenance is aimed at removing faults that have produced one or more errors and have been reported, while preventive maintenance is aimed to uncover and remove faults before they might lead to errors during normal operation.

Fault forecasting consists in performing an evaluation of the system behavior with respect to fault occurrence or activation. The evaluation can be a qualitative assessment by ranking the component failure modes that might lead to a system failure or a quantitative, i.e., probabilistic, evaluation aiming at analyzing the extent to which some of dependability attributes (measures) are satisfied. Fault forecasting is essentially modeling the behavior of system components and their interactions and processing the model(s) to obtain values of dependability measures.

2.3 Dependability Modeling

Dependability modeling is a common way engineers have used to quantify and evaluate system dependability [38], [39]. Methods to evaluate the dependability of a certain system are fundamental during all stages of the system lifecycle. Availability and reliability are key quantitative dependability measures of technical systems and the assessment and evaluation methods of these measures can be divided into two main categories: data-driven or model-driven methods [40]. The former are suitable methods for quantifying and evaluating system components or subsystems but, for large systems, the latter are more preferable [39], [40].

Solving model-driven methodologies can be through either discrete-event simulation or analytic-numeric techniques. However, the choice of the applied solution may depend on the application, and in general, it is advised that a reasonable combination of both techniques should be employed for solving large and complex system models [39], [40].

There are three main model-driven methodologies used to asses dependability measures: non state-space models (sometimes called combinatorial or static models), state-space models (otherwise called dynamic models), and multi-level models (often referred to as hierarchical models). A brief illustration of these three types, which is primarily based on [39] and [40], is given in the following.

Non state-space models include Reliability Block Diagrams (RBD), Fault-trees (FT), and Reliability Graphs (RG). These models allow a relatively quick quantification of measures because they have a simple and intuitive graphical representation [41]. RBDs and FTs are typically used to represent the logical structure of a system, with respect to how availability or reliability of system components impacts the overall system availability.

RBDs are graphical structures with a start and end dummy nodes, which are interconnected through blocks representing the system components. At any instant of time,

the existence of a path between the two dummy nodes reflects an operational system, otherwise the system is considered failed.

Fault-trees are acyclic graphs with nodes that are logic gates (e.g., AND, OR) and external nodes (leaves) that represent system components. The edges transmit the failure information from the leaf nodes (components) and if a component has failed, a TRUE logic value is transmitted; otherwise, a FALSE is transmitted. At any instant of time, the logic value at the root node determines the status of the system, i.e., operational or failed. RBDs and FTs have been extensively used in reliability and availability modeling, see for example [64]–[66].

RGs have been commonly used to quantify and estimate network dependability measures [67]. Their graphical representation is an acyclic graph consisting of nodes and edges, where edges represent system components that can fail or the relationship between components. The existence of at least one path from a source node to a destination node represents a reliable/available network (system). On the one hand, the main difference between RBDs/FTs and reliability graphs is that failure parameters (e.g., probabilities, rates, or functions) are assigned to edges, and not to nodes. On the other hand, under certain conditions, the three models can be used interchangeably since it is possible to derive one model from another. For example, FTs without repeated events (or shared nodes) are equivalent to RBDs [42]. Both RBDs and FTs are subsets of reliability graphs, which are in turn subset of fault trees with repeated events.

Despite their advantages, non state-space models rely on strong assumptions, which may induce in pitfalls when quantifying measures of interest. All the three models heavily rely on statistical independence among system components. For example, if the system is composed of different subsystems, each of the subsystems is repaired independently. This assumption is translated into having dedicated repairman for each individual subsystem, which in realistic scenarios is not common. In addition, such models assume that the system and recovery actions behave as intended, i.e. perfect repair, and they lack the required flexibility for characterizing the dynamic nature of systems where sequences of events influence the occurrence of other events. This is why they are sometimes called static models.

State-space models are suitable models for representing complex interactions and behaviors within a system. When specific assumptions are made, state-space models are also suitable methods for modeling large and complex systems, especially when regarding specific behaviors with repetition throughout the large system. There exist a variety of state-space models used in the literature. They can be simple Markov-based models like discrete/continuous-time Markov chains (D/CTMC) or semi-Markov Processes. When a reward function is associated with the chain, for the evaluation of a certain metric, they are known as Markov reward models (MRM). Other representatives of state-space models, which are more human intuitive, include Petri-net (PN)-based models like stochastic-Petri nets (SPN) and generalized-SPN (GSPN). When a reward rate is associated with the net, it is called a stochastic reward net (SRN). An additional variant of stochastic PNs are stochastic activity networks (SANs), which are based on activity networks and this kind of the extension is similar to how stochastic Petri nets are constructed from (classical) Petri nets.

For large and complex systems, the number of system states can grow extremely large. This is called the *largeness* problem for Markov-based models. One widely used way to limit the state space growth is by using hierarchical models, which help the designer avoid the largeness problem [68], [69]. Hierarchical models are multi-level models where higher levels are frequently non-state space models and lower levels are typically state-space models, which are more suitable for capturing individual complex behavior. The upper levels of a hierarchical model can be for instance, RG for network modeling, FT or RBD for structured modeling of individual systems, whereas in the lower level, state-space models such as CTMC or SAN can be used to capture complex dynamic behaviors of subsystems or individual components.

From a modeling capability perspective, Malhotra *et al.* [42] classifies the RBDs and FTs without repeated events as the least powerful models among the non-state-space models in terms of modeling conciseness and system representation. RGs are more powerful than RDBs and FTs without repeated events but less powerful than FTs with repeated events. State-space models like SPNs, CTMC, and SRN offer similar modeling power, which is higher than non-state-space type of models in that they provide the modeler with the ability to capture component dependencies, common mode failures, and shared repair facilities.

2.3.1 Stochastic Activity Networks

Stochastic activity networks are a probabilistic extension of activity networks, similar to how stochastic Petri nets are developed based on classical (un-timed) Petri nets [70]. SANs offer several advantages due to the small size of their descriptions, and their visual interpretation clarity allows the designer to focus more on the system being modeled rather than on error-prone and tedious manual constructions of lower-level Markov-based models.

In this thesis work, SAN modeling formalism, i.e., a formal language for expressing models, is used on Paper A, B, C, and D for constructing availability models of NFV-based service components, and their inter-dependencies, by abstracting and characterizing their failure and repair dynamics. The availability models are designed and solved using a well-referenced software tool called Möbius [71], which is introduced in more detail in the following sub-section.

Before introducing the formal definition of stochastic activity network let us first define activity networks. Such definitions are taken from [70], [72], [73].

An activity network (AN) is a mathematical modeling language in the form of a bipartite directed graph. Formally, an AN is a generalized version of Petri nets consisting of the following modeling elements [72]:

- *Places*, which are similar to Petri nets, contain a certain number of tokens that in turn represent the *marking* of the place. The set of all place markings in the model represent the state of the modeled system.

- *Activities*, which can be either instantaneous or timed, define actions that take a certain amount of time to complete. Each type of activity is associated with a

non-zero integral number of cases, which defines a possible action taken upon the completion of the activity.

- *Input gates*, each of which accepts a finite set of input arcs and one single output. Each input is associated to an *n*-ary enabling predicate and *n*-ary computable input function over the set of natural numbers. The input function is defined for all values for which the enabling predicate is true.

- *Output gates*, each of which has a finite set of outputs and one input. Each output is associated with an *n*-ary computable function on the set of natural numbers, called output function.

Timed activities represent the actions of the system being modeled whose durations influence the systems's ability to perform. Instantaneous activities resemble system activities that, relative to the performance metric being measured, are completed in a negligible amount of time. Activity cases model the uncertainty about the enabled activity to complete. Both timed and instantaneous activities can have case probabilities, which can be marking dependent and their sum should equal one. The definition of an activity network is as follows:

Definition 1 – *An activity network (AN) is defined as an eight-tuple [72]*

$$AN = (P, A, I, O, \gamma, \tau, \iota, o) \tag{2.1}$$

where P is a finite set of places, A is a finite set of activities, I is a finite set of input gates, and O is a finite set of output gates. Furthermore, $\gamma : A \rightarrow \mathbb{N}^+$ specifies the number of cases for each activity, and $\tau : A \rightarrow \{$Timed, Instantaneous$\}$ defines the activity type. The network structure is defined via the functions $\iota : I \rightarrow A$, which maps the input gates to activities and $o : O \rightarrow \{(a, c)|a \in A \wedge c \in \{1, 2, ..., \gamma(a)\}\}$, which maps the output gates to cases of activities.

The behavior of an activity network is a characterization of possible completions of activities, selection of cases, and changes in markings. If S is a set of places such that $S \subseteq P$, a marking of S is a mapping $\mu : S \rightarrow \mathbb{N}$. An activity $a \in A$ may complete in a marking μ if i) a is enabled in μ and ii) if a is timed, there is no instantaneous activities enabled in μ. A marking μ is considered stable if no instantaneous activities are enabled in μ. The set of reachable markings of network AN in the initial marking μ_0 is the set of markings $R(AN, \mu_0)$ where $R(AN, \mu_0) = \{\mu|\mu_0 \stackrel{*}{\rightarrow} \mu\}$. As a result, the set of stable reachable markings of the activity network AN in an initial marking μ_0 is the set $SR(AN, \mu_0) \subseteq R(AN, \mu_0)$ of reachable markings of AN from μ_0 that are stable. Moreover, an activity network AN in a marking μ_0 is stabilizing if, for every $\mu \in SR(AN, \mu_0)$, the set $S(\mu)$ is finite.

Based on these premises, we now present the definition of the stochastic extension of activity networks, as given in [73].

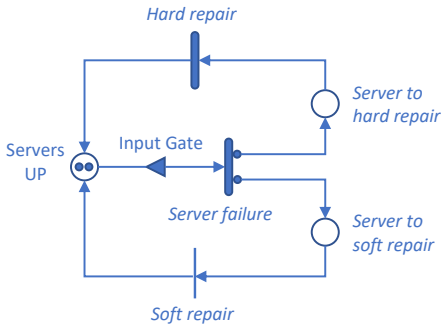


FIGURE 2.7: An example of a stochastic activity network.

Given an activity network, which is stabilizing in a given initial marking μ_0, a stochastic activity network is formed by adjoining functions C, F, and G to an activity network. Function C specifies the probability distribution of case selections, F the probability distribution of activity delay times and G defines the set of reactivation markings for each possible marking.

Definition 2 – *A stochastic activity network (SAN) is defined as an five-tuple [73]*

$$SAN = (AN, \mu_0, C, F, G) \tag{2.2}$$

where AN is an activity network, μ_0 is the initial marking and is a stable marking in which AN is stabilizing. C is the case distribution assignment of functions to activities such that for any activity a, $C_a : M_{IP(a) \cup OP(a)} \times \{1, ..., \gamma_a\} \rightarrow [0, 1]$. F is the activity distribution function assignment. It assigns a continuous function to timed activities such that for any timed activity a, $F_a : M_P \times \mathbb{R} \rightarrow [0, 1]$. Moreover, for any stable marking $\mu \in M_P$ and timed activity a that is enabled in μ, $F_a(\mu, \cdot)$ is a continuous probability distribution function called the activity time distribution function of a in μ. G represents the reactivation function assignment, which assigns functions to timed timed activities such that for any timed activity a, $G_a : M_P \rightarrow \wp(M_P)$ where $\wp(M_p)$ denotes the power set of M_P. Finally, for nay stable marking $\mu \in M_P$ and timed activity a that is enabled in μ, $G_a(\mu, \cdot)$ is a set of markings called the reactivation markings of a in μ.

A graphical representation of the network is typically the preferred way of resembling the system not only because using the tuple formulation is quite cumbersome and error-prone also for small networks, but it also provides a compact visualization and greater insight into the network behavior [73]. Figure 2.7 illustrates a simple SAN model of a server system. Places are graphically represented as circles, which may contain a certain number of tokens that in turn represent the *marking* of the place. The set of all place markings in the model represent the state of the modeled system. Activities are actions that take a certain amount of time to complete. They impact the system performance and can be *timed*, represented as thick vertical lines, or *instantaneous*, which are

depicted as thin vertical lines. A timed activity may be characterized by a deterministic duration or can have a distribution function associated with its duration. In addition, activities can have distribution case probabilities, which are graphically represented as small circles on the right of the activities. Upon completion, an activity fires and enables token movements from places connected by outgoing arcs to places connected by incoming arcs. This way a system state update occurs and tokens are moved from one place to another by redefining the places markings. Input and output gates define marking changes that occur when an activity completes. Different from output gates, the input gates are also able to control the enabling of activity completion, i.e., firing.

As mere illustration, in Figure 2.7, the place *Servers UP* contains two tokens, which abstract the presence of two servers being operational. This state represents the initial marking of the system. The input gate models the enabling of the timed activity *Server failure* and defines the token movement; move a token from the place *Servers UP* to the place *Server to hard repair* with probability p or move the same token to place *Server to soft repair* with probability 1-p. For each of the cases there is a specific recovery procedure, modeled by either an instantaneous activity, which may for example resemble a fast software restart, or a timed activity modeling a software fix through a specific distribution. When completed, the activities fire a token present in their respective places and place it in *Servers UP* indicating that the failed server has been recovered.

2.3.2 Möbius tool

Since their conception, SANs have served as the basis for different modeling tools including METASAN [74], UltraSAN [75], and Möbius [76]. In particular, the Möbius software tool has grown into a major research project, which has been supported, among others, by Motorola and DARPA agency grants. The project is currently developed and maintained by the Performability Engineering Research Group (PERFORM) at the University of Illinois at Urbana-Champaign, USA.

Möbius software offers a powerful and flexible modeling framework to specify models by using a variety of modeling formalisms. Although the original version was designed for SAN-based modeling, it now includes, among others, formalisms like fault-trees, Performance Evaluation Process Algebra (PEPA) [77], and Adversary View Security Evaluation (ADVISE) [78]. This evolution was primarily built on the belief that no formalism fits all in terms of being the best for building and solving models across many system and application domains.

Model Construction

Constructing a model is a process consisting of several steps where each step involves a specific framework component. Figure 2.8 highlights the framework components and illustrates the bottom-up procedure of model construction within the framework.

The first step is to use some formalism for generating a model, also called atomic model, made up of state variables (e.g., places in SANs), actions (transitions in SANs) and properties. The atomic models can be constructed with ADVISE [78], Bucket and

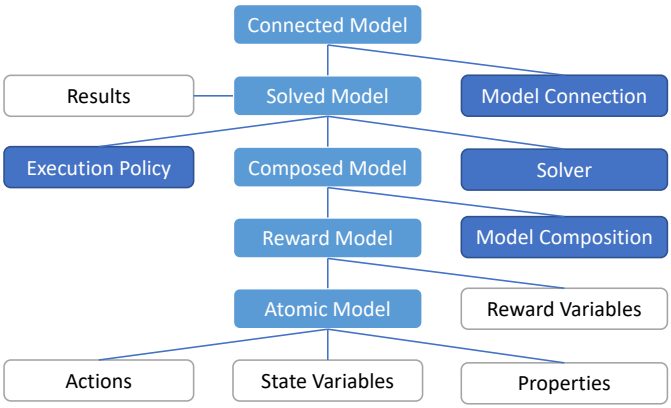


FIGURE 2.8: Möbius architecture components (source [79]).

Balls [80], Fault trees, PEPA [77] or SAN formalim. Once an atomic model is constructed, the next step is to define some measures of interest using reward specifications in regard to performance, dependability and/or performability [81]. This step defines a reward model that enhances the atomic model with reward variables. Subsequently, in case the atomic model is expected to be part of a larger model, it can be composed with other models. This can be achieved by leveraging composition formalisms such as Replicate/Join and graph composition in which system symmetries and state lumping techniques can be exploited. The next step is to execute a solver to compute a solution of the model. This step generates a solved model and the mechanism that calculates the solution to reward variables can be exact, approximate, or statistical. The computed solution is called a result, and since the reward variables are random variables, the result is expressed as some characteristics of a random variable, e.g., mean, variance, or distribution of the reward variable. The result may also include solution information such as error levels, confidence intervals, or stopping criterion. If the result is intended as an intermediate step toward the final desired measure, it may capture the interaction between multiple reward models that form together a connected model. This is a useful technique of modeling using decomposition approaches, such as those used in [82], where the model of interest is a set of reward models with dependencies expressed through intermediate results.

Model Composition

A special feature of Möbius is the ability to construct composed models from previously defined ones. Such models can be atomic models or other composed models. This way, a modeler may exploit a hierarchical approach, by constructing submodels as specific units and then linking them together in a well-defined manner to construct a model of a system. The composition method makes use of two approaches; i) the action-sharing approach in which submodels are composed through superposition of a subset of their

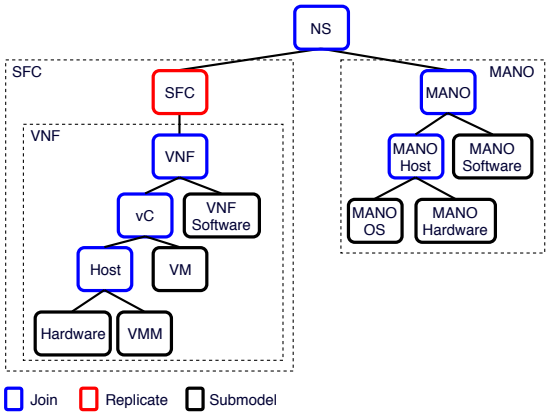


FIGURE 2.9: A Replicate and Join composition model of a virtualized network service (Paper A).

actions and ii) the state-sharing approach, which links submodels together by identifying sets of state variables. For example, it is possible to compose two SAN models by causing them to hold a particular place in common. This allows for interaction between the submodels since both can read from and write to the identified state variable. This form of state sharing is known as equivalence sharing since both submodels have the same relationship to the shared state variable. The two supported composition formalisms that use equvalence sharing are Replicate/Join and graph composition. To illustrate, Figure 2.9 depicts the Replicate/Join composed formalism utilized in Paper A for constructing a hierarchical model of an NFV-enabled network service. This way, system symmetries can be evidenced and exploited by reducing and compacting the model abstraction. For example, the left-hand side model is build on multiple levels, each of which joins separate elements, i.e., submodels, up to the higher level that represents the whole VNF element. The Replicate node, i.e., SFC, replicates the VNF model to the number of VNFs that compose the considered service chain.

Model Solution

Once the system is modeled, it needs to be solved for computing the various measures of interest, e.g., system availability. The Möbius tool supports two classes of solution techniques: discrete event simulation, and state-based analytic/numerical techniques. While the choice of solver may be subject to several factors, the simulation solver can be used for all models present in the tool, whereas only models that have exponentially distributed and deterministic delays may be solved through analytic solvers. In addition, simulation is a more suitable solution technique also for models that generate arbitrarily large state space compared to numerical solvers, which are limited to models that have finite, small state-space. In case models generate large state space, the solution runtime of analytic solvers can easily become unsustainable and this represents a serious

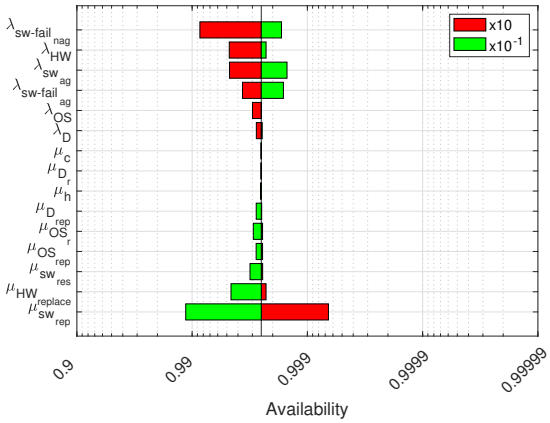


FIGURE 2.10: Sensitivity analysis of the MANO *manager* deployment without software rejuvenation (Paper C).

limitation compared to simulation solvers [83]. Such premises served as motivation for adopting simulation-based solvers for the availability models proposed in papers A, B, C, and D.

Sensitivity Analysis

During the model construction phase in Möbius, global variables can be used to parametrize model characteristics. For example, failure and repair intensities of components can be defined as global variables which are not assigned a specific value until the models are ready to be solved. Assigning values to global variables forms an experiment and these variables can be used to construct specific experiments which can be grouped together and form a study of some measures of interest, e.g., availability or performance. The most sophisticated study in the Möbius tool is based on a design of experiments (DOE) which generates a set of experiments and then analyzes the reward variable solution to determine how the chosen global variable(s) impact the reward variables. This is referred to as sensitivity analysis in which factors that mostly influence the model output measures are determined. This is an important step in model development as it does not only provide the means to carry out model validation but also helps determine: i) which parameters deserve additional research effort for reinforcing the knowledge base; ii) which input parameters are insignificant and can be neglected; iii) which inputs mostly contribute to output variability; iv) what consequences result from input variability. As a result, model-based dependability studies frequently apply sensitivity analysis to assess the effect of changes on system measures of interest.

In general, there are two kinds of sensitivity analysis: non-parametric and parametric sensitivity analysis. The former studies the system output variations as a result of system structure modifications, like removal or addition of a given model component [84]. The latter, and most relevant for the scope of this thesis work, performs the study of output

variations due to a change in the input parameters [85], [86]. In particular, parametric methods that are widely used in dependability studies can be classified into the group of methods that operate on one variable at a time [86]–[89]. One fundamental technique is the differential sensitivity, also called direct sensitivity in which partial derivatives with respect to each input parameter are calculated; however, although this technique is computationally efficient, the effort required in solving partial derivatives of complex models can be quite intensive [90]. Moreover, this method is only valid for small perturbations of parameter values and the partial derivatives need to be calculated for each change of the baseline parameters.

An alternative method is to repeatedly vary one parameter at a time while holding the other input parameters fixed and observe the output variability. This method, also called discrete sensitivity analysis, is one of the most common approaches because of its simplified application and also the majority of related work studies adopt this technique for analyzing dependability of NFV-based services (refer to the related work presented in Section 3.1). Similarly, this thesis contributions, included in Paper A, B, C, and D, apply discrete sensitivity analysis aiming at assessing system availability, identifying critical factors, and evaluating fault-tolerant mechanisms. Specifically, model input parameters are evaluated at their mean value and then varied within specific bounds that represent parameter uncertainties. Subsequently, models are solved and results are analyzed by observing the impact on service availability caused by perturbation of input parameters. As mere illustration, Figure 2.10 depicts a sensitivity analysis regarding the impact that variations of one order of magnitude of failure and repair parameters have on the NFV-MANO availability. Such analysis helps identify critical parameters that mostly impact system availability, either positively or negatively, and the extent of their impact.

2.4 NFV Resource Allocation

The NFV architecture, seen in Figure 2.1, is composed of an interaction of various components for providing, managing, and orchestrating network services. An excellent technique for composing specialized end-to-end network services is by means of service function chaining. A service function chain defines an ordered set of VNF types through which traffic is steered through, e.g., a firewall chained with a deep packet inspector, an encryption element, and a data monitoring tool. This becomes particularly advantages in NFV because the flexibility introduced by virtualization allows operators to dynamically deploy and orchestrate service chains, unlike traditional chain composition that is constrained by the fixed position of the middleboxes. For achieving the NFV expected benefits, several challenges have to be addressed and one important challenge regards the resource allocation to service function chains for satisfying strict performance and/or dependability requirements [8], [31], [34].

The allocation of NFV resources to service function chains involves a set of decision making at both the network-level (VNF placement on the NFVI) and the flow-level (flow routing) [8], [46], [91]. In particular, the authors of [8], [46] emphasize that in order to achieve the economies of scale expected from NFV, there is a need for efficient resource

allocation algorithms, which are capable of determining the optimal *places* where VNFs are to be deployed, i.e., which of the computing nodes will host which VNF type, the *chaining* of the VNFs for providing effective SFC, depending on the type of VNFs that services request, and an efficient *allocation* of physical resources to every VNF, based on their demanded capacity.

On the network infrastructure level, the decisions include: i) deciding the optimal number of VNFs to deploy in the network infrastructure and ii) the optimal location, i.e., computing node, at which each VNF shall be placed. The set of flow-level decisions regards iii) the optimal selection (or assignment) of the VNFs that will form the service chain as requested by traffic flows and iv) finding the optimal routing paths among the VNFs composing the chain. In addition, NFV allocation decisions have to be determined considering also the limited amount of resources that the network infrastructures have. The amount of bandwidth, computing capacity, storage, and memory impose constraints that should not be violated and at the same time the resources should be used efficiently. The allocation decisions have to be made taking into consideration such constraints while fulfilling service performance and dependability requirements.

Although seemingly separated, the placement and flow-routing problems are tightly coupled and inter-dependent. This is because traffic flows shall be assigned to VNF instances, which in turn are instantiated during the placement task. Therefore, a complete solution to the resource allocation problem should stem from addressing the joint problem of accomplishing all four decisions. The joint problem can be formulated as a mathematical optimization problem with one or more specific objectives that may represent some operator's goals such as: i) maximizing remaining network resources, ii) minimizing network power consumption, iii) minimizing service latency, and/or iv) minimizing CAPEX and OPEX costs. In addition, the optimization problem will be subject to some design constraints such as computing node capacity, end-to-end service delay, VNF assignment for composing the correct SFC request, and so forth.

Regarding the nature of the optimization problem, many studies ([31], [46], [52], [92]–[94]) regard the problem as closely related to the well-known virtual network embedding (VNE) problem [95]. In particular, [31] regards the VNF placement problem, coined VNF forward graph embedding (VNF-FGE), as a generalization of the VNE problem. Differently, [96] regard the joint VNF placement and routing problem as closer to a facility location problem [97] rather than a generalization of a classical VNE. The authors argue that a holistic covering of all VNF operations can not be adequately represented by classical VNE. Nevertheless, both lines of argument agree that the joint placement and routing of VNFs differs from the classical VNE in several aspects including: i) VNF ordering constraints are aspects that cannot be easily represented in a VNE, and ii) VNF resource demands are dynamic, whereas in VNE they are mostly static. Moreover, there is a common agreement that the NFV resource allocation problem formulation has to be directly tailored to the NFV environment, with all it's peculiarities, and that the problem itself is an $\mathcal{NP}$-hard problem [31], [96]. Therefore, the optimization problem may become intractable and solution runtimes can quickly increase for large instances of the problem, hence creating scalability issues.

The two tasks can be addressed either in a coordinated or uncoordinated fashion [31].

Coordinated approaches try to jointly address both tasks in a way that one task's result is achieved with the aim of optimizing the next task. In contrast, an uncoordinated fashion simply uses the output of one task as the input of another without any coordination among them for obtaining a better solution. Generally, there are three ways to solve the overall problem: i) providing an exact solution, derived from an optimization problem, which is typically expressed as an ILP-based mathematical model [91], [92], [96], [98], ii) proposing heuristic algorithms that provide a 'good enough' solution [91], [94], since the execution time of the ILP models may easily become excessive for large instances of the problem, or iii) making use of metaheuristic algorithms such as genetic, tabu search, or simulated annealing algorithms, which aim at finding near-optimal solutions in a reasonable running time [99]–[101]. For the broad nature of the various objectives and the related problem constraints, the reader may refer to works [8], [31], [52] and the references therein.

Redundancy Allocation in NFV

NFV-based networks are expected to enable a multitude of telecommunication services that require high-availability levels [24]. Availability is threatened by failures, which are unavoidable events that operators need to account for when planning, designing, and operating the network infrastructure and the services running on top. A common way to deal with failures is providing fault-tolerance by leveraging redundancy. Employing redundancy involves the allocation of spare resources to compensate for failures of primary ones. Several studies have shown that given the resiliency challenges that NFV faces, simply allocating primary network function instances is not sufficient for satisfying the demanding availability levels of carrier-grade services [50], [51]. Therefore, allocating redundant instances, often called backup units, in addition to primary network functions becomes mandatory for guaranteeing high-availability figures.

The redundancy allocation problem, also known as the availability (reliability)-aware VNF deployment, is a resource allocation problem where the availability of the SFC requests needs to be factored in the problem formulation. It consists of the optimal *placement* of redundant VNFs into the network substrate and the optimal *assignment* of backup functions to network traffic flows that demand specific service chains, i.e., backup chain composition. The objective goal(s) can be varied and depend on the operator's needs. One objective can be maximizing the availability of the service chains subject to cost constraints or minimizing the number of deployed resources while meeting service availability demands [11]. However, there are certain important factors that need to be taken into account when performing NF redundancy allocation. In order to be effective, redundancy allocation schemes need to regard single points of failure that may affect both primary and backup instances. There may be service outages due to correlated failures as a result of topological dependencies, hence affecting both primary and backup instances at the same time. For example, in a data center network, a top-of-the-rack switch failure will impact the connectivity of all the servers placed on the rack. Therefore, if not carefully planned, correlated failures/service outages may hinder the effects of redundancy. An effective redundancy allocation deployment should deal with

such dependencies. One important drawback of redundancy is that it may be costly in terms of engaged resources. If not properly designed, the number of required backup instances may become unsustainable, especially when high-availability levels are required [34], [50], [55]. In addition, NFV networks make an extensive use of commodity servers, which may have diverse availability levels and capacities [102], [103], and service requests may have diverse availability requirements [11]. Therefore, the decision of *how* many backup instances to instantiate, *where* to place them, and *which* backup instance to use for each NF type, so that network resources are efficiently utilized and service availability demands are satisfied, is not trivial. A resource efficient and effective redundancy allocation scheme needs to incorporate the aforementioned aspects and at the same time be scalable and feature low time complexity.

In the related work section, a more detailed description concerning availability-aware resource allocation contributions will be provided since they fall within the scope of this work.

Chapter 3
Related Work

In this chaper, the current state-of-the-art literature related to the thesis objectives (introduced in Section 1.2) is revised. First, the related work regarding availability modeling and assessment of NFV-based services is introduced. Then, the main trends related to problem definition and proposed algorithms for computing availability-aware resource allocation in NFV are described. Additionally, open issues identified from the published literature and addressed by this thesis are summarized in the last part of this chapter.

3.1 Availability Modeling of NFV-based Services

The "softwarization" of hardware network appliances carries many benefits but at the same time brings additional challenges spanning from performance to resource allocation, service resiliency, security and privacy, and energy efficiency [8], [12]. One of the key aspects that can determine the success of its adoption is represented by the service availability [7], [11]–[14]. Indeed, the readiness for correct service delivery, i.e., availability, [10], is not only a user expectation, but often a regulatory requirement. In particular, traditional carrier-grade services provided by telco operators are services which are characterized by 5-nines availability; hence the transition to NFV-based carrier-grade services needs to be characterized by at least the same level of availability.

In regard to NFV availability modeling, there has been an increasing interest in proposing and investigating new models that characterize and quantify service availability. Gonzalez *et. al* [43] propose a stochastic activity networks model for assessing the steady-state availability of a virtual Evolved Packet Core (vEPC) supported by NFV. The authors analyze the system availability as a composition of several submodels constituting the vEPC (i.e., Serving Gateway (S-GW), Packet Data Network (PDN) Gateway (P-GW), Mobility Management Entity (MME)) together with a particular model representing catastrophic failures, i.e., multiple VNF failures due to natural disasters. They investigate relevant factors impacting the overall availability and highlight the need for adequate redundancy in order to cover individual failures and that catastrophic failures are a dominant source of system availability reduction. However, they assume the same SAN model for the different VNF submodels and same steady-state availability for software, hardware and hypervisor components, despite this not being a realistic case. In addition, their findings are related to the delivery of network functions and as the authors

state, the availability of end users' services needs to include the data center network topology integration.

A two-level hierarchical availability model of a network service in NFV architectures proposed in [104] aggregates RBDs on the higher level and SRNs on the lower level. Leveraging this hierarchical model, the authors evaluate the steady-state availability and perform a sensitivity analysis to determine the most critical parameters influencing the network service availability. Similarly, in [44], they extend the analysis by including the VIM functionality, as the entity responsible for the management of the network service resources, into the RBD. Their main findings indicate that a relatively small increment of hypervisor or VNF software failure intensity has a marginal effect on the service availability. In addition, they identify the most appropriate redundancy configuration in terms of additional replicas for providing fine-nines availability. The same authors model and assess the availability of an NFV-oriented IP multimedia subsystem (IMS) in [105]. Exploiting the same modeling technique, consisting of a hierarchical model composed of RBD and SRN, they assess the availability of the IMS and perform a sensitivity analysis on failure and repair rate of some of the IMS components. In addition, they identify the best k-out-of-n redundancy configuration for each elements of the IMS such that a five-nine availability is reached. Very similar to [105], in [106], the same authors assess the availability of a containerized IMS. The work performs a sensitivity analysis on failure and repair rate of some of the IMS components and identifies the best k-out-of-n redundancy configuration for each of the elements of the IMS such that a five-nine availability is reached.

In [45], the authors leverage universal generating functions (UGF) to evaluate system availability of a virtualized SFC. In particular, they extend the UGF formalism to a multidimentional version, which is capable of handling multiple performance figures. The work considers a multi-tenant SFC where the VNFs that compose the SFC are shared among multiple tenants. The availability model adopts parallel redundancy for each VNF and the the authors solve an optimization problem for a virtualized IMS case study by finding the best redundancy configuration, which satisfies 5-nines availability and at the same time provides the required processing capacity. In addition, a sensitivity analysis determines the range of failure and repair nominal values' variation that the best configuration can still handle.

A hierarchical model based on stochastic petri nets and RDBs is proposed for the availability analysis of a generic SFC in [107]. The model incorporates software rejuvenation mechanisms and VM live migration services. The work analyzes different deployment scenarios aiming at evaluating the steady-state availability for each of the scenarios. However, the analysis is limited to the availability evaluation of the system and lacks insights into the failure and repair dynamics of single system components and their relative impact on the overall availability.

Surprisingly, despite the importance and criticality that the MANO system has on the service orchestration and in particular on the fault management [30], [108], [109], there is very limited work that investigate resiliency of the MANO components. In [43] the model includes the MANO, as a central entity in the service life-cycle management,

but the relative model is rather simple and does not reflect current state-of-art implementations. In addition, the authors do not analyze the impact or effects that the MANO has on the service availability.

Soenen *et. al* [110] propose tunable and scalable mechanisms that make use of a shared state to allow the recovery of the various MANO components after they experience a failure event. In addition, they consider both centralized and distributed MANO deployed through a microservice-based architecture. Using a cost function they were able to find a trade-off between availability and fault recovery timing on the one hand and bandwidth and compute power on the other hand. While their suggested mechanism offers some important insights regarding the trade-off that operators need to choose in fulfilling service objectives, there are still several challenges that need to be addressed as specified by [30], where the most important is the MANO with no single point of failure. It is not clear how the different microservices have been deployed in a physical infrastructure since the author consider only redundancy on an instance level. Thus, more refined fault tolerance methods are needed, where both software and hardware redundancy levels are considered.

3.2 Availability-aware Resource Allocation in NFV

Planning for high availability involves the orchestration of NFV resources by allocating redundant resources to cope with unavoidable failures of the primary ones. Both the research community [8], [31] as well as standardization bodies [11] recognize that service availability, and in particular the service orchestration for achieving highly-available NFV services, represent an important challenge that needs to be addressed, such that a successful NFV adoption can embrace its potential benefits. As a result, there has been a continuous effort to investigate and propose efficient, scalable, and optimized algorithms addressing NFV resource allocation challenges while satisfying highly demanding availability needs. As presented earlier, the allocation of redundant resources in NFV encompasses two distinct, but entangled, problems which are the *placement* of redundant virtual instances running network functions and the *service chain composition* (or flow routing) that determines the traffic routing of service requests through specific functions composing an end-to-end service. Both problems can be considered as extended versions of two $\mathcal{NP}$-hard problems [31]: the virtual network embedding [95], [111] and the location-routing problems [112].

On the one hand, several studies formulate the problems as optimization models with specific objective functions and introduce ad-hoc heuristics for near optimal solutions [33], [35], [48], [50], [53], [54], [103], [113]–[118]. On the other hand, due to the problem complexity and the associated drawbacks such as high execution times for large problem instances, other studies simply propose heuristic or meta-heuristic algorithms [32], [34], [55], [102], [119], [120].

Fan *et al.* [32] present a heuristic algorithm that aims at minimizing the employed

physical resources for hosting network functions while satisfying reliability requirements. The authors propose a joint protection mechanism where a single instance protects two primary instances by reserving resources equal to the sum of the primary ones, and compare it with dedicated and shared protection where in the former, the same amount of resources allocated for the primary is used for the backup and in the latter, the amount of resources for the backup is the maximum among the two primary instances being protected. However, the study addresses only the chain composition problem and does not regard the backup placement. The same authors extend their contribution in [50] by proposing an optimized ILP model for minimizing the amount of backup resources. Nevertheless, only VM failures are considered and important factors in the VNF placement such as hosting node and link availabilities are ignored. In a later work [116], they propose a framework for minimizing resource usage while providing SFC availability demands. The main idea is to incrementally add backups to the VMs that mostly improve the SFC availability until the availability request is met. Nonetheless, in both works [50], [116], only on-site redundancies are allocated and the algorithms disregard, as the authors themselves highlight, the impact that events such as correlated failures may have within a data center. In order to guarantee carrier-grade service availability, it is important to also have backups distributed geographically [11], [13].

An investigation on the suitability of various data-center topologies for resilient deployments of service chains is carried out by Herker *et al.* in [119]. They consider both switch-centric, i.e. two or three-tier, and fat-tree topology, as well as server-centric architectures such as BCube and DCell [121], [122]. A heuristic algorithm that performs VNF placement and backup routing, which is based on a constraint-based shortest path algorithm, is proposed. It embeds backup chains through an iterative process until the chain availability is fulfilled. However, the analysis considers only hardware failures affecting servers and switches, and does not regard VNF instance failures. In addition, the problem setup assumes a heterogeneous system with devices having the same availability, which is different from real deployments where computing and network devices can have diverse availability figures.

Reference [33] addresses the redundancy allocation problem by constructing three different ILP models for the VNF placement and service chaining with resiliency protection against single node/link, single link, and single node failures. The ILP models have as objective function the minimization of the number of VNF nodes while satisfying latency constraints and achieving resiliency according to these different scenarios. The evaluation shows that providing protection against the considered scenarios comes with at least twice the amount of resources in terms of the number of nodes being deployed into the network. The same authors extend the investigation in [48]. They use the same core models and investigate the change of the objective function from minimizing the number of nodes to load balancing among link bandwidths. However, for both references, the models only place the VNFs without verifying that availability requirements are met with the assignment of the backup chains. Thus, these approaches do not actually guarantee the fulfillment of the availability requirement of user requests.

In [113], the reliability-aware service chaining deployment is formulated through an

ILP problem, dubbed REACH. The key idea of the algorithm is similar to the one proposed in [32] and [50], where repeatedly, the least available VNF of a service chain is provided with backup until the chain reliability demand is satisfied. The same authors address the reliable provisioning of carrier-grade service chains with resource sharing in [49]. An ILP model and customized heuristics, with the objective of guaranteeing carrier-grade reliability while taking into consideration the sharing of adjacent VNFs, are presented. The approach is similar to the one adopted in [32] where for the purpose of protecting two adjacent instances the maximum amount of resources among them is allocated in a single node. Nonetheless, the achieved service reliability is calculated considering only physical node reliability and does not regard VNF instances and network elements present in the primary and backup paths, i.e., forwarding nodes and links.

Differently to [32], [50] and [113], the authors of [55] optimize the design of [32] and propose a method that exploits a cost-aware importance measure to select the set of VNFs, which need to be backed up for enhancing the overall service reliability. On similar lines, the work in [118] proposes an algorithm for reliability-guaranteed VNF redundancy allocation that is based on a criticality importance measure (CIM). The allocation scheme exploits CIM for finding the best suited VNFs to protect and factors in the computational costs so that the output results in a cost-efficient and reliability-guaranteed VNF placement. In [102], VNF redundancy is allocated with the objective of reducing resource consumption while assuming heterogeneous VNF resource requirements. However, similar to [113], also the investigation of [55] and [102] is limited to three-nines SFC availability requests, which is far below the high expectations that carrier-grade services have.

A Mixed Integer Programming (MIP) for ensuring high availability when placing VNFs is formulated in [35]. The developed MIP model works only for small problem instances; hence the authors propose two heuristic approaches; a heuristic solution based on bin-packing and a meta-heuristic algorithm consisting in a variable neighborhood search. An extension of this work is carried out in [54] by proposing a flexible VNF placement, which enables VNF protection only if needed, and the redundant VNF can also be shared among multiple active instances. Although shared protection cannot be integrated in the MIP model due to increased model complexity, the authors propose a methodology for obtaining lower and upper bounds of the availability under shared protection. However, the investigations are performed on rather limited-scale networks, i.e., three NFVI PoPs containing up to 28 servers in total. Moreover, although the authors present a possible extension of the shared protection to cope with common cause failures, i.e., correlated, they do not investigate the impact that such failures have. Finally, both works perform only VNF placement and do not address the service chain composition problem.

Certainly, unless carefully planned, redundancy can come at a high cost in terms of employed resources. As in traditional IP/MPLS networks [123]–[125], several of the related works consider the sharing of resources in order to confine the amount of additional resources to be deployed for protection against failures [32], [49], [54], [115]. Common to these works is the assumption that only one service request, i.e., single-tenancy is served as the backup instance protects two adjacent instances of the same

service chain. A multi-tenancy approach is proposed in [53]. It allows the sharing of backup resources by multiple service requests and the analysis shows that it outperforms the single-tenancy approaches. Nonetheless, this approach is constrained by the placement of backup chains onto the same computing node, thus limiting the resource efficiency as backup chains are prevented from utilizing different hosting nodes and, in addition, the same node represents a single point of failure for the whole backup chain.

3.3 Open Challenges

From the background and the presented state of the art, the following open challenges regarding NFV service availability modeling and availability-aware resource allocation are identified. These challenges can be considered as "gaps" in the current literature, which this thesis aims to address, and they are also reflected in the research objectives in terms of aspects that shall be factored in the objective achievement.

CH1: *Exhaustive Failure Modes and Element Inter-dependencies*

Network services provided with the support of NFV infrastructure comprise a set of elements which are involved in the service delivery. As a result, an end-to-end service is subject to different types of failures that may affect any of the elements involved in the service delivery. Still, the current literature contributions are rather limited in the kinds of system elements and the inter-dependencies between them. A comprehensive availability model of NFV-supported services should include and take into account the failure and repair processes of the involved elements on both hardware and software levels. In particular, the inter-dependencies of these elements need to be fine-grained/reflected. To illustrate, consider a failure of the physical hardware, which will impact all software elements running on top of it. The hardware repair does not bring the system up and running unless software elements are rebooted/restarted. This last operation is often omitted in the current literature. A crash of the operating system (OS) does bring down the VNF software yet, the reboot of the OS is not enough for the system to be considered operational unless the VNF software is restarted too. In addition, the failure of the OS should not influence the status of the underlying hardware since the latter may fail independently on whether the OS is running or not. Such inter-dependencies are often omitted in the availability models that the current state-of-art presents, see for example [43], [45], [104], [107].

CH2: *NFV-MANO System Availability Assessment*

Ensuring the resiliency of an NFV service from its instantiation throughout its operation requires adequate management of the entire network service. The NFV MANO, being a logically-centralized entity that maintains a global view of the network, is responsible for the correct orchestration and management of end-to-end services and for ensuring service availability adherence to terms specified in SLAs [30]. In particular, tasks like VNF re-instantiation/re-creation, VNF scaling, VNF

migration, or failure detection and containment are core operations that are triggered by MANO components for ensuring the expected NFV service availability. In addition, any misoperation or logical/physical faults in the MANO components may jeopardize and severely impact the provisioning of network services [108]. Despite the importance and criticality that the MANO system has on the service continuity, most of the related work focuses on the data plane availability and disregards the control plane. The only related work that models the availability of the MANO [43] relies on a very simple model, which does not fully represent currently developed standard-compliant frameworks like [126], [127]. Moreover, the investigation lacks a thorough analysis of MANO availability and the factors that mostly impact it. Consequently, it is pivotal to investigate and identify these factors through a proper evaluation of MANO availability by exploiting more realistic models derived from standardized implementations.

CH3: *Network-aware Modeling*

In its specification regarding end-to-end reliability [29], ETSI emphasizes that a correct availability evaluation should incorporate all the service elements and components involved in the end-to-end delivery. The supporting infrastructure, both computing and transport network, and the inter-dependencies with the software providing the service, i.e., VNFs, are required to be taken into account when estimating the service availability. As presented in the related work section, several previous works have quantified the availability of NFV-oriented services, either in "general" terms or by selecting specific NFV service use cases [43]–[45], [104], [106], [107]. Nevertheless, none of these works has performed a comprehensive assessment of an end-to-end NFV service availability since they lack key service elements like physical network links and forwarding/routing devices, which are essential networking elements inter-connecting VNFs composing a service chain. As highlighted also by Gonzalez *et. al* [43], predicting the availability of end user services requires the incorporation of both datacenter and transport network, i.e., the network that interconnects the datacenters since VNFs can be geo-distributed. Henceforth, integrating the network and the component dependencies in the availability model remains a fundamental endeavor for a detailed end-to-end service availability assessment.

CH4: *Network Topology Dependencies Impact*

The availability-aware NFV resource allocation problem regards the optimal placement of redundant VNFs into the network substrate and the optimal assignment of backup functions to network traffic flows that demand specific service chains. While redundancy is the "de-facto" technique for achieving high availability, in order to be effective, the allocation schemes need to regard single points of failure that may simultaneously affect both primary and backup service chains. In [29] ETSI recommends anti-affinity placement policies, i.e., deployment of primary and backup VNFs into separate computing nodes so that they will not experience a simultaneous failure. However, applying such rules may not be enough

because the operator needs to ensure that a failure of a primary VNF will not impact its respective backup VNFs (and vice versa) and both should not be subject to a common failure or outage mode. For example, in a data center network, a top-of-the-rack switch failure will impact the connectivity of all the servers placed on the rack, hence placing them into separate servers but in the same rack may not be sufficient. In particular, for cases where high-availability levels are demanded which in turn may require more than one backup instance, the failure of a primary resource should not impact any of the relative backup instances. Several of the related works, see for example [32], [34], [55], [102], [113], consider the deployment of primary and backup VNFs into different nodes, yet they do not check any topological dependencies that may affect both primary and backup chains. Therefore, unless carefully designed, such correlation may undermine the benefits of redundancy and an effective redundancy allocation deployment should be aware of such dependencies.

CH5: *Efficient Resource Utilization*

An important limitation of redundancy is that it may be costly in terms of resource utilization. If not accurately planned, the number of required backup instances may grow up to an unsustainable level, i.e., more than twice the required resources for primary allocation [33], [34]. This can be further exacerbated when high-availability levels are required [50], [55]. As a result, this may lead to a limitation of the network capability to accommodate new flows and thus restrict the operator's acceptance of new incoming service requests. To this end, it becomes essential to design and plan redundancy allocation in a cost-efficient manner such that resource utilization can be maximally achieved. A common way to achieve this is by sharing redundant resources among network functions, similar to what traditional IP/MPLS-based networks have been doing [123]–[125]. The work in [32] exploits a joint protection mechanism that protects two adjacent VNFs, of the same service chain, by allocating a common VNF that is shared among the two as a backup. Differently, in [53] a multi-tenancy concepts is exploited where multiple service chain requests, i.e., multi-tenancy, may use the same VNF for backup purposes. Yet, both approaches perform backup sharing of one VNF instance with dedicated capacity being the sum of the capacities of the respective primary resources. The *resource overbuild*, as one important figure of merit for resource efficiency, which expresses the amount of extra resources needed for providing protection as a percentage of the required amount without protection [124], would equal 100% in case only one backup is required. Therefore, designing and adopting approaches that achieve lower resource overbuild figures would provide significant benefits to network operators both in terms of employed resources and increased capacity in accommodating new requests.

Chapter 4
Contributions and Concluding Remarks

In this section, a summary of the contributions that this thesis advances, followed by concluding remarks, are presented. The paper contributions and how they relate to research questions, objectives, and identified open challenges, are discussed in Section 4.1. Then, the summary of each of the included papers is presented in Section 4.2. Section 4.3 concludes the thesis by outlining the main remarks and Section 4.4 discusses suggestions for future work.

4.1 Summary of Contributions

The author of the present thesis wrote and contributed to 6 publications in conference proceedings and journals, out of which 5 are already published and 1 is currently submitted for peer reviewing. Table 4.1 presents a list of the publications included in the thesis and the order in which the included papers are listed is not necessarily chronological but rather related to the research objectives.

The contribution of the thesis aims at providing an answer to the research questions, and at filling the gap related to the open challenges. The research objectives, presented in Section 1.3 and pursued in order to answer the research questions, are tightly coupled with the open challenges identified in Section 3.3, since the challenges are considered as important aspects that shall be taken into consideration for achieving specific objectives.

Figure 4.1 presents the progression and the mapping of the research effort to the research questions, and the relationship between the research questions, research objectives, and the open challenges. The figure also highlights the core contribution of the papers and how the different papers relate to each other. To better illustrate the relationship, let's consider the first research question RQ1. In order to design analytic models that allow to characterize in detail and assess the availability of NFV-based services (RQ1), there is a need to develop availability models that allow to describe detailed failure dynamics of the involved service elements (OB1) and accomplish a quantitative assessment and analysis of the service availability for identifying availability bottlenecks (OB3). These research objectives are addressed considering important aspects such as characterizing significant failure modes and integrating element inter-dependencies (CH1), and including the assessment of the NFV-MANO system availability (CH2). Following the same illustration, Papers A, B, and C answer to the first research question

TABLE 4.1: List of publications included in the thesis.

Paper	**Title** — Authors — *Conference/Journal*
Paper A	**Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes** Besmir Tola, Gianfranco Nencioni, Bjarne E. Helvik, and Yuming Jiang *IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, 2019, pp. 1-5.*
Paper B	**On the Resilience of the NFV-MANO: An Availability Model of a Cloud-native Architecture** Besmir Tola, Yuming Jiang, and Bjarne E. Helvik *IEEE Proceeding of 16th International Conference on the Design of Reliable Communication Networks (DRCN), Milano, Italy, 2020, pp. 1-7.*
Paper C	**Model-Driven Availability Assessment of the NFV-MANO with Software Rejuvenation** Besmir Tola, Yuming Jiang, and Bjarne E. Helvik *IEEE Transactions on Network and Service Management, 2021* *Early Access - doi: 10.1109/TNSM.2021.3090208*
Paper D	**Network-Aware Availability Modeling of an End-to-End NFV-Enabled Service** Besmir Tola, Gianfranco Nencioni, and Bjarne E. Helvik *IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1389-1403, Dec. 2019.* *doi: 10.1109/TNSM.2019.2948725*
Paper E	**Towards Carrier-Grade Service Provisioning in NFV** Yordanos T. Woldeyohannes, Besmir Tola, and Yuming Jiang *IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, 2019, pp. 130-137.*
Paper F	**CoShare: An Efficient Approach for Redundancy Allocation in NFV** Yordanos T. Woldeyohannes, Besmir Tola, Yuming Jiang, and K.K. Ramakrishnan *Submitted to IEEE/ACM Transactions on Networking*

by proposing SAN-based availability models that incorporate the failure dynamics of all the NFV elements involved in the service provisioning, i.e., computing infrastructure, supporting OS, virtualization software, VNFs, and the NFV-MANO, and also assess the system availability with the aim to identify critical NFV elements and suitable fault-tolerant mechanisms that allow to achieve high-availability levels. Similarly, challenge CH3 serves as an essential dimension for performing end-to-end NFV-enabled service availability modeling and assessment, i.e., achieving objective OB2 and OB3 which in turn will provide an answer to the second research question. Such objectives are accomplished by the contributions of Paper D. In the same line, challenges CH4 and CH5 represent critical design factors for enabling scalable, cost-efficient, and effective NFV redundancy allocation, i.e., research objective OB5. In the remainder of this section, the distinct contributions of each of the included papers are described.

In this thesis work, six major contributions are made. *First*, the availability of a service function chain provided through an NFV infrastructure is modeled and evaluated

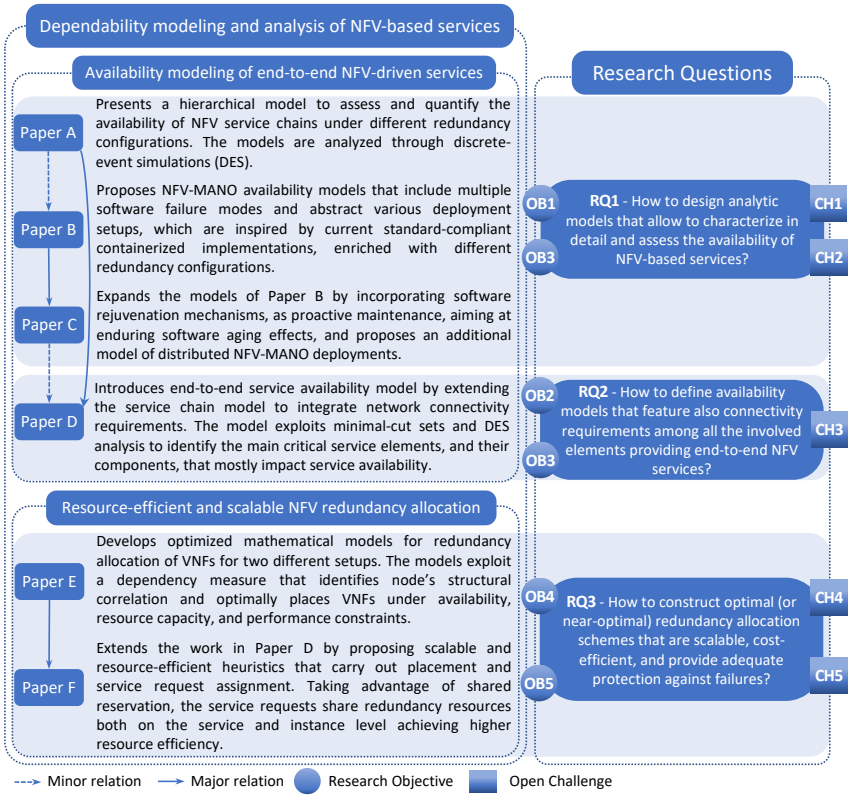


FIGURE 4.1: Outline of paper contributions and their mapping to research questions, objectives, and open challenges.

in Paper A. The considered service chain is assumed to incorporate different redundancy configurations, called availability modes, which are portrayed through distinct models featuring diverse recovery mechanisms. The models are defined using a two-level hierarchical approach. On the low level, the individual behavior of the NFV elements are abstracted using Stochastic Activity Networks and such models are composed through a Replicate and Join formalism, which represents the high level. The service chain model includes several VNFs, composing the service chain, and the MANO, as the central entity responsible for the service lifecycle operation. The distinct models, mirroring the availability modes, are designed using Möbius software tool [71] and subsequently solved through a discrete-event simulator, which is integrated in the tool. An extensive sensitivity analysis is performed for identifying the main critical failure and repair parameters. Moreover, insightful observations are made by comparing the different redundancy setups aiming at identifying the most suitable configuration for different availability levels.

The MANO model presented in Paper A abstracts a hypervisor-based virtualized MANO and the investigation focus is more on the impact that the MANO operational status has on the service availability rather than on the failure dynamics of the MANO itself. Inspired by several current open-source MANO development projects (e.g., OSM [126], OpenBaton [127]), Paper B makes the *second* contribution. It proposes an availability model for a cloud-native MANO implementation, i.e., containerized deployment, and performs an extended investigation on the failure and repair dynamics, sensitivity analysis, and steady-state availability assessment for different containerized deployments of the MANO. A special highlight is the impact of software-aging effects on the availability and the relationship that software aging has with software aging-induced failures. The research is further extended in Paper C by modeling a more realistic software aging behavior and the effect that software-aging error accumulation has on the software failure intensity. The paper also investigates additional deployment options for component-wise MANO software and distributed MANO implementations. The different models, abstracting various MANO deployments, combine software rejuvenation mechanisms to withstand software-aging effects. Moreover, a fault-injection based experimental campaign is performed on a real-life MANO deployment aiming at retrieving a number of realistic recovery parameters. Sensitivity analysis allows us to identify optimal rejuvenation policies for achieving maximal steady-state availability and investigate the impact that software-aging intensity has on the system availability. This refined MANO availability model and the additional models, abstracting various deployment options together with the relative analysis, represent the core of the *third* contribution provided by this research work.

The models proposed in Paper A, B, and C provide important observations regarding dependability bottlenecks of NFV-based services and suggest how to overcome them by providing adequate protection against failures. However, they are not detailed in regard to portraying end-to-end NFV-enabled services because they lack key network components that realize the interconnection and composition of service chains in a distributed infrastructure, such as NFV networks. To address this issue, the *fourth* contribution of this thesis work is made in Paper D. It consists of an availability model that incorporates connectivity requirements among NFV and network elements. The model relies on a two-level approach. On the low level, SAN-based dynamic models abstract both NFV and interconnecting devices. The high level, i.e., structural model, exploits mincut analysis to derive end-to-end service availability by including both data and control/management plane availability. An extensive numerical analysis identifies critical components and their relative failure/repair parameters, which can degrade or enhance the end-to-end service availability the most. Moreover, the models also include the integration of an SDN-enabled network architecture, where in addition to networking devices, i.e., SDN-enabled switches, the presence of an SDN controller contributes to additional connectivity requirements. The resultant analysis provides useful insight on the vulnerability of both traditional and SDN-based NFV networks by identifying availability bottlenecks and adequate redundancy levels for achieving highly available NFV services.

Model-based availability prediction and analysis can aid an operator on deciding an

efficient redundancy strategy, yet, redundant resources need to be provisioned in the network infrastructure such that the traffic flows can be properly processed and served. Resource provisioning (or allocation) in NFV consists in a set of decisions that an operator needs to perform for establishing, among others, the right amount of resources and where in the network these resources shall be deployed such that availability and performance requirements that services demand are to be satisfied. In particular, availability-aware resource allocation strategies, which decide where and how many VNF instances are needed to be deployed, will determine the level of availability that services can achieve. Concerning this, Paper E provides the *fifth* contribution, which consists of the development of mathematical models for allocating redundant resources in NFV networks aiming at minimizing resource consumption while satisfying, among others, flow availability demands. In particular, two different ILP-based models are proposed. They are referred to as *AllOne* and *AllAny*, and their main distinction lies on the fact that in the former model, redundant VNFs, composing the backup service chain, are allowed to be hosted on only one backup node, whereas the *AllAny* model allows the allocation of backup resources in multiple backup nodes. Both models feature an algorithm that enables the quantification of the structural dependency among network nodes as a result of the network topology. This algorithm is exploited in the placement decisions such that correlated failures or outages, due to the inherent topology structure, are avoided. Both model performances in terms of resource utilization, cost-efficiency, and additional delay due to path stretch are evaluated. However, due to the complex nature of the problem (refer to Section 2.4), while the models are well suited for small- to medium-scale problem instances, they become intractable for large instances. To cope with this problem, Paper F extends the work in Paper E by proposing ad-hoc heuristics that offer better algorithmic scalability. The heuristic algorithm, namely *CoShare*, places backup instances and assigns them to service requests, i.e., flows, with the focus on avoiding simultaneous failures of both primary and backup chains, which can be due to network structural dependencies, while ensuring that service chain availability is satisfied. Moreover, the paper exploits a flow-level resource sharing mechanism, which contributes with higher resource efficiency, compared to previous related investigations, by achieving significantly lower resource overbuild, i.e., less than 100%. This constitutes the *sixth*, and final, contribution of this thesis work.

Five of the included papers have been subject to international peer-reviewing in well established publication venues. Papers A, B and E are published in conference proceedings, while Papers C and D are published in IEEE transactions journals. Paper F is currently submitted for review in the *IEEE/ACM Transactions on Networking* journal.

In the following, a summary of the included publications is presented along with the main findings.

4.2 Summary of the Papers

This section presents a summary of the Papers included in the contributions of the thesis and lists supplementary publications that are not included as contributions in the thesis.

The paper summaries are introduced in the same order as illustrated in Figure 4.1.

PAPER A: Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes
Besmir Tola, Gianfranco Nencioni, Bjarne E. Helvik, and Yuming Jiang
IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, 2019, pp. 1-5.

Paper Summary

Modeling techniques are convenient tools that may help to quantify and analyze system availability, in particular when technical systems are large and complex [40]. A hierarchically-composed availability model for NFV-enabled service function chains and the MANO system is proposed in this paper. The model is constructed using a combination of state-space and hierarchical models, and features various failure modes, including hardware, operating system, hypervisor layer, VNF, and MANO software. Specifically, Stochastic Activity Networks are used to characterize failure dynamics of the components and their related repair mechanisms. The relationship and dependencies among individual service components are modeled through a state-sharing Replicate and Join formalism. Moreover, three distinct models, referred to as availability modes, which exploit different recovery mechanisms, namely Standard Availability, Cold Protection, and Hot Protection, are designed. A single VNF is assumed to be deployed as a load-sharing cluster where several VNF units, composing the cluster, are needed to satisfy a certain load demand and the SFC is assumed to be composed of three VNFs. A single VNF is considered to be operational if at least N out of K (with $K > N$) units are working. On top of load-sharing, we consider the different availability modes where M redundant units provide protection to the load-sharing cluster. As a result, by tuning the K, N and M parameters we investigate different redundancy configurations and their achievable steady-state availability. In a similar fashion, the MANO is implemented as a load-sharing cluster within each of the availability modes.

The models are solved through discrete-event simulations, and sensitivity analysis is carried out for each of the availability mode. The numerical analysis led to interesting observations regarding the most critical failure and repair parameters that can impact service availability and helped identifying the most appropriate redundancy setup that achieves 5-nines availability. Specifically, service availability is more vulnerable to VNF software failure intensity compared to the less susceptible hardware or operating system failures. Moreover, a small resource overprovisioning on the VNF cluster, i.e., one additional VNF unit, achieves an increase of up to three orders of magnitude of the service availability. On the contrary, even for a higher MANO overprovisioning, the service availability gain is not significant.

PAPER B: On the Resilience of the NFV-MANO: An Availability Model of a Cloud-native Architecture
Besmir Tola, Yuming Jiang, and Bjarne E. Helvik

IEEE Proceeding of 16th International Conference on the Design of Reliable Communication Networks (DRCN), Milano, Italy, 2020, pp. 1-7.

Paper Summary

Failures on the MANO could affect the functionality of all the network and potentially impact the service delivery by inducing severe outages, which sometimes might be hard to deal with [108], [128]. It is thus important to identify dependability bottlenecks and ensure a highly dependable management and orchestration system.

Paper B takes a model-driven approach to perform a quantitative assessment of the steady-state availability of a containerized MANO. The work proposes a SAN-based availability model, inspired by open-source ETSI-compliant architectures adopting cloud-native designs, and performs a sensitivity analysis aiming at identifying the factors that mostly impact system availability. The model incorporates different failure modes and relative repair mechanisms on both software and hardware level of the MANO framework. In particular, the model integrates aging and non-aging related software failures and investigates their impact on the overall MANO availability. The basic model, which mirrors default deployments, is further expanded to portray more advances container deployments that combine different redundant configurations on the host or software level.

The numeric analysis indicates that adopting containerized technologies with standard deployments having both single and multiple software replicas deployed into a single physical node is not sufficient for achieving "5-nines" availability. The analysis also indicates that non-aging-related software failures and software repair intensity stand out as key important failure and repair parameters, respectively. When clustering mechanisms such as Docker swarm mode with distinct worker and manager nodes are adopted, we observed that the MANO availability is further increased and the critical parameters become less significant when multiple MANO container replicas are engaged. Software aging may have a considerable impact on the availability and the related failure rate impact magnitude depends on the time it takes for the software to age. The shorter the time it takes to experience aging symptoms the higher will be the impact induced by aging-caused software failures.

PAPER C: Model-Driven Availability Assessment of the NFV-MANO with Software Rejuvenation
Besmir Tola, Yuming Jiang, and Bjarne E. Helvik
IEEE Transactions on Network and Service Management, 2021
Early Access - doi: 10.1109/TNSM.2021.3090208.

Paper Summary

This paper further expands the investigation of Paper B in several directions. First, it proposes a modified software aging model that resembles a more realistic aging behavior. The model proposed in Paper B can be considered as a "one-shot" model representing

a failure rate increase due to software aging once a certain aging threshold is exceeded. However, this representation slightly deviates from a realistic behavior where failure intensity follows a gradual growth as the number of accumulated aging errors increases. The new MANO model, proposed in this paper, mirrors the effects on the software failure intensity due to the software aging phenomenon by simulating a continuous increase of the failure intensity, which is directly proportional to the amount of accumulated aging errors. In addition, the proposed model also includes a proactive software maintenance mechanism, commonly referred to as software rejuvenation, aiming at relieving the effects of the manifestation of the software aging phenomenon. Moreover, experimental trials on a real MANO testbed allows to retrieve empirical recovery parameters regarding a number of system elements. Finally, in addition to the readjusted models of Paper B, the work proposes two other models for portraying distributed MANO deployments, featuring redundancy configurations on both hardware and software levels, and a software component-wise MANO model.

The numerical results underline that non-aging-related software failures and software repair rates stand out as main detrimental parameters. However, adopting clustering mechanisms, which provide redundancy on both hardware and software system components, alleviates the impact of such critical parameters and increases the system availability. In addition, software aging can have a significant impact on the MANO availability and highly utilized software can benefit from optimized rejuvenation scheduling policies by achieving up to 61% uptime increase. Moreover, the analysis of the MANO software with distinct components, i.e., component-wise MANO, shows that modeling of the MANO software as a single component yields a representative analysis of the systems steady state availability.

PAPER D: Network-Aware Availability Modeling of an End-to-End NFV-Enabled Service
Besmir Tola, Gianfranco Nencioni, and Bjarne E. Helvik
IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1389-1403, Dec. 2019.

Paper Summary

The models proposed in Papers A-C provide insights into failure and recovery dynamics of NFV-enabled services, and the system that manages and orchestrates the services. However, the delivery of end-to-end NFV-enabled services involves also network devices, such as routers and physical links, that assist the interconnection of NFV elements, and an exhaustive availability assessment of end-to-end NFV services should also include such elements [11], [29].

Paper D proposes an availability model that compensates the lack of regard of network elements in the previous models, and integrates the connectivity requirements that the interconnection of NFV elements imposes such that an end-to-end NFV service availability assessment can be performed. The modeling approach has two levels; i) the *structural* model of the network topology and ii) the *dynamic* models of NFV-based service

elements. The first level exploits structural analysis based on minimal-cut sets [129] and the second level consists of SAN-based dynamic models of each of the network and NFV elements. The structural analysis allows the definition of connectivity requirements and helps determine the most critical elements involved in the end-to-end service delivery. The analysis defines minimal-cut sets of elements that are required to be operational such that the end-to-end service can be considered available. The dynamic models characterize and quantify the steady-state availability of each of the involved elements and the end-to-end service availability is evaluated by merging the two levels. The *inclusion-exclusion principle*, which is a probabilistic technique to obtain the elements in a union of finite sets, is used to merge the two levels and define the end-to-end service availability. Moreover, the work also investigates the impact that the integration of NFV with a Software-Defined Networking-enabled network can have in terms of service vulnerability to failures of SDN elements. i.e., the SDN controller and SDN-enabled switches.

An extensive numerical evaluation sheds light on the impact that service elements, element's components, and single or double element redundancy have on the end-to-end service availability. Observations highlight that in case of traditional networks, the service availability is mostly negatively impacted by the availability of IP routers and VNFs, and this kind of impact can be as large as two orders of magnitude. Making use of more robust IP routers brings significant benefits only when this is accompanied with redundant NFV elements and this allows achieving values of 5-nines availability. Compared to a traditional network, an SDN-integrated solution degrades the service availability because of the introduction of additional connectivity requirements, i.e., SDN switches in the end-to-end path are required to have a working path with their SDN controller. In particular, the SDN controller is the most critical elements that can even hinder the advantages of redundant NFV elements. From an element's component perspective, the service availability is significantly impacted by the degrading of router hardware and operational and management (O&M) software failure intensities.

PAPER E: Towards Carrier-Grade Service Provisioning in NFV
Yordanos T. Woldeyohannes, Besmir Tola, and Yuming Jiang
IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, 2019, pp. 130-137.

Paper Summary

One of the main benefits of NFV is the flexibility to deploy VNFs across different computing infrastructures and to be able to migrate them when network demand changes. The choice of where to place VNFs, how much resources to allocate, and how to steer traffic through them is regarded as the NFV resource allocation problem [31] and when a particular emphasis is put on ensuring a certain availability level, it is commonly referred to as the redundancy allocation problem or availability-aware resource allocation.

Paper E proposes optimized algorithms for allocating redundant resources, otherwise called backup, to NFV service requests such that service availability requirements

can be met. The problem is formulated as a mathematical multi-objective optimization problem aiming at minimizing the total number of VNF backup instances, minimizing the number of backup hosting nodes, and minimizing the backup chain path delay. The said objectives are subject to various system constraints including service availability, resource capacity, service flow routing, and function statefulness. Specifically, two optimization models are considered where the first model assumes that all the services of a chain are backed-up using the same hosting node, i.e., *All-One* model, and the second model assumes that all of the functions of a given service chain request can be hosted into separate backup nodes, i.e., *All-Any* model.

It is common that for simplicity, network failures are often assumed to happen independently. Nonetheless, several studies recognize that correlated failures are non-negligible events that can have significant effects on the network functionality [37], [130]. A novel contribution of this paper, compared to related work, is the proposal of an algorithm that identifies nodes that are inherently correlated due to the network topology structure. For each node, the algorithm identifies a set of nodes whose operational status is affected by the failure of a primary node, i.e., nodes hosting primary functions, and these nodes are to not be considered as candidate backup nodes in the redundancy placement decision of the services running in the primary one. This way, the algorithm is exploited in the placement decision in order to avoid simultaneous failure or outage of both working and backup chains.

The performance analysis outlines the benefits that structural correlation analysis brings in terms of the ability of flows to achieve higher availability compared to disregarding structural correlation. Moreover, the evaluation investigates the trade-off between cost and system performance and highlights that for the considered setup and carrier-grade availability targets, employing more reliable servers is advantageous only if their related cost is at most twice as less reliable ones. Otherwise, the total expenditure will exceed the cost of using only less reliable, but more economic, servers for reaching the same availability target.

PAPER F: CoShare: An Efficient Approach for Redundancy Allocation in NFV
Yordanos T. Woldeyohannes, Besmir Tola, and Yuming Jiang
Submitted to IEEE/ACM Transactions on Networking. A preliminary version is available on arXiv: `https://arxiv.org/pdf/2008.13453.pdf`

Paper Summary

The optimization problem formulated in Paper E finds an optimal placement of VNFs in the network substrate and assigns service requests to the correct service chains, i.e., performs flow routing. However, the problem complexity is $\mathcal{NP}$-hard, hence limiting the algorithmic applicability to being intractable for large-scale problem instances. As a result, the optimization model does not scale well for large instances and in particular, the execution time for finding an optimal solution becomes unsustainable.

Paper F tackles the problem-scalability drawback by proposing customized heuristics that assemble the subtleties of the optimized problem. Specifically, the approach, named

CoShare, considers the various fundamental aspects of the original problem, including the network structural dependency, the heterogeneity of nodes and NF instances, and service availability requirements, and does scale well even for large system setups. A distinctive feature of the heuristic algorithm is the introduction of a novel idea, referred to as *NF shared reservation*, for achieving higher efficiency in regard to resource utilization. The idea is based on the consideration that flows requiring service chains, which are composed of VNFs that in turn are deployed into non structurally correlated nodes, can share common reserved capacity for fault-tolerance purpose. This is because the failure of one node will not impact the other and thus, the common shared capacity can be used to provide redundancy protection for the instance hosted in the failed node. Therefore, the capacity shared among non-structurally correlated flows shall suffice because they will unlikely fail simultaneously. In addition, for comparison, CoShare considers also the case where dedicated reserved capacity is allocated for backup instances.

A number of experiments, characterizing different system setups, provide insights regarding the performance of the algorithm when compared to the optimized project. Moreover, simulated experiments concerning the performance of *shared reservation* in terms of resource overbuild, i.e., the ratio of the amount of backup resources over the amount of primary resources, and flow acceptance ratio are also carried out. The experimental results show that CoShare with *shared reservation* requires less resource overbuild compared to both *dedicated reservation* version of CoShare and the optimized project. In particular, for the experimented networks, this results in almost halving of the necessary number of backup instances compared to the other two approaches while still satisfying the service availability demands. In addition, while solving the optimized problem takes more than 12 minutes in a standard Intel® multi-core workstation, CoShare generates the solution in less than 1 second. A closer inspection of the performance of CoShare with *shared reservation* compared to the dedicated version highlights the advantages of the former also for situations where highly demanding services are expected to be executed in the network. Specifically, for 700 flows, all demanding 5-nines availability, *shared reservation* results in 93% overbuild compared to 178% achieved by *dedicated reservation*. Furthermore, *shared reservation* achieves a higher acceptance ratio, i.e., the ratio of the number of accommodated over the total number of service requests, compared to the *dedicated* case, 100% and 59%, respectively.

Secondary papers were also published while working on the doctoral thesis. Since their relevance to this thesis is marginal, only references and abstracts are included in **Part III**, and Table 4.2 lists the publications.

4.3 Conclusions

Modern society is becoming more and more reliant on the services provided by telecommunication networks. The global pandemic situation, and its lockdown consequences, have accentuated even more the network dependency as a result of everyday life activities like working or leisure being performed through connected services. With the

TABLE 4.2: List of supplementary publications not included in the thesis.

Paper	**Title** — Authors — *Conference/Journal*
Paper G	**Failure Process Characteristics of Cloud-enabled Services** Besmir Tola, Yuming Jiang, and Bjarne E. Helvik *Proceeding of the 9th Workshop on Resilient Networks Design and Modeling (RNDM), Alghero, Italy, 2017, pp. 1-7.*
Paper H	**On Monolithic and Microservice Deployment of Network Functions** Sachin Sharma, Navdeep Uniyal, Besmir Tola, and Yuming Jiang *Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, pp. 387-395.*
Paper I	**Keeping Connected When the Mobile Social Network Goes Offline** Øystein Sigholt, Besmir Tola, Yuming Jiang *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 2019, pp. 59-64.*

increase in importance and usage of communication services, the end-user's expectations are also growing, not only with respect to data rate capabilities but also in regard to the level of dependability offered by the services. The forthcoming 5G mobile networks are anticipated to meet extreme user's expectations by offering a whole new level of highly performing and dependable services. At the core of 5G networks lies the successful adoption of the NFV paradigm, which brings into play the desired network programmability, flexibility, scalability, and economies of scale. NFV alters the rigid deployment of traditional network appliances by separating the software implementing the network function from the underlying hardware. This is achieved through the exploitation of server virtualization technologies, which enable the execution of network functionalities in virtualized environments. However, a successful adoption of NFV is tightly coupled with several challenges and service availability plays a crucial role. With more and more end users taking for granted network and service continuity, ensuring highly available NFV-based services becomes of paramount importance for embracing the promising benefits of NFV.

The target of this thesis work is at *modeling, analyzing and providing highly available NFV-based services*. To achieve this objective, this work has focused on three main aspects: i) modeling and quantifying the availability of service function chains provided through NFV infrastructures, ii) integrating network connectivity requirements into the availability model for evaluating end-to-end NFV-based services, and iii) propose solutions with fault-tolerance capabilities by allocating redundant resources to NFV services. The different aspects have been tackled by proposing various solutions, which are summarized as follows:

- A two-level hirearchical model is proposed to quantitatively assess the availability of NFV-supported services. On the low level, the model consist of SAN-based dynamic models that abstract the failure and repair dynamics of the NFV elements such as VNFs, virtualization layer, supporting software, hardware infrastructure, and the MANO framework. On the high level, the model exploits a Replicate

and Join formalism to compose network service chains. Based on the model, the service chain availability analysis is exemplified for three distinct recovery strategies and results show that carefully selecting the recovery strategy not only enhances the service availability but also reduces the impact that failure and repair parameter variations have on the availability.

- Given the central role of the MANO in the NFV architecture for ensuring, among others, correct service delivery, resource and fault management and configuration, it is crucial to employ a robust MANO framework. Inspired by current cloud-native MANO software implementations, this research work proposes a model to predict and analyze the availability of a hypothetical containerized MANO system. The model serves as basis for abstracting different redundancy configurations and MANO deployments that can be actualized in line with current containerized deployment practices. Moreover, the model features software rejuvenation mechanisms for mitigating software aging effects. The sensitivity analysis sheds light into the most critical availability bottlenecks and suggests suitable protection mechanisms that can help improve MANO availability.

- One important facet of NFV architectures is the ability to deploy, operate, and migrate network functions on-the-fly and anywhere in the network. Such network flexibility implies that VNFs can be distributed onto the network and the composition of specialized service chains requires traffic steering into specific geographically distributed VNFs that compose end-to-end services. From an availability evaluation perspective, it is required that all the service elements, which process and carry traffic, ought to be operational so that the service can be deemed available. This condition imposes connectivity requirements among the elements such that an accurate availability estimation can be performed. This work proposes an approach for modeling and assessing service availability by integrating connectivity requirements among NFV and network elements like VNFs, MANO, routers/switches, and network links. The approach consists of a two-level model where on the lower level, SAN-based models characterize failure and repair dynamics for each singular element and on the higher level, the structural model, which is based on min-cut analysis, imposes connectivity requirements. The steady-state availability of the end-to-end service is retrieved by merging the two models through the inclusion-exclusion principle.

- Model-driven availability assessment can be an excellent tool for quantifying service availability, and its associated threats, yet, a network operator needs also to decide how to provision highly available services by allocating redundant resources to cope with failures of primary ones. To this end, the operator needs to perform some decision making in regard to *where* and *how many* redundant instances are required to be allocated in the network for satisfying traffic flow's availability requirements while still optimizing other system objectives such as cost minimization, maximizing resource efficiency and so forth. This thesis contribution proposes a set of both optimization models (ILP-based) and customized

heuristic strategies that implement redundancy allocation for NFV services. In particular, the heuristic scheme is scalable enough for executing large problem instances in polynomial time and achieves better resource efficiency compared to related literature by exploiting a network function sharing mechanism.

4.4 Future Work

To finish Part I of this work, some future research directions are presented in this section.

First, model-driven research is based on data inputs that are fed to the model and metrics of interest are retrieved by solving it. Although the parameter values utilized in this work are retrieved from previous literature that investigates systems of similar nature and complexity, it would be extremely interesting to analyze real data from NFV production systems. This would not only help refine and validate the model parameters, but would also allow the characterization of failure and repair processes of real NFV network elements. Unfortunately, NFV deployments are still at its infancy for many of the major operators. In addition, it is difficult to retrieve failure data from real systems, as network operators and providers are, understandably, reluctant to release this type of information.

Another interesting direction to explore would be to characterize and assess NFV-enabled services provided through the interaction of multi-domain and multi-provider networks. For example, 5G networks are expected to provide tailored services through a concept called network slicing that will span across multiple network domains and involve multiple providers. A core issue of network slicing is the isolation of slices, which can seriously impact their dependability attributes [131]. The domain integration points could represent potential dependability bottlenecks. Hence, it would be worthy to explore the main threats and mitigation techniques that would allow achieving highly robust network slices.

Furthermore, a worth exploring aspect could be the integration of machine learning techniques into the resource allocation problem. The problem addressed in this work implicitly assumes that service requests are apriori known to the service provider. However, there could be scenarios where service request can rapidly evolve and being able to predict or estimate service demands can allow an operator to rapidly adapt resource allocation decisions.

Finally, another interesting course for future work would be implementing and testing the redundancy allocation heuristic scheme in a real NFV infrastructure. For example, the Placement Optimization engine [132], integrated with the OSM framework, could be enhanced with availability figures of computing nodes and used to determine an optimized VNF redundancy placement. This way, the performance of the proposed algorithms can be validated in a realistic scenario.

References

[1] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and issues", RFC 3234, February, Tech. Rep., 2002.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service", in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 13–24, ISBN: 9781450314190.

[3] K. Edeline and B. Donnet, "On a middlebox classification", in *IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)*, 2015.

[4] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, "The middlebox manifesto: Enabling innovation in middlebox deployment", in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X, Cambridge, Massachusetts: Association for Computing Machinery, 2011, ISBN: 9781450310598. DOI: 10.1145/2070562.2070583.

[5] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "Simple-fying middlebox policy enforcement using sdn", *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 27–38, Aug. 2013, ISSN: 0146-4833. DOI: 10.1145/2534169.2486022.

[6] J. M. Halpern and C. Pignataro, *Service Function Chaining (SFC) Architecture*, RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt, (accessed: 04.04.2021).

[7] ETSI, "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action", *White Paper*, no. 1, pp. 1–16, 2012. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper.pdf, (accessed: 04.04.2021).

[8] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De 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, 2016. DOI: 10.1109/COMST.2015.2477041.

[9] ETSI, "User Group; Quality of ICT services; Part 3: Template for Service Level Agreements (SLA)", 2015. [Online]. Available: https://www.etsi.org/deliver/etsi_eg/202000_202099/20200903/01.03.00_50/eg_20200903v010300m.pdf, (accessed: 04.04.2021).

[10] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004. DOI: 10.1109/TDSC.2004.2.

[11] ETSI, "Network Functions Virtualisation(NFV); Resiliency Requirements", *Group Specification, ETSI GS NFV-REL 001 V1.1.1*, 2015.

[12] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations", *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015. DOI: 10.1109/MCOM.2015.7045396.

[13] 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. DOI: 10.1109/MCOM.2017.1601201.

[14] D. Cotroneo, L. De Simone, and R. Natella, "NFV-Bench: A Dependability Benchmark for Network Function Virtualization Systems", *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 934–948, 2017. DOI: 10.1109/TNSM.2017.2733042.

[15] A. Avizienis, J.-C. Laprie, B. Randell, *et al.*, *Fundamental concepts of dependability*. University of Newcastle, Computing Science, 2001.

[16] M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein, "A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability", *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, pp. 106–124, 2009.

[17] ITU-T E.860 (06/02), *Framework of a service level agreement*, 2002.

[18] T. Trygar and G. Bain, "A framework for service level agreement management", in *MILCOM 2005 - 2005 IEEE Military Communications Conference*, Oct. 2005, 331–337 Vol. 1.

[19] H. Chantre and N. L. S. d. Fonseca, "Reliable broadcasting in 5g nfv-based networks", *IEEE Communications Magazine*, vol. 56, no. 3, pp. 218–224, 2018.

[20] ETSI, "Network Functions Virtualisation (NFV); Architectural Framework", *Group Specification, ETSI GS NFV 002 V1.2.1*, 2014.

[21] T. C. Bressoud and F. B. Schneider, "Hypervisor-based fault tolerance", *ACM Transactions on Computer Systems (TOCS)*, vol. 14, no. 1, pp. 80–107, 1996.

[22] Cheng Tan, Yubin Xia, Haibo Chen, and Binyu Zang, "Tinychecker: Transparent protection of vms against hypervisor failures with nested virtualization", in *IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*, Jun. 2012, pp. 1–6.

[23] M. Le and Y. Tamir, "ReHype: Enabling VM Survival across Hypervisor Failures", *SIGPLAN Not.*, vol. 46, no. 7, pp. 63–74, Mar. 2011, ISSN: 0362-1340.

[24] ETSI, "Network functions virtualisation (NFV); Use Cases", *Group Report, ETSI GR NFV 001 V1.2.1*, 2017.

[25] ——, "Network Functions Virtualisation (NFV); Network Operator Perspectives on NFV priorities for 5G", *White Paper*, no. 1, pp. 1–15, 2017.

[26] B. Blanco, J. O. Fajardo, I. Giannoulakis, E. Kafetzakis, S. Peng, J. Pérez-Romero, I. Trajkovska, P. S. Khodashenas, L. Goratti, M. Paolino, *et al.*, "Technology pillars in the architecture of future 5g mobile networks: Nfv, mec and sdn", *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[27] Amazon Web Services, *Amazon compute service level agreement*. [Online]. Available: `https://aws.amazon.com/it/compute/sla/`, (accessed: 04.04.2021).

[28] Google LLC, *Compute engine service level agreement (sla)*. [Online]. Available: `https://cloud.google.com/compute/sla`, (accessed: 04.04.2021).

[29] ETSI, *Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability*, 2016.

[30] ——, *Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities*, 2017.

[31] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey", *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.

[32] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "GREP: Guaranteeing reliability with enhanced protection in NFV", in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ACM, 2015, pp. 13–18.

[33] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Virtual network function placement for resilient service chain provisioning", in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, IEEE, 2016, pp. 245–252.

[34] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains", in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2016, pp. 128–133.

[35] M. Casazza, P. Fouilhoux, M. Bouet, and S. Secci, "Securing virtual network function placement with high availability guarantees", in *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, IEEE, 2017, pp. 1–9.

[36] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "Heading off correlated failures through independence-as-a-service", in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 317–334.

[37] G. Nencioni, B. E. Helvik, and P. E. Heegaard, "Including Failure Correlation in Availability Modeling of a Software-Defined Backbone Network", *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1032–1045, Dec. 2017, ISSN: 1932-4537.

[38] D. M. Nicol, W. H. Sanders, and K. S. Trivedi, "Model-based evaluation: from dependability to security", *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 48–65, Jan. 2004, ISSN: 2160-9209.

[39] P. R. Maciel *et al.*, "Dependability modeling", in *Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions*, IGI Global, 2012, pp. 53–97.

[40] K. S. Trivedi and A. Bobbio, *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge University Press, 2017.

[41] D. Bailey, E. Frank-Schultz, P. Lindeque, and J. L. Temple III, "Three reliability engineering techniques and their application to evaluating the availability of IT systems: An introduction", *IBM Systems Journal*, vol. 47, no. 4, pp. 577–589, 2008.

[42] M. Malhotra and K. S. Trivedi, "Power-hierarchy of dependability-model types", *IEEE Transactions on Reliability*, vol. 43, no. 3, pp. 493–502, 1994.

[43] A. Gonzalez, P. Gronsund, K. Mahmood, B. Helvik, P. Heegaard, and G. Nencioni, "Service Availability in the NFV Virtualized Evolved Packet Core", in *Global Communications Conference (GLOBECOM), 2015 IEEE*, IEEE, 2015, pp. 1–6.

[44] M. Di Mauro, M. Longo, F. Postiglione, and M. Tambasco, "Availability Modeling and Evaluation of a Network Service Deployed via NFV", in *International Tyrrhenian Workshop on Digital Communication*, Springer, 2017, pp. 31–44.

[45] M. Di Mauro, M. Longo, and F. Postiglione, "Availability evaluation of multi-tenant service function chaining infrastructures by multidimensional universal generating function", *IEEE Transactions on Services Computing*, 2018.

[46] R. Mijumbi, J. Serrat, J. Gorricho, S. Latre, M. Charalambides, and D. Lopez, "Management and orchestration challenges in network functions virtualization", *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016. DOI: 10.1109/MCOM.2016.7378433.

[47] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue, "Guaranteed-availability network function virtualization with network protection and vnf replication", in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–6.

[48] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning", *Networks*, vol. 70, no. 4, pp. 373–387, 2017.

[49] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 558–573, 2018.

[50] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups", in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*, IEEE, 2017, pp. 1–10.

[51] N.-T. Dinh and Y. Kim, "An efficient availability guaranteed deployment scheme for IoT service chains over fog-core cloud networks", *Sensors*, vol. 18, no. 11, p. 3970, 2018.

[52] F. Schardong, I. Nunes, and A. Schaeffer-Filho, "Nfv resource allocation: A systematic review and taxonomy of vnf forwarding graph embedding", *Computer Networks*, vol. 185, p. 107 726, 2021, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2020.107726.

[53] D. Li, P. Hong, K. Xue, and J. Pei, "Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy", *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[54] M. Casazza, M. Bouet, and S. Secci, "Availability-driven NFV orchestration", *Computer Networks*, vol. 155, pp. 47–61, 2019.

[55] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme", in *Communications (ICC), 2017 IEEE International Conference on*, IEEE, 2017, pp. 1–6.

[56] C. R. Kothari, *Research methodology: Methods and techniques*. New Age International, 2004.

[57] B. Yi, X. Wang, K. Li, M. Huang, *et al.*, "A comprehensive survey of network function virtualization", *Computer Networks*, vol. 133, pp. 212–262, 2018.

[58] M. Eder, "Hypervisor-vs. container-based virtualization", *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 1, 2016.

[59] ETSI, "Network functions virtualisation (NFV); Terminology for main concepts in NFV", *Group Specification, ETSI GS NFV 003 V1.2.1*, 2014.

[60] International Telecommunications Union – Telecommunications System Sector, "Terms and definitions related to quality of service and network performance including dependability", *Recommendation*, vol. 800, 1994.

[61] International Telecommunications Union – Telecommunications Standartization Sector, "Y1540: Internet protocol data communication service-IP packet transfer and availability performance parameters", *Geneva: ITU-T*, 2002.

[62] P. Cholda, A. Mykkeltveit, B. E. Helvik, O. J. Wittner, and A. Jajszczyk, "A survey of resilience differentiation frameworks in communication networks", *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, pp. 32–55, 2007.

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

[64] R. E. Barlow and F. Proschan, "Statistical theory of reliability and life testing: probability models", Florida State Univ Tallahassee, Tech. Rep., 1975.

[65] B. S. Dhillon, "Engineering reliability: new techniques and applications", Tech. Rep., 1981.

[66] K. S. Trivedi, *Probability and statistics with reliability, queuing, and computer science applications*. Wiley Online Library, 1982, vol. 13.

[67] C. J. Colbourn, *The combinatorics of network reliability*. Oxford University Press, Inc., 1987.

[68] R. Sahner, K. S. Trivedi, and A. Puliafito, "Hierarchical models", in *Performance and Reliability Analysis of Computer Systems*, Springer, 1996, pp. 261–311.

[69] H. Sukhwani, A. Bobbio, and K. S. Trivedi, "Largeness avoidance in availability modeling using hierarchical and fixed-point iterative techniques", *International Journal of Performability Engineering*, vol. 11, no. 4, pp. 305–319, 2015.

[70] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application", in *International Workshop on Timed Petri Nets*, USA: IEEE Computer Society, 1985, pp. 106–115, ISBN: 0818606746.

[71] *Möbius: Model-based environment for validation of system reliability, availability, security and performance*, "`https://www.mobius.illinois.edu`", (accessed: 04.04.2021).

[72] A. Movaghar, "Performability Modeling with Stochastic Activity Networks", AAI8520952, PhD thesis, USA, 1985.

[73] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts", in *Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science Bergen Dal, The Netherlands, July 3–7, 2000 Revised Lectures*, E. Brinksma, H. Hermanns, and J.-P. Katoen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 315–343, ISBN: 978-3-540-44667-5. DOI: `10.1007/3-540-44667-2_9`.

[74] W. H. Sanders and J. F. Meyer, "METASAN: A Performability Evaluation Tool Based on Stochastic Activity Networks", in *Proceedings of 1986 ACM Fall Joint Computer Conference*, ser. ACM '86, Dallas, Texas, USA: IEEE Computer Society Press, 1986, pp. 807–816, ISBN: 0818647434.

[75] W. H. Sanders, W. D. Obal II, M. A. Qureshi, and F. Widjanarko, "The UltraSAN modeling environment", *Performance Evaluation*, vol. 24, no. 1-2, pp. 89–115, 1995.

[76] D. Daly, D. D. Deavours, J. M. Doyle, P. G. Webster, and W. H. Sanders, "Möbius: An extensible tool for performance and dependability modeling", in *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Springer, 2000, pp. 332–336.

[77] S. Gilmore, J. Hillston, L. Kloul, and M. Ribaudo, "PEPA nets: a structured performance modelling formalism", *Performance Evaluation*, vol. 54, no. 2, pp. 79–104, 2003.

[78] M. D. Ford, K. Keefe, E. LeMay, W. H. Sanders, and C. Muehrcke, "Implementing the ADVISE security modeling formalism in Möbius", in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2013, pp. 1–8.

[79] D. D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derisavi, J. M. Doyle, W. H. Sanders, and P. G. Webster, "The Mobius framework and its implementation", *IEEE Transactions on Software Engineering*, vol. 28, no. 10, pp. 956–969, 2002.

[80] T. Courtney, S. Gaonkar, K. Keefe, E. W. Rozier, and W. H. Sanders, "Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models", in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, IEEE, 2009, pp. 353–358.

[81] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability and performability", in *Dependable computing for critical applications*, Springer, 1991, pp. 215–237.

[82] G. Ciardo and K. S. Trivedi, "A decomposition approach for stochastic reward net models", *Performance Evaluation*, vol. 18, no. 1, pp. 37–59, 1993, Analysis of Petri Net Performance Models, ISSN: 0166-5316. DOI: https://doi.org/10.1016/0166-5316(93)90026-Q.

[83] N. Milanovic, "Models, Methods and Tools for Availability Assessment of IT-Services and Business Processes", PhD thesis, habilitationsschrift, 2010.

[84] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method", *Computer Methods in Applied Mechanics and Engineering*, vol. 71, no. 2, pp. 197–224, 1988, ISSN: 0045-7825.

[85] P. M. Frank and M. Eslami, "Introduction to system sensitivity theory", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, no. 6, pp. 337–338, 1980.

[86] D. M. Hamby, "A review of techniques for parameter sensitivity analysis of environmental models", *Environmental monitoring and assessment*, vol. 32, no. 2, pp. 135–154, 1994.

[87] J. T. Blake, A. L. Reibman, and K. S. Trivedi, "Sensitivity analysis of reliability and performability measures for multiprocessor systems", in *Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, 1988, pp. 177–186.

[88] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system", in *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, IEEE, 2009, pp. 365–371.

[89] R. d. S. Matos, P. R. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, "Sensitivity analysis of server virtualized system availability", *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012.

[90] R. L. Iman and J. C. Helton, "An investigation of uncertainty and sensitivity analysis techniques for computer models", *Risk analysis*, vol. 8, no. 1, pp. 71–90, 1988.

[91] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation", *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.

[92] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions", in *10th International Conference on Network and Service Management (CNSM) and Workshop*, IEEE, 2014, pp. 418–423.

[93] M. T. Beck and J. F. Botero, "Coordinated allocation of service function chains", in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7417401.

[94] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in NFV deployments", in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[95] A. Fischer, J. F. Botero, M. T. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey", *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013.

[96] M. Gao, B. Addis, M. Bouet, and S. Secci, "Optimal orchestration of virtual network functions", *Computer Networks*, vol. 142, pp. 108–127, 2018, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.06.006.

[97] Z. Drezner and H. W. Hamacher, *Facility location: applications and theory*. Springer Science & Business Media, 2001.

[98] A. Gupta, M. F. Habib, P. Chowdhury, M. Tornatore, and B. Mukherjee, "On service chaining using virtual network functions in network-enabled cloud systems", in *2015 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, IEEE, 2015, pp. 1–3.

[99] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions", in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2015, pp. 1–9.

[100] J. Cao, Y. Zhang, W. An, X. Chen, Y. Han, and J. Sun, "VNF placement in hybrid NFV environment: Modeling and genetic algorithms", in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, IEEE, 2016, pp. 769–777.

[101] L. Ruiz, R. J. Durán, I. De Miguel, P. S. Khodashenas, J.-J. Pedreno-Manresa, N. Merayo, J. C. Aguado, P. Pavon-Marino, S. Siddiqui, J. Mata, *et al.*, "A genetic algorithm for VNF provisioning in NFV-Enabled Cloud/MEC RAN architectures", *Applied Sciences*, vol. 8, no. 12, p. 2614, 2018.

[102] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-Aware Backup Allocation For A Chain of Virtual Network Functions", in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, IEEE, 2019, pp. 1918–1926.

[103] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains", in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, IEEE, 2017, pp. 1–9.

[104] M. Di Mauro, M. Longo, F. Postiglione, G. Carullo, and M. Tambasco, "Service function chaining deployed in an NFV environment: An availability modeling", in *Standards for Communications and Networking (CSCN), 2017 IEEE Conference on*, IEEE, 2017, pp. 42–47.

[105] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "IP multimedia subsystem in an NFV environment: Availability evaluation and sensitivity analysis", in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, IEEE, 2018, pp. 1–6.

[106] ——, "IP Multimedia Subsystem in a containerized environment: availability and sensitivity evaluation", in *2019 IEEE Conference on Network Softwarization (NetSoft)*, IEEE, 2019, pp. 42–47.

[107] E. Guedes and P. Maciel, "Stochastic Model for Availability Analysis of Service Function Chains using Rejuvenation and Live Migration", in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, IEEE, 2019, pp. 211–217.

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

[109] N. F. S. de Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey", *Computer Communications*, 2019.

[110] T. Soenen, W. Tavernier, D. Colle, and M. Pickavet, "Optimising microservice-based reliable NFV management & orchestration architectures", in *Resilient Networks Design and Modeling (RNDM), 2017 9th International Workshop on*, IEEE, 2017, pp. 1–7.

[111] M. R. Rahman and R. Boutaba, "SVNE: Survivable virtual network embedding algorithms for network virtualization", *IEEE Transactions on Network and Service Management*, vol. 10, no. 2, pp. 105–118, 2013.

[112] C. Prodhon and C. Prins, "A survey of recent research on location-routing problems", *European Journal of Operational Research*, vol. 238, no. 1, pp. 1–17, 2014.

[113] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A Reliability-Aware Network Service Chain Provisioning With Delay Guarantees in NFV-Enabled Enterprise Datacenter Networks", *IEEE Transaction on Network and Service Management*, vol. 14, no. 3, 2017.

[114] J. Kang, O. Simeone, and J. Kang, "On the trade-off between computational load and reliability for network function virtualization", *IEEE Communications Letters*, vol. 21, no. 8, pp. 1767–1770, 2017.

[115] A. Tomassilli, N. Huin, F. Giroire, and B. Jaumard, "Resource requirements for reliable service function chaining", in *2018 IEEE International Conference on Communications (ICC)*, IEEE, 2018, pp. 1–7.

[116] J. Fan, M. Jiang, O. Rottenstreich, Y. Zhao, T. Guan, R. Ramesh, S. Das, and C. Qiao, "A framework for provisioning availability of NFV in data center networks", *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2246–2259, 2018.

[117] Y. Alahmad and A. Agarwal, "VNF Placement Strategy for Availability and Reliability of Network Services in NFV", in *2019 Sixth International Conference on Software Defined Systems (SDS)*, IEEE, 2019, pp. 284–289.

[118] N.-T. Dinh and Y. Kim, "An efficient reliability guaranteed deployment scheme for service function chains", *IEEE Access*, vol. 7, pp. 46 491–46 505, 2019.

[119] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements", in *2015 IEEE Globecom Workshops (GC Wkshps)*, IEEE, 2015, pp. 1–7.

[120] S. Bian, X. Huang, Z. Shao, X. Gao, and Y. Yang, "Service chain composition with failures in NFV systems: A game-theoretic perspective", in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6.

[121] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers", in *SIGCOMM*, 2008, pp. 75–86.

[122] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers", in *SIGCOMM*, 2009, pp. 63–74.

[123] S. Ramamurthy and B. Mukherjee, "Survivable wdm mesh networks. part i-protection", in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 2, 1999, 744–751 vol.2.

[124] G. Li, D. Wang, C. Kalmanek, and R. Doverspike, "Efficient distributed path selection for shared restoration connections", in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, IEEE, vol. 1, 2002, pp. 140–149.

[125] ——, "Efficient distributed restoration path selection for shared mesh restoration", *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 761–771, 2003.

[126] *OSM website*. [Online]. Available: `https://osm.etsi.org`, (accessed: 04.04.2021).

[127] *OpenBaton website*. [Online]. Available: `https://openbaton.github.io`, (accessed: 04.04.2021).

[128] G. Nencioni, R. G. Garroppo, A. J. Gonzalez, B. E. Helvik, and G. Procissi, "Orchestration and control in software-defined 5G networks: Research challenges", *Wireless communications and mobile computing*, vol. 2018, 2018.

[129] M. Rausand, A. Barros, and A. Hoyland, *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, 2020.

[130] A. J. Gonzalez, B. E. Helvik, J. K. Hellan, and P. Kuusela, "Analysis of Dependencies between Failures in the UNINETT IP Backbone Network", in *2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing*, 2010, pp. 149–156.

[131] A. J. Gonzalez, J. Ordonez-Lucena, B. E. Helvik, G. Nencioni, M. Xie, D. R. Lopez, and P. Grønsund, "The Isolation Concept in the 5G Network Slicing", in *2020 European Conference on Networks and Communications (EuCNC)*, 2020, pp. 12–16.

[132] *OSM PoC 5 - Placement of Workloads in Distributed Cloud Networks*. [Online]. Available: `https://osm.etsi.org/wikipub/images/c/c9/OSM_PoC_5_Report.pdf`, (accessed: 04.04.2021).

Part II

Included Papers

PAPER A

Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes

Besmir Tola, Gianfranco Nencioni, Bjarne E. Helvik, and Yuming Jiang

IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN)

Coimbra, Portugal, 2019

2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)

Modeling and Evaluating NFV-Enabled Network Services under Different Availability Modes

Besmir Tola*, Gianfranco Nencioni†, Bjarne E. Helvik*, and Yuming Jiang*
*NTNU-Norwegian University of Science and Technology, Norway
†University of Stavanger, Norway
Email:*{besmir.tola, bjarne.e.helvik, yuming.jiang}@ntnu.no, †gianfranco.nencioni@uis.no

***Abstract*—Network and Telecom operators are continuously embracing the adoption of Network Function Virtualization (NFV) as a means to provide more agile, flexible and cost-efficient services. Many telecommunication services need to possess carrier-grade quality of service; therefore, future NFV-enabled telecom services should present high levels of availability. In this paper, we present a composed availability model of NFV-enabled network services under different availability modes, namely Standard Availability, Cold Protection, and Hot Protection. We model and analyze the availability of NFV-enabled network services for each of the availability modes aiming at finding the best redundancy configuration to ensure carrier-grade quality. Through discrete-event simulation analysis we are able to identify the most suitable redundancy configuration for each of the availability modes.**

***Index Terms*—NFV, Service Function Chaining, Availability Modes, Cold Protection, Hot Protection.**

I. Introduction

Network Function Virtualization (NFV) is expected to change the way operators provide their services by entailing greater network programmability, dynamic service delivery, and service automation. Through decoupling network functions into software and hardware, NFV aims at replacing legacy network functions with virtualized instances, called Virtual Network Functions (VNFs) [1], running as software into commodity servers. By linking together many VNFs, NFV provides the ability to define specialized services as an ordered set of network functions (e.g., firewalls, intrusion protection etc.), commonly referred to as Service Function Chain (SFC).

The VNFs are network function software implementations running over an NFV infrastructure (NFVI), which provides, through a virtualisation layer commonly referred to as Virtual Machine Monitor (VMM) or hypervisor, the virtual resources needed to support the execution of VNFs. The management and orchestration of resources and services is performed by the NFV-Management and Orchestration (NFV-MANO), which represents a logically central entity in charge of service lifecycle operations. The NFV-MANO is composed of three main components: Virtual Infrastructure Manager (VIM), VNF Manager (VNFM), and NFV Orchestrator (NFVO).

The transition to NFV deployments introduces additional resilience challenges which may threaten the benefits that NFV architectures embrace [2]. In addition, NFV-enabled telecommunication services are expected to fulfill very strict carrier-grade availability requirements, i.e., five-nines or more [3]. As a result, NFV resilience challenges have drained significant attention from both academia and industry research. To this end, ETSI has provided several guidelines regarding reliability concepts and requirements [4] (and the references within).

Server virtualization represents the core enabling technology for NFV. The authors of [5] paved the way of availability modelling involving virtualized systems with multiple failure modes. Using fault-tree analysis and continuous-time Markov chains (CTMC), they perform a sensitivity analysis for the system performability, i.e., performance and reliability, and extend the analysis for different scalability considerations in [6], [7]. Zhang *et al.* [8] and Dantas *et al.* [9] use a combination of CTMC and Reliability Block Diagram (RBD) approaches to represent and evaluate the dependability of virtualized systems and cloud computing infrastructure, respectively.

An availability model of a virtualized Evolved Packet Core is presented in [10]. Using Stochastic Activity Networks (SANs), the authors assess the system availability in case of multiple and catastrophic failure events since similar events may seriously impact the system availability. In [11], the authors propose a two-level model and evaluate the availability of an SFC deployed in an NFV architecture. By merging RBDs and Stochastic Reward Nets (SRNs) they perform a sensitivity analysis to identify critical parameters. Similarly, in [12], they extend the analysis by including the VIM functionality.

In this paper, we propose an availability model which distinctively to the previous works considers multiple availability modes featuring different fault recovery mechanisms. The considered availability modes include Standard Availability (SA), Cold Protection (CP), and Hot Protection (HP), where each mode can be suitable for different service-level availability requirements. Furthermore, we investigate the impact of redundancy configuration and protection schemes on ensuring a carrier-grade level of service dependability. The availability model is implemented by using two formalisms: i) Replicate/Join, a state sharing composition model that captures the dependencies among components, and ii) the Stochastic Activity Networks (SAN), suitable for describing the failure dynamics of the individual components.

The paper is structured as follows. Section II illustrates the proposed service availability model. Section III presents the salient features of the different availability modes. The SAN models of the individual components are presented in Section IV. Numerical results of the simulation analysis for each of the availability modes are presented in Section V.

978-1-5386-8461-0/19/$31.00 ©2019 IEEE

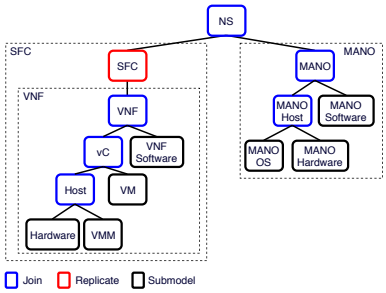


Fig. 1. Network Service SAN model using Replicate/Join formalism.

Finally, Section VI concludes the paper by highlighting the most important insights.

II. Availability Model

In this section, the composed model used to evaluate service availability is presented. The model is implemented through a Replicate/Join formalism by using the Möbius software tool [13]. The formalism enables the modeler to compose a model in the form of a tree, where each leaf node represents a system submodel and each non-leaf node can be a Join or Replicate node. A Join node is a state-sharing node used to compose two or more submodels, whereas a Replicate node is used to compose submodel replicas.

The delivery of an NFV-enabled network service results from the interaction of the SFC (as an ordered sequence of VNFs composing the service) and the MANO (which deploys, instantiates and manages the service lifecycle). While it is argued that a MANO failure shall not affect existing VNFs [14], as specified in [4] and highlighted by the authors of [15], the MANO actually plays a critical role in ensuring the VNF's resiliency. Aligned with [15], we consider the service is available when both SFC and MANO are available.

Fig. 1 depicts the composed service model. We assume a VNF is deployed through a hypervisor-based virtualization running directly on hardware, i.e., bare-metal virtualisation. In addition, we assume a Virtual Machine (VM) is dedicated to a single VNF. Therefore, the model is composed of the *Host* subsystem which symbolizes an NFVI server consisting of the computing, storage and network hardware resources. This level joins two submodels representing the *Hardware* and *VMM* components. The intermediate level represents the virtual Container (*vC*) providing the virtualized environment where a VNF is executed by joining the *VM submodel* with the *Host* level. Lastly, the *VNF* level joins the *vC* and the *VNF software* submodels.

A high-level architecture of a widely referenced solution, namely Open Baton [16], is used as a reference for the MANO model. A common deployment involves a commodity server running its own OS, e.g., Linux-based kernel OS, and the installation of the various MANO software component's packages, e.g., NFVO, VNFM etc. For simplicity, we consider the MANO software as a single component where the failure of any of its software packages causes a failure of the MANO functionality. Therefore, on the *Host* level, the MANO model is composed by joining the *MANO Hardware* with the *MANO OS*. On the higher level, the *MANO software* is joined with the *MANO Host* node. When any of the elements fails, the MANO becomes unavailable.

The SFC consists of an ordered sequence of VNFs. Therefore by replicating the same VNF non-leaf node, through the SFC replicate node, we obtain a representative model of a SFC where the number of replicas indicate the number of VNFs composing the chain. The SFC, being a replicate node, allows state-sharing among the different replicas. We assume that each replica, i.e., VNF, fails independently. Thus, by not sharing any state among the VNFs, we simulate such independence. By joining the SFC and the MANO subsystems, i.e., the top join node, the model represents a series configuration where each subsystem (MANO, VNF_1, VNF_2,..., VNF_O) needs to be working in order for the service to be available.

From a modeling perspective, there are similarities among the submodels composing the VNF model and the MANO model. Specifically, the same submodel, with related failure and repair parameters, is used to describe the failure dynamics of both the *VNF* and the *MANO software* components. The same submodel is used for the *VNF* and *MANO hardware* components, and so is the submodel used for the VNF *VMM* and *MANO OS* components.

Each component's behavior dynamics are captured through a specific SAN submodel which we introduce in more detail in Section IV.

III. Availability Modes

The VNF availability modes we investigate are Standard Availability (SA), Cold Protection (CP) and Hot Protection (HP). The former one is regarded as a baseline mode since it features the simplest recovery procedure. Whereas, the later ones, driven from typical implementations using virtualization technologies (see for example [17]), embody more sophisticated recovery strategies.

In this paper, we consider that each VNF composing the SFC is deployed as a load-sharing cluster where several VNF units, making up the cluster, are needed to satisfy a certain load demand. The VNF is considered to be operational if at least N out of the K units are working. Therefore, the cluster itself is able to provide protection for up to $K - N$ simultaneous failures. On top of load-sharing we consider an additional level of protection through our availability modes where M redundant units provide protection to the load-sharing cluster. As a result, by tuning the K, N and M parameters we investigate different redundancy configurations.

Similarly, for the MANO is implemented as a load-sharing cluster where R defines the number of MANO units and the MANO is operational if at least S out of the R units are up.

A. Standard Availability (SA)

The SA mode represents a case where a VNF does not rely on any redundancy mechanism. Failures on the different levels, which are discussed in Section II and shown in Fig. 1,

are detected through heartbeat mechanisms. Once a failure on the host level is detected, the recovery process requires the summoning of an operator to execute a manual replacing or repairing of the failed component. Whereas, in case a failure on a software level is detected, i.e., VMM, VM or VNF software, the recovery follows a two-step procedure. At first, an automatic restart/reboot of the failed component is triggered by the MANO and only if the component restart/reboot does not recover the service, a hard repair, i.e., patch fixing or software updating, is performed.

B. Cold Protection (CP)

The CP mode consists of a solution where the aim is to minimize the downtime caused by a failure on the host level.The CP mode leverages multiple hosts configured as a cluster. Specifically, for a primary host, there is a secondary host ready to takeover the VMs affected by a primary-host failure. A primary host sees the secondary one by exchanging heartbeat messages. In case of failures within the host level, i.e., hardware or VMM, the CP mode features an automatic restart of the affected VMs, activated by the MANO, by performing a similar to "live migration" procedure, on the secondary host. In case the failure is experienced within the VM/VNF software level, the MANO restarts the affected VM on the same host. Similar to the SA mode, in case a VM/VNF software restart does not successfully recover the service, a hard repair is executed. Note that the redundancy is provided only on the host level and the redundancy restoration is performed by either replacing/repairing the failed hardware component or by performing a soft repair followed by an eventual hard repair of the VMM in case the former does not restore the redundancy.

C. Hot Protection (HP)

Hypervisor-based Fault Tolerance represents a powerful technology promising continuous service availability [17]. Similarly to this solution, in the HP-mode implementation a VM, i.e., primary VM, is protected by creating and synchronizing a secondary VM, that is identical and continuously available in a different host. The secondary VM is ready to take over in the event of a failure caused in the host level, i.e, hardware and VMM, VM or VNF application level. In this mode, the failure detection uses a combination of heartbeat messages and logging traffic to monitor the status of the primary VM. In case the logging traffic and/or heartbeat miss or exceed a specific timeout interval (order of seconds), a failure is detected. Once the failure is detected, an automatic and seamless failover to the secondary VM is performed. The redundancy restoration is carried out similarly to failure recovery in SA. When the hardware fails, a manual repair is preformed. In case the VMM, VM or the VNF software fails, the same two-step procedure of SA and CP is performed.

Driven by the fact that HP provides a VM fault-tolerant solution that promises service continuity, we consider in the remaining that the MANO adopts only the HP mode.

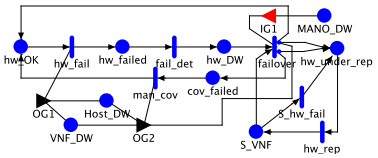


Fig. 2. Hardware SAN availability models.

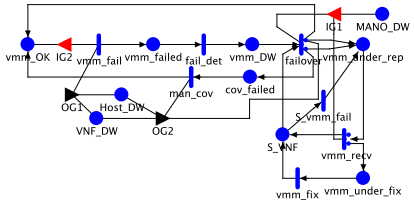


Fig. 3. VMM SAN availability models.

IV. SAN Submodels

In this section, the SAN models of the elements, composing the service, for each of the availability modes are illustrated. A SAN model is composed of *places, activities, input gates*, and *output gates* primitives. Through activity firings and following specific distributions, tokens are moved among places resulting in system state changes. Input and output gates enable and control activity firings.

The availability modes differ from each other only on the recovery mechanisms. In particular, the HP mode includes all the SAN primitives utilized in the SA and CP modes. Therefore, due to space limitations we illustrate only the HP mode since the two others may be induced from the HP mode. Note that the MANO submodels are identical to the VNF submodels as specified at the end of Section II hence, we avoid illustrating.

A. Hardware Submodel

The *hardware* SAN availability model is depicted in Fig. 2. The model comprises the following shared places, i.e., states shared among the different *hardware, VMM, VM and VNF software* submodels:

- *VNF_DW* indicate the number of failed VNF units;
- *Host_DW* represents the number of hosts that are down;
- *MANO_DW* represents the status of the MANO. In case more than $R-S$ tokens are present, the MANO is down;
- *S_VNF* is populated with M tokens and represents the secondary VNF redundant units ready to takeover the service from the failed VNFs;

In addition, the following output gates enable token marking movements for the shared places:

- *IG1* enables the failover operation activity. Only in case there are less than $R-S$ tokens in *MANO_DW*, i.e., the MANO cluster is operational, the failover is performed;
- *OG1/OG2*, when the *hw_fail/hw_rep* timed activity is completed, the output gate increases/decreases with 1 token the places *Host_DW* and *VNF_DW*;

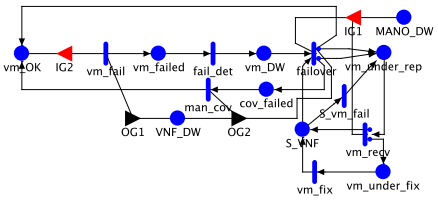


Fig. 4. VM SAN availability model.

The following places define the component operational status:

- *hw_OK* corresponds to the fully working state of the hardware components and is initialized with K tokens;
- *hw_failed* is populated with 1 token in case a hardware component fails, 0 otherwise;
- *hw_DW* represents the detection of a hardware failure;
- *hw_under_rep* represents the number of hardware components undergoing a repair process;
- *cov_failed* defines the state where the failover procedure fails and a manual coverage is required;

The places are connected by mean of the following negative exponentially distributed (n.e.d.) timed activities:

- *hw_fail* and *hw_rep* represent the hardware failure and repair events with rates λ_{hw} and μ_{hw}, respectively;
- *fail_det* represents the failure detection with rate μ_{det};
- *failover* represents the HP failover event with rate μ_{fo}. Since the failover is an automatic procedure, the MANO triggers the recovery procedure. There are two cases, with probability C_{fo} the failover is successful and 1 token is moved into *hw_under_rep* and another token is fetched from *S_VNF* and is moved into *hw_OK*. Whereas, with probability $1 - C_{fo}$ the failover procedure fails and 1 token is placed into *hw_under_rep* and the previous one fetched from spare units is put into *cov_failed*;
- *S_hw_fail* represent the hardware failure event of the redundant host with rate λ_{hw}. The redundant host provides resources to other services as well; therefore, they experience hardware failures similarly to the primary;
- *man_cov* represents the intervention of an operator performing a manual coverage with rate μ_{cov};

B. VMM Submodel

Fig. 3 illustrates the *VMM* SAN availability submodel. Compared to the *Hardware* model, the difference lies on the redundancy restoration identified by the *vmm_recv* timed activity. With probability C_{res}, a VMM restart recovers the service and with probability $1 - C_{res}$ the VMM undergoes a manual fixing. Due to space constraints, we omit further description.

C. VM and VNF Submodels

Fig. 4 illustrates the *VM* submodel. Although apparently similar to the *VMM*, the submodel slightly differs on the fact that the *VM submodel* is an element of a higher level, i.e., vC. Thus, VMs can fail only if their underlying hosts have not failed. To this end, *IG2* enables a VM failure only if the number of tokens in *VNF_DW* are less than K. Similarly, the *VNF software* submodel belongs to the higher level and the relative SAN model is identical to the *VM* model.

TABLE I
MODEL PARAMETERS USED IN THE EVALUATION.

Parameter	Time	Description [mean time to]
$1/\lambda_{hw} = 6.5$	months	next hardware failure
$1/\mu_{hw} = 1$	hour	hardware repair
$1/\mu_{fo} = 5$	secs	VM failover
$1/\mu_{det} = 5$	secs	failure detection
$C_{fo} = 0.95$		VM failover coverage factor
$1/\mu_{mig} = 1$	minute	VM migrate
$C_{mig} = 0.95$		VM migrate coverage factor
$1/\lambda_{vmm} = 4$	months	next VMM failure
$1/\mu_{vmm} = 1$	hour	VMM fix
$1/\mu_{vmm_{res}} = 30$	secs	VMM reset
$1/\lambda_{vm} = 2$	months	next VM failure
$1/\mu_{vm} = 1$	hour	VM hard fix
$1/\mu_{vm_{res}} = 30$	secs	VM reset
$1/\lambda_{sw} = 2$	weeks	next VNF software failure
$1/\mu_{sw} = 1$	hour	VNF software fix
$1/\mu_{sw_{res}} = 15$	secs	VNF software restart
$C_{res} = 0.8$		restart coverage factor
$1/\mu_{\Delta} = 30$	minutes	summon an operator
$1/\lambda_{Msw} = 1$	month	next MANO software failure
$1/\mu_{Msw} = 1$	hour	mean time to MANO software fix
$1/\mu_{MSW_{res}} = 15$	secs	MANO software restart
$1/\lambda_{OS} = 1$	month	next OS failure
$1/\mu_{OS} = 1$	hour	OS fix
$1/\mu_{OS_{res}} = 1$	minute	OS reboot
$1/\mu_{cov} = 30$	minutes	manual coverage
$O = 3$		# VNFs composing the SFC

V. NUMERICAL EVALUATION

In this section we evaluate the different VNF cluster configurations for each of the availability modes. We compute the steady-state service availability for the composed model using discrete-time simulations implemented in Möbius with 95% confidence interval for a one year time simulation. The set of numerical values regarding failure, repair intensities and coverage probabilities, retrieved from previous literature [6], [7], [11], [12], are presented in Table I.

Cluster overprovisioning is an excellent means for providing high level of protection, i.e., providing extra units to cope with multiple simultaneous failures. For this purpose, we define the VNF load-sharing cluster *overprovisioning-ratio* as $\gamma = \frac{K-N}{N}$. We assume that each VNF cluster is composed of $K = 4$ units and vary N so that γ is increased from 0 to 0.25 and 0.5. The same definition and assumption apply to the MANO cluster as well with $\gamma_M = \frac{R-S}{S}$ and $R = 4$.

Table II illustrates the service availability with varying number of redundant units M, overprovisioning-ratio γ and recovery coverage factors for the modes that make use of redundancy, i.e., CP and HP. We observe that for an increasing M there is an almost negligible availability increase irrespective of the availability mode. On the other hand, an increase of the overprovisioning-ratio is associated with up to three orders of magnitude of availability increase hence, suggesting that it is much more beneficial to scale-out a cluster than to provide the same unit(s) in the form of redundant backups. Furthermore, we notice that the HP mode is more sensitive

TABLE II
AVAILABILITY FOR DIFFERENT VNF REDUNDANCY CONFIGURATIONS AND RECOVERY COVERAGE FACTOR.

		Cold Protection		Hot Protection	
γ	M	$C_{mig} = 0.8$	$C_{mig} = 0.99$	$C_{fo} = 0.8$	$C_{fo} = 0.99$
0	1	99.25%	99.30%	99.59%	99.96%
	2	99.26%	99.31%	99.60%	99.98%
	3	99.27%	99.45%	99.61%	99.99%
0.25	1	99.9964%	99.9970%	99.9981%	99.99971%
	2	99.9967%	99.9976%	99.9992%	99.99992%
	3	99.9968%	99.9985%	99.9994%	99.99997%

For all the results $\gamma_M = 0.25$.

TABLE III
EFFECTS OF VNF CLUSTER OVERPROVISIONING ON SERVICE AVAILABILITY FOR DIFFERENT FAILURE INTENSITIES.

Failure Intensities	γ	Standard Availability	Cold Protection	Hot Protection
λ_{ref}	0	98.9%	99.30%	99.88%
	0.25	99.994%	99.997%	99.9997%
	0.5	99.999941%	99.99997%	99.999993%
$10 \cdot \lambda_{ref}$	0	90.35%	93.18%	97.59%
	0.25	99.47%	99.71%	99.80%
	0.5	99.91%	99.98%	99.99%

For all the results $M = 1$ and $\gamma_M = 0.25$.

to coverage factor variations compared to the CP mode. Increasing the robustness of the failover mechanism, i.e., higher coverage, may generate up to one order of magnitude higher availability. The explanation lies within the mode itself since the CP mode exploits a VM migration only for hardware and VMM failure events, whereas the HP mode fully exploits the failover procedure for all kinds of failures.

Table III shows the service availability for each mode when the provisioning ratio is varied. Two cases are considered, one with failure intensities taken from Table I, denoted with λ_{ref}, and the case where failure intensities are $10 \cdot \lambda_{ref}$. We notice that in the former case, only the HP mode achieves a carrier-grade quality (5 nines) when each VNF cluster is overprovisioned with one additional VNF unit. By providing two extra units as the means for protection, all the modes achieve more than 5 nines. On the other hand, for higher failure intensities, none of the modes reaches 5 nines availability.

With respect to the MANO provisioning ratio, Table IV illustrates the results when varying γ_M. We observe that the availability is augmented by one nine when the provisioning ratio is increased from 0 to 0.25, but remains almost unchanged when the ratio becomes higher. Therefore, while overprovisioning of the MANO cluster provides protection to the service, a high overprovisioning does not gain accordingly on the service availability.

VI. CONCLUDING REMARKS

In this paper, an availability model based on SAN composition has been proposed. The model is flexible and can be extended to incorporate even more failure types on both hardware (memory, disk, CPU) and VNF (VNF components) level. A sensitivity analysis aiming at identifying the configuration

TABLE IV
EFFECTS OF MANO CLUSTER OVERPROVISIONING ON SERVICE AVAILABILITY.

γ_M	Standard Availability	Cold Protection	Hot Protection
0	99.97%	99.97%	99.98%
0.25	99.99425%	99.9970%	99.999731%
0.5	99.99428%	99.9971%	99.999732%

For all the results $M = 1$ and $\gamma = 0.25$.

that achieves the so-called "fine-nines" availability has been carried out. Three different protection mechanisms have been investigated and the outcomes show that service availability is sensitive to a correct dimensioning of the VNF and MANO clusters. Increasing the VNF cluster size by one unit coincides with an increase of up to three orders of magnitude of the service availability but a high MANO overprovisioning does not bring a substantial advantage. Moreover, when a Hot Protection mode is configured, the failover robustness, i.e., higher coverage factor, can be exploited to achieve up to one order of magnitude availability boost.

ACKNOWLEDGMENT

This research was funded by the joint EU FP7 Marie Curie Actions Cleansky Project, Contract No. 607584.

REFERENCES

[1] G. N. ETSI, "ETSI GS NFV 002 v1.2.1: Network Functions Virtualisation (NFV); Architectural Framework," 2014.
[2] 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.
[3] R. Swale and D. Collins, *Carrier Grade Voice Over IP*. McGraw Hill Professional, 2013.
[4] I. N. ETSI, "ETSI GR NFV-REL 007 v1.1.2: Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities," 2017.
[5] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *PRDC'09*. IEEE.
[6] R. d. S. Matos, P. R. Maciel, F. Machida, D. S. Kim, and K. S. Trivedi, "Sensitivity analysis of server virtualized system availability," *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012.
[7] D. S. Kim *et al.*, "Availability modeling and analysis of a virtualized system using stochastic reward nets," in *CIT'16*. IEEE.
[8] X. Zhang, C. Lin, and X. Kong, "Model-driven dependability analysis of virtualization systems," in *ICIS'09*. IEEE, 2009, pp. 199–204.
[9] J. Dantas, R. Matos, J. Araujo, and P. Maciel, "An availability model for eucalyptus platform: An analysis of warm-standy replication mechanism," in *SMC'12*. IEEE, 2012, pp. 1664–1669.
[10] A. Gonzalez *et al.*, "Service availability in the NFV virtualized evolved packet core," in *GLOBECOM, 2015 IEEE*. IEEE.
[11] M. Di Mauro *et al.*, "Service function chaining deployed in an NFV environment: An availability modeling," in *CSCN'17*. IEEE.
[12] ——, "Availability modeling and evaluation of a network service deployed via NFV," in *TWDC'17*. Springer, pp. 31–44.
[13] Möbius: Model-based environment for validation of system reliability, availability, security and performance. [Online]. Available: www.mobius.illinois.edu
[14] I. N. ETSI, "ETSI GS NFV-REL 001 v1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.
[15] A. J. Gonzalez, G. Nencioni, A. Kamisiński, B. E. Helvik, and P. E. Heegaard, "Dependability of the NFV orchestrator: State of the art and research challenges," *IEEE Communications Surveys & Tutorials*, 2018.
[16] Open Baton: An open source reference implementation of the ETSI NFV MANO specification. [Online]. Available: openbaton.github.io
[17] VMware vSphere Availability. Accessed 2019 Jan. [Online]. Available: docs.vmware.com/en/VMware-vSphere/6.5/vsphere-esxi-vcenter-server-65-availability-guide.pdf

PAPER B

On the Resilience of the NFV-MANO: An Availability Model of a Cloud-native Architecture

Besmir Tola, Yuming Jiang, and Bjarne E. Helvik

IEEE Proceeding of 16th International Conference on the Design of Reliable Communication Networks (DRCN)

Milano, Italy, 2020

On the Resilience of the NFV-MANO: An Availability Model of a Cloud-native Architecture

Besmir Tola, Yuming Jiang, and Bjarne E. Helvik
Department of Information Security and Communication Technology
NTNU-Norwegian University of Science and Technology
Trondheim, Norway
Email: besmir.tola@ntnu.no, yuming.jiang@ntnu.no, bjarne@ntnu.no

***Abstract*—With Network Function Virtualization (NFV), the management and orchestration of network services require a new set of functionalities to be added on top of legacy models of operation. Due to the introduction of the virtualization layer and the decoupling of the network functions and their running infrastructure, the operation models need to include new elements like virtual network functions (VNFs) and a new set of relationships between them and the NFV Infrastructure (NFVI). The NFV Management and Orchestration (MANO) framework plays the key role in managing and orchestrating the NFV infrastructure, network services and the associated VNFs. Failures of the MANO hinders the network ability to react to new service requests or events related to the normal lifecycle operation of network services. Thus, it becomes extremely important to ensure a high level of availability for the MANO architecture. The goal of this work is to model, analyze, and evaluate the impact that different failure modes have on the MANO availability. A model based on Stochastic Activity Networks (SANs), derived from current standard-compliant microservice-based implementations, is proposed as a case study. The case study is used to quantitatively evaluate the steady-state availability and identify the most important parameters influencing the system availability for different deployment configurations.**

***Index Terms*—NFV-MANO, OSM, Availability, SAN models, Docker.**

I. Introduction

Network Functions Virtualization (NFV) is expected to bring significant changes in today's network architectures. By decoupling the network function software from the underlying hardware infrastructure, hence, allowing the software to run on commodity hardware, it provides the necessary flexibility to enable agile, cost-efficient, and on-demand service delivery combined with automated management.

The European Telecommunications Standards Institute (ETSI) defines the NFV-Management and Orchestration (NFV-MANO) framework [1], in the following referred to as simply MANO, as a set of three main functional blocks: the NFV Orchestrator (NFVO), the VNF Manager (VNFM), and the Virtualized Infrastructure Manager (VIM). The NFVO orchestrates all the functionality on the service level including operations like on-board, instantiate, scale, or terminate network services. The VNFM is responsible for the lifecycle management (e.g. instantiation, scaling, and healing) of one or more virtual network function (VNF) instances. It receives management (e.g. deploy, scale, and terminate) instructions for VNFs from the NFVO, which it executes through its interface with the VNFs. The third major component, the VIM, manages the physical infrastructure (NFVI) where the VNFs are executed.

Operating-system-level virtualization technologies, commonly referred to as containers (e.g., Docker [2] or LXD [3]) have enabled a shift in the way applications are deployed going from a monolithic to a microservice-based, i.e., cloud-native, architecture. The later empowers the development, deployment, and operation of large and complex applications as a set of independent smaller and lighter components (i.e., microservices) where each component provides a specific service, and communicates through well-defined lightweight mechanisms. This way, service provisioning becomes more flexible, agile, and reliable [4]. Driven by such benefits, several open source MANO projects leverage a micro-service architecture in deploying and operating MANO components through lightweight containers [5]–[7].

Network operators demand that some of their NFV-based services ensure a carrier grade quality of service [8], i.e. highly reliable and trustworthy. However, service outages, induced by various component failures, are inevitable events that service operators need to deal with. To this end, an automated management and orchestration system embracing resiliency aspects is mandatory for conducting correct counteractions to such events. Failures on the management and orchestration level could jeopardize the functionality of all the network and potentially impact the service delivery by inducing severe outages, which sometimes might be hard to deal with [9], [10]. It is thus of an utmost importance to ensure that a logically centralized control and orchestration system is highly dependable and able to ensure *service continuity* [11]. To this end, ETSI has streamlined several guidelines and requirements of the management and orchestration resiliency capabilities [12].

The objective of this paper is to model and quantitatively analyze MANO steady-state availability when deployed on container-based technologies for various deployment options. To this end, we present an NFV-MANO availability model,

derived from current ETSI-compliant architectures, based on Stochastic Activity Networks (SANs) and perform a quantitative availability assessment aiming at finding the factors mostly impacting system availability. In the model, we incorporate various failure modes on both hardware and software level of the MANO framework. A sensitivity analysis helps us identify the most critical components of the system in terms of relative impact on the system steady-state availability. Moreover, we examine several ways of deploying the software stack aiming at providing higher availability, inspired by current MANO implementations adopting cloud-native practices.

The paper structure is organized as follows. Section II presents the related work and highlights the key contributions. The MANO architecture used to provide the basis of the model is illustrated in Section III. The case study availability model is presented in Section IV. In Section V, we show the results of the analysis and conclude the paper by highlighting the most important insights in Section VI.

II. Related Work

Even though the MANO may have a huge impact on the NFV-enabled network service performance [9], [10], a study of its failure dynamics and overall availability analysis is still missing in the literature. Almost all related work focus on network service availability modeling and quantification. They either focus on specific use cases like virtualized-EPC [13] and virtualized-IMS [14], or model and analyze generic network services provided through NFV-based networks [15], [16] without considering the impact of the MANO on the overall service performance. By aggregating non-state space (Reliability Block diagrams) and state-space models (Stochastic Reward Nets) they quantify and give insights on the service availability and propose appropriate redundancy configurations aiming at providing 5-nines availability.

In a more recent study [17], a composed availability model of an NFV service, based on SANs, is proposed. Each VNF, composing the network service, is considered as a load-sharing cluster and the authors propose separate models for various redundancy mechanisms called Availability Modes and investigate the impact that a faulty orchestrator has on the service availability. Differently, in this paper we propose availability models derived from current micro-service based implementations, i.e., containerized, and provide insights on the most critical parameters affecting the availability for different deployment options.

Availability models of containerized systems for different configurations have been proposed in [18]. The authors propose and compare various container deployments and through both analytic and simulation results they investigate k-out-of-N availability and the system sensitivity to failure parameters. In [19], the same authors present a software tool, called ContAv, for the evaluation of containerized systems' availability. Through the use of both non-state and state-space models designed by the authors, the tool assess the system availability for different configurations and allows a system architect to easily parametrize and perform sensitivity analysis.

In [20], even though not related to availability modeling, the authors propose centralized and distributed mechanisms for a providing a reliable and fault-tolerant microservice-based MANO. The mechanisms exploit load balancing and state sharing and include some tunnable parameters which can help an operator optimise the trade-offs between reliability and the associated costs in terms of resource usage. The proposed setup allows the definition of a cost function which can help the operator determine the best configuration among the centralized and distributed mechanisms.

Compared to the related studies our contribution aims at filling the current gap regarding the availability assessment of a critical element of the NFV architecture. We model and assess a hypothetical MANO system inspired by the current trend of adopting cloud-native software development and maintenance embraced by several architectures. In addition, we investigate various containerized deployment options aiming at achieving high availability levels and identify critical failure parameters impacting the MANO availability.

III. Case Study

An ETSI-compliant MANO should adhere to the specifications streamlined by ETSI and include the main functional blocks which should interact with each other through well-defined reference points and provide an end-to-end network service orchestration. In this paper we extrapolate the deployment options of OSM [5], a well-established architecture hosted by ETSI and led by a large community including both operators and research institutions [5]. OSM is closely aligned with NFV specifications and consists in a production-quality and VIM-independent software stack. Seven releases have been distributed up to now. Release 6 (Release 0 has had a relatively short lifetime) is currently the latest release and includes different installation procedures where the MANO components can be deployed as *dockerized* instances [2] into a hypervisor-based virtualized environment, a public hosting infrastructure, or directly into a proprietary commodity hardware. The latter represents the most common way of deploying and running the OSM stack. It represents the most advanced release including among others network service and slicing capabilities, enhanced user interface, and a lighter orchestrator.

The default installation deploys 13 docker containers running in a Docker *swarm mode* with each component having one single replica. Docker *swarm mode* is a native feature of Docker for managing and orchestrating a cluster of Docker engines called *swarm*. It entails several cluster management characteristics like: i) decentralized configuration of cluster nodes at runtime, ii) automatic scaling, iii) automatic cluster state reconciliation, and iv) integrated load balancing. A swarm is a cluster of Docker nodes, running in a *swarm mode*, and they act as managers, who manage the swarm membership and delegate tasks, and workers which run swarm services.

A Docker node can be a manager, worker, or both. A *swarm* may consist in only one node which by default will act as a manger and worker at the same time, but it cannot be only a worker without a manager. We refer to this as the

Manager configuration. In case the cluster is composed of worker and manager nodes, we refer to it as *Manager-Worker* configuration.

One of the key features of a swam is the automatic cluster state reconciliation. This is very important in terms of fault management policies. In case one of the services of the cluster is down, the swarm state changes and the manager immediately respawns the failed container/containers on other available node and the service stack becomes healthy again.

IV. Availability Model

A SAN is a modeling formalism with which detailed performance, dependability, or performability models can be implemented in a comprehensive manner [21]. SANs are stochastic extentions of Petri Nets consisting of four primitives: *places*, *activities*, *input gates*, and *output gates*. Places are graphically represented as circles and contain a certain number of tokens which represent the *marking* of the place. Activities are actions that take a certain amount of time to fire and move tokens from one place to another. Input and output gates define marking changes that occur when an activity completes. Different from output gates, the input gates are also able to control the enabling of activity completion, i.e., firing.

In the following, we illustrate the proposed models representing the different MANO configurations.

A. Manager Configuration

Past studies classify software faults into two main categories, Bohrbugs and Mangelbugs [22]. Bohrbugs, otherwise called deterministic, are typically easily reproducible since they tend to manifest themselves consistently under the same conditions. They often may lead to a software crash or process hanging and the bugs need to be identified and resolved. Mandelbugs are bugs whose activation and error propagation are complex. As a result, it is quite hard to reproduce and their manifestation is transient in nature. They are usually caused by timing and synchronization issues resulting in race conditions. A retry operation or software restart may resolve the issue. There is a further subtype of Mandelbugs that is related to an aspect know as software aging. Software aging is a well-know issue which characterizes the software failure rate due to phenomena like the increase of software execution period [23]. It has been shown that the increase of process runtime is a common cause to the increase of software failure rate and the system performance may degrade over time. Typical faults in IT systems caused by aging effects include resource leakages, numerical error, or data corruption accumulation. Therefore, the failure might occur as a result of the increase of system uptime. Common methods of recovering from such failures rejuvenation techniques consisting of restart and/or reboots procedures [24].

On the software level, for the scope of our investigation, we differentiate between two types of software failures, non-aging related failures and aging related failures. The former set aims at representing both Bohrbugs and non-aging Mangelbugs where the majority of these failures can be recovered through a manual intervention for software repair and the latter represents the failures due to aging effects where the majority of these failure are recovered by a software restart/reboot.

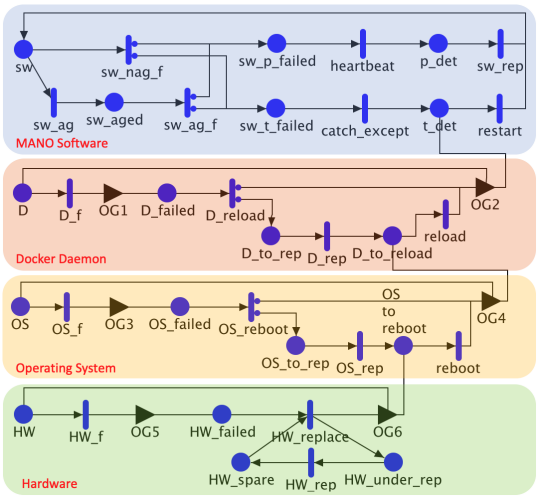


Fig. 1: SAN availability model of the MANO deployed in a Manager configuration.

Fig. 1 illustrates the SAN model of the *Manager* configuration. It consists of the deployment of the MANO containerized software into one physical node which acts as both manger and worker for the service tasks. The model includes the MANO software, Docker daemon, OS, and hardware components and their relative places *sw*, *D*, *OS*, and *HW* are initialized with 1 token each, indicating a fully working system. Similar to previous works (see related work [13]–[16]), it is assumed that all the timed activities follow a negative exponential distribution unless otherwise specified.

The aging effect is represented through a specific timed activity *sw_ag* with rate $\lambda_{\mathrm{sw_{ag}}}$ which defines the time it takes for the software to age, i.e., the average time that the software accumulates errors that might lead to an aging-related failure. The timed activities *sw_ag_f* and *sw_nag_f* represent the aging and non-aging related software failure events with rates $\lambda_{\mathrm{sw-fail_{ag}}}$ and $\lambda_{\mathrm{sw-fail_{nag}}}$, respectively. For both events, we differentiate between two types of software failures based on their recovery process. To this end, we make use of case probabilities associated to the timed activities where C_{nag} defines the probability that a non-aging related failure event is recovered with a software restart. With probability $1-C_{\mathrm{nag}}$ the failure recovery requires a manual intervention for executing a software repair. Similarly, C_{ag} defines the probability that an aging related failure is recovered with a software restart and with $1 - C_{\mathrm{ag}}$ the recovery requires a software repair.

Once a software failure is experienced, a token is placed in either *sw_p_failed* or *sw_t_failed* defining the recovery process the software will undergo. *heartbeat* and *catch-exception* represents the detection of the failures and are defined with de-

terministic times μ_h and μ_c. *sw_rep* and *restart* represents the repair (including any eventual reboot or upgrade of software) and restart events of the software with rate $\mu_{\mathrm{SW_{rep}}}$ and $\mu_{\mathrm{SW_{res}}}$. On the docker engine level, i.e., daemon, *D_f* and *D_reload* represents the failure and recovery events of the daemon with rates λ_{D} and $\mu_{\mathrm{D_r}}$, respectively. The recovery entails a daemon reload where with probability C_{D} a daemon reload recovers the failure and with $1 - C_{\mathrm{D}}$ a hard repair is needed. The later is defined through the activity *D_rep* with rate $\mu_{\mathrm{D_{rep}}}$. Once the daemon is repaired, an additional reload is performed to fully recover it. Similarly to the daemon, the operating system level is modeled with the same dynamics having specific failure and repair parameters which we introduce in Section V. On the hardware level, *HW_f* and *HW_replace* represents the failure and recovery events of the hardware with rates λ_{HW} and $\mu_{\mathrm{HW_{rep}}}$, respectively. The place *HW_spare* indicates the spare hardware equipment used to replace the failed hardware and is initialized with 1 token.

Finally, the following output gates define the token marking movements among places: *OG1/OG3/OG5* manage the failure events of the daemon, OS, and hardware levels, respectively. When their related timed activities fire, connected to their incoming arcs, the output gate places 1 token in the respective failed position and sets to zero the upper-level places. This is because a failure of the physical hardware will cause a failure of the OS which in turn impacts the operational state of the daemon and MANO software as well; *OG2/OG4/OG6* places 1 token in their relative working place, i.e., *D/OS/HW*, and the relative upper-level places to which they are connected by outgoing arcs. For example, a recovery from a daemon failure brings the daemon in the up state but requires a restart of the MANO software for a fully working MANO.

B. Manager-Worker Configuration

The *Manager-Worker* swarm configuration consists of two separate nodes forming a cluster where the OSM stack is deployed on the worker node and the Manager node performs the control and scheduling of tasks. Fig. 2 depicts the *Manager-Worker* SAN model. To distinguish the models of the two entities, we make use of a suffix *_M* for all the places and activities regarding the manager part. The system is fully working if there is a token in either of the *sw*, *sw_aged*, or *sw_M* places.

On the worker node, the MANO software component is similar to the *Manager* configuration except for the recovery phase where once a failure is detected, the containers running the software are respawned, through the timed activities *respawn* or *respawn1*, in the manager node. We distinguish two cases: when a software repair is needed, the token is moved from *p_det* of the worker node to *p_det_M* of the manager. In the other case, the token is moved from *t_det* to *sw_M* indicating that a respawn, i.e., container restart, is sufficient to recover the system. However, for both cases, we consider the eventuality of a respawn process that fails. To this end, we consider two case probabilities associated with the timed activities. With probability C_{respawn}, the container respawn

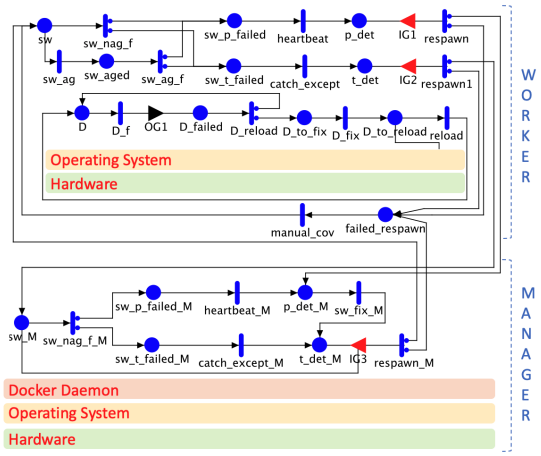


Fig. 2: SAN availability model of the MANO deployed in a Worker-Manager configuration.

is successful and $1 - C_{\mathrm{respawn}}$ it fails. In the latter, there is a need for a manual coverage, represented by *manual_cov*, and the token is placed back in place *sw*. In order for the respawn to instantiate, the hosting manager node needs to be operational and this is controlled by the enabling gates *IG1/IG2* which enable the respawn only if the daemon, OS and hardware of the manager are working, i.e., their respective places *D_M*, *OS_M*, and *HW_M* contain each 1 token. In addition, differently to the *Manager* setup, once the daemon fails, there is just the recovery of the daemon since the MANO software is immediately respawned in the manager node. The rest of the model is similar to the *Manager* configuration, hence we omit further illustrating.

On the manager node, once a token is deposited in *sw_M*, the system is again operational. While the software is running in this node, we assume that it is subject to only non-aging software failures. This is because *swarm mode* best practices suggest that the worker node should be the dedicated node for handling task requests in a 'normal' condition. Therefore, we limit the hosting of the MANO software to the manager node only for the period the worker node is failed. To this end, the input gate *IG3* enabled a respawn of the software containers from the manager node to the worker node once the worker node is up and running again and ready to accommodate the containers. As a result, the manager node will host the containers for a relatively short time compared to the software aging time, hence making the assumption of only non-aging failure events while the software is running on the manager node a reasonable assumption. The rest of the manager components, i.e., daemon, OS, and hardware are similar to the *Manager* configuration.

C. Replicated Configuration

One of the most advantageous *swarm* features is automatic scaling and integrated load balancing. In case MANO utilization gets close to its resource limits, an operator can easily

TABLE I: Availability model parameters.

Intensity	Time	Description [Mean time to]
$\lambda_{\mathrm{sw_{ag}}}^{-1} = 1$	week	MANO software aging
$\lambda_{\mathrm{sw-fail_{ag}}}^{-1} = 3$	days	next MANO software failure after aging
$\lambda_{\mathrm{sw-fail_{nag}}}^{-1} = 2$	month	next MANO non-aging software failure
$\mu_{\mathrm{sw_{rep}}}^{-1} = 1$	hour	MANO software repair
$\mu_{\mathrm{sw_{res}}}^{-1} = 30$	seconds	MANO software restart
$\mu_{h}^{-1} = 10$	seconds	heartbeat*
$\mu_{c}^{-1} 10 = 1$	millisecond	catch exception*
$\lambda_{\mathrm{D}}^{-1} = 4$	months	next daemon failure
$\mu_{\mathrm{D_{rep}}}^{-1} = 1$	hour	daemon repair
$\mu_{\mathrm{D_{l}}}^{-1} = 15$	seconds	daemon reload
$\lambda_{\mathrm{OS}}^{-1} = 4$	months	next OS failure
$\mu_{\mathrm{OS_{rep}}}^{-1} = 1$	hour	OS repair
$\mu_{\mathrm{OS_{r}}}^{-1} = 5$	minutes	OS reboot
$\lambda_{\mathrm{HW}}^{-1} = 6$	months	next hardware failure
$\mu_{\mathrm{HW_{rep}}}^{-1} = 4$	hours	hardware repair
$\mu_{\mathrm{HW_{replace}}}^{-1} = 1$	hour	hardware replace
$\mu_{\mathrm{respawn}}^{-1} = 1$	minute	respawn MANO software containers
$C_{\mathrm{nag}} = 0.3$		prob. for non-aging transient failures
$C_{\mathrm{ag}} = 0.7$		prob. for aging transient failures
$C_{\mathrm{D}} = 0.9$		daemon reload coverage factor
$C_{\mathrm{OS}} = 0.9$		OS reboot coverage factor
$C_{\mathrm{respawn}} = 0.9$		respawn coverage factor
$N_{\mathrm{spare}} = 1$		Number of spare hardware

*Deterministic time

spin up additional replicas of the containers and the swarm integrated load balancer will manage the task scheduling without any additional configuration required from the operator. Spinning up additional replicas can bring advantages both in terms of performance and availability. To this end, we consider the case where multiple MANO instances are running in both *Manager* and *Manager-Worker* setups and the system is considered availability if at least one replica is working.

For modeling the replicated configuration, it is sufficient setting the number of tokens in the *sw* place equal to the number of replicas for both *Manager* and *Manager-Worker* configurations. This way, the models resemble a setup where multiple containers, for each of the MANO components, are launched and run in the same OS and physical hardware.

V. Numerical Analysis

In this section we present the sensitivity analysis of the steady-state availability for both configurations and failure impact on the overall unavailability. The presented models are defined in the Möbius software tool [25] and the numerical analysis is performed using discrete-event simulations, integrated in the tool, with 95% confidence interval and 0.05 width of relative confidence interval.

A. Manager configuration: Sensitivity Analysis

We performed a sensitivity analysis to determine which of the parameters have the highest impact on the steady-state availability of the Manager configuration. The SAN model parameters are retrieved from previous literature [17]–[19] and are illustrated in Table I. They represent the reference values and given these parameters, the achieved MANO availability is presented in Table II, together with its component availabilities where the latter are derived from individual dynamics, i.e., not influenced by underlying component failures. It can be

TABLE II: Steady-state availability of Manager Configuration.

	MANO	MANO Sw	Daemon	OS	Hardware
Availability	0.99751	0.99787	0.99964	0.99967	0.99975

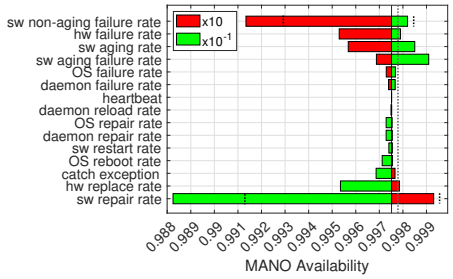


Fig. 3: Sensitivity analysis for the Manager configuration.

seen that the software part is the most fragile component. For computing a sensitivity analysis, the parameters regarding failure and recovery events were increased and reduced by one order of magnitude, i.e., $\times 10$ and $\times 10^{-1}$, from their reference values. The availability sensitivity to these parameters, separated into failure and recovery events, is presented in Fig. 3.

From a failure events perspective, the most important parameter is software non-aging failure rate followed by hardware, software aging, and software aging failure rates. Among these, it is the latter that brings the highest improvement on the steady-state availability when there is an order of magnitude reduction of the relative intensities. On the contrary, for the same level of parameter reduction, software repair rate has the highest impact by reducing the system availability from 0.997 to almost 0.988 followed by the hardware replace rate. At the same time, the highest improvement is achieved for a software repair rate increase reaching 0.9993.

Beside the failure and repair parameters, the probability factors that define the types of software failures which may have an important influence on the overall availability. The choice of the reference values is driven by common assumptions that non-aging failures, i.e., deterministic failures, often lead to system crashes and debugging processes can improve software robustness by identifying and resolving the bug. Hence, choosing a $C_{\mathrm{NAG}} = 0.3$ means that the majority of such failures require a software repair. The opposite is valid for aging-related failures which more often may lead to *transient* failures that can be resolved by simply restarting the software. To this end, we explore a range of these factors and their impact on the availability.

Fig. 4 presents the MANO availability for the different combinations. We notice that for the same level of reduction, the aging factor achieves a much higher availability improvement compared to the non-aging factor. This indicates that the system can benefit more from the transient nature of aging related bugs than those of deterministic failures, otherwise called Bohrbugs.

1) Failure Impact: It is expected that on average around 52 failures per year will contribute to a total duration of

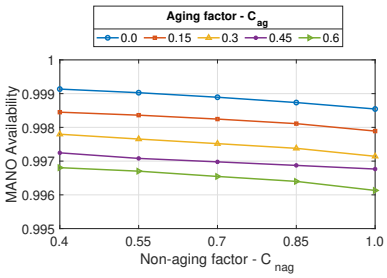


Fig. 4: Impact of aging and non-aging factors on the availability.

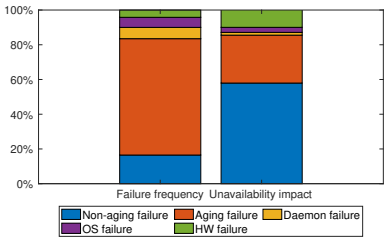


Fig. 5: Failure frequency and relative impact on the unavailability.

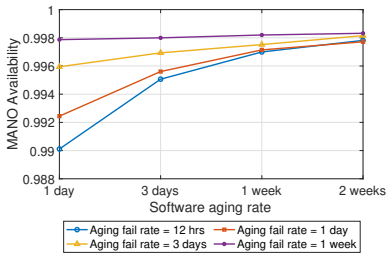


Fig. 6: Impact of software aging and its failure rate on the availability.

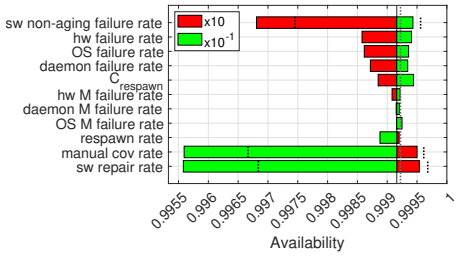


Fig. 7: Sensitivity analysis for the Manager-Worker configuration.

21.7 hours of unavailability. The contribution of different failure types, in terms of their frequency and the impact on the system downtime, is presented in Fig. 5. It can be observed that software failures are the predominant events, accounting for almost 84% of all the failures. In particular, we notice that despite software aging failures represent almost 60% of the overall failures, they lead to only 27.5% of the MANO downtime. On the other hand, non-aging software related failures consists of the 16% of the total failure but contribute to almost 58% of the total downtime. In addition, hardware failures represent around 4% of all the failures and they contribute to more than 10% of the system downtime.

2) Software aging impact: In the sensitivity analysis we noticed that aging failure rate may have a considerable impact on the availability of the MANO. However, software aging rate and aging failure rate are very unpredictable parameters since they depend on several factor that may be out of developer's control such as software utilization rate, i.e., system load, operating infrastructure and software implementation. We explore a wide range of software aging parameters, varying the aging rate between 1 day and 2 weeks and the aging failure rate between 12 hours and 1 week. The MANO availability for different combinations of the parameters is presented in Fig. 6. It can be seen that the impact of the aging failure rates depends greatly on the rate of aging. When the time it takes the software to age is short, i.e, lower than 3 days, the aging failures have a much higher impact on the availability.

B. Manager-Worker Configuration: Sensitivity Analysis

The deployment of the MANO stack in a *Manager-Worker* configuration entails an automatic respawn of the containers in the manager node in case the MANO experiences a failure. This setup is suitable for recovering system outages due to external components like the daemon engine, OS, or hardware failures of the hosting node, i.e., worker node. However, such procedure is triggered only if the manager is capable, i.e., fully working, of hosting the containers running the software components. Therefore, the respawn procedure is constrained by the operational state of the manager node. Fig. 7 depicts the sensitivity analysis of some of the critical parameters in this setup. First, we notice that applying this configuration, i.e, by joining another node into the swarm, the MANO availability is increased from 0.99751 of the *Manager* case to 0.99916 for the reference parameters (refer to Table I). Second, we observe that software repair and manual coverage rate, i.e., the mean time to manually recover a failed respawn procedure, may have a tremendous impact in deteriorating the system availability where for the latter despite the successful respawn factor is set to 0.9. Moreover, we evidence that the external failures on the worker node (hardware, OS, and daemon) have a much higher impact than those of the manager node parameters. This is due to the policy that a respawn of the containers from the manger to worker is done as soon as the worker node is ready to re-host the containers, i.e, it is recovered from failures which triggered a respawn to the manager in the first place.

C. Replicated Configuration

Fig. 8 illustrates the gains in terms of system availability due to the introduction of additional replicas. For both cases, there is a relatively restricted gain which is due to the constraints posed by external components availabilities since all the replicas are still running on the same physical node. However, engaging multiple replicas brings additional benefits in terms of the reduction of the impact of critical parameters as highlighted by the dotted lines in both Fig. 3

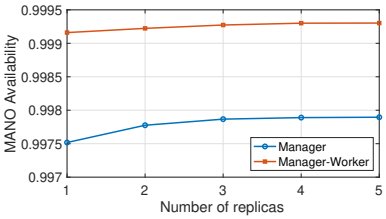


Fig. 8: Availability for different container replicas.

and Fig. 7, representing the cases with 4 replicas for the most impactful parameters in the *Manager* and *Manager-Worker* configurations. We notice that for both configurations, the largest reduction is achieved on software repair rate followed by the non-aging failure rates when the related reference parameters are degraded by one order of magnitude. In addition, a significant reduction is achieved for the impact the manual coverage rate has on the *Manager-Worker* setup.

One solution to the limited gain when multiple replicas are applied can be Docker Universal Control Plane (UPC) for enterprises which envisions a complex architecture that maximally leverages docker swarm scalability for achieving high availability [26]. It consists of a docker swarm with multiple manager and worker nodes instantiated into separated physical nodes. This solution promises much higher availability levels than those anticipated in our study and could represent a viable solution for network operators deploying and managing a cloud-native MANO. We leave the investigation of this solution for future work.

VI. Conclusion

In this paper, we present an availability model for a cloud-native NFV-MANO architecture from which we analyze and quantify its steady-state availability. We have included the most typical failure modes and evaluated their impact through sensitivity analysis for different containerized deployments. The proposed model, based on Stochastic Activity Networks (SANs), captures both failure and recovery dynamics involving containerized applications and the effects of software aging. The investigation has shown that adopting containerized technologies with standard deployments having both single and multiple replicas deployed into a single physical node is not sufficient for achieving "5-nines" availability. The sensitivity analysis also revealed that non-aging-related software failures and software repair stand out as key important failure and repair parameters, respectively. When clustering mechanisms such as Docker swarm mode with separated worker and manager nodes are adopted, we observed that the MANO availability is further increased and the above parameters become less critical when multiple MANO container replicas are engaged. Software aging may have a considerable impact on the availability and we showed the relationship between aging effects and failures related to it. As future work we will consider modeling and analysis of more complex swarm deployment options similar to those designed for Docker Enterprise solutions.

Acknowledgment

This research was funded by the joint EU FP7 Marie Curie Actions Cleansky Project, Contract No. 607584.

References

[1] G. N. ETSI, "Network Functions Virtualisation (NFV): Architectural Framework," *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.
[2] Docker Website. Accessed: 2019-10-30. [Online]. Available: "https://www.docker.com/"
[3] LXD. Accessed: 2019-10-30. [Online]. Available: "https://linuxcontainers.org/"
[4] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," in *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.
[5] OSM website. Accessed: 2019-10-30. [Online]. Available: "https://osm.etsi.org"
[6] OpenBaton website. [Online]. Available: "https://openbaton.github.io"
[7] SONATA website. Accessed: 2019-10-30. [Online]. Available: "https://www.sonata-nfv.eu/"
[8] ETSI, "Reliability; Report on Models and Features for E2E Reliability," ETSI, Tech. Rep. GS REL 003 v1.1.2, 2016-07.
[9] A. J. Gonzalez *et al.*, "Dependability of the NFV orchestrator: State of the art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3307 – 3329, 2018.
[10] G. Nencioni *et al.*, "Orchestration and control in software-defined 5G networks: Research challenges," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.
[11] I. N. ETSI, "ETSI GS NFV-REL 001 v1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.
[12] ——, "ETSI GR NFV-REL 007 v1.1.2: Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities," 2017.
[13] A. Gonzalez *et al.*, "Service availability in the NFV virtualized evolved packet core," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.
[14] M. Di Mauro *et al.*, "IP multimedia subsystem in an NFV environment: availability evaluation and sensitivity analysis," in *2018 IEEE NFV-SDN*. IEEE, 2018, pp. 1–6.
[15] ——, "Service function chaining deployed in an NFV environment: An availability modeling," in *IEEE CSCN*. IEEE, 2017, pp. 42–47.
[16] ——, "Availability modeling and evaluation of a network service deployed via NFV," in *International Tyrrhenian Workshop on Digital Communication*. Springer, 2017, pp. 31–44.
[17] B. Tola, G. Nencioni, B. E. Helvik, and Y. Jiang, "Modeling and evaluating NFV-enabled network services under different availability modes," in *IEEE DRCN*. IEEE, March 2019.
[18] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Transactions on Services Computing*, 2018.
[19] S. Sebastio, R. Ghosh, A. Gupta, and T. Mukherjee, "ContAv: A Tool to Assess Availability of Container-Based Systems," in *IEEE SOCA*. IEEE, 2019, pp. 25–32.
[20] T. Soenen *et al.*, "Optimising microservice-based reliable NFV management and orchestration architectures," in *IEEE RNDM*, Sep. 2017, pp. 1–7.
[21] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *School organized by the European Educational Forum*. Springer, 2000, pp. 315–343.
[22] M. Grottke and K. S. Trivedi, "Software faults, software aging and software rejuvenation," *The Journal of Reliability Engineering Association of Japan*, vol. 27, no. 7, pp. 425–438, 2005.
[23] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *2008 IEEE ISSRE Wksp*, Nov 2008, pp. 1–6.
[24] K. S. Trivedi *et al.*, "Recovery from failures due to Mandelbugs in IT systems," *Proceedings of IEEE PRDC*, no. December, pp. 224–233, 2011.
[25] Möbius: Model-based environment for validation of system reliability, availability, security and performance. Accessed: 2019-10-30. [Online]. Available: "https://www.mobius.illinois.edu"
[26] Docker Reference Architecture: Docker Enterprise Best Practices and Design Considerations. Accessed: 2019-10-30. [Online]. Available: "https://success.docker.com/article/docker-enterprise-best-practices\#highavailabilityindockerenterprise"

PAPER C

Model-Driven Availability Assessment of the NFV-MANO with Software Rejuvenation

Besmir Tola, Yuming Jiang, and Bjarne E. Helvik

IEEE Transactions on Network and Service Management, 2021
Early Access - DOI: 10.1109/TNSM.2021.3090208.

Model-Driven Availability Assessment of the NFV-MANO with Software Rejuvenation

Besmir Tola, Yuming Jiang, and Bjarne E. Helvik

***Abstract*—Network Function Virtualization enables network operators to modernize their networks with greater elasticity, network programmability, and scalability. Exploiting these advantages requires new and specialized designs for management, automation, and orchestration systems which are capable of reliably operating and handling new elements such as virtual functions, virtualized infrastructures, and a whole new set of relationships among them. Operations such as resource allocation, instantiation, monitoring, scaling, or termination of virtual functions are key lifecycle operations that NFV management and orchestration (NFV-MANO) frameworks need to correctly perform. Failures of the NFV-MANO prevent the network ability to respond to new service requests or events related to the normal lifecycle operation of network services. Thus, it is important to ensure robustness and high availability of the MANO framework. This paper adopts a model-driven approach to predict the availability of the NFV-MANO and assess the impact that different failure modes have. We propose different models, based on Stochastic Activity Networks (SANs), which abstract various MANO deployment configurations, inspired by current containerized open-source MANO implementations. Moreover, we integrate software rejuvenation and investigate the trade-off between its associated overhead and system availability increase. An extensive experimental campaign with fault-injection techniques on a real-life MANO implementation allows to derive a number of realistic recovery parameters. The case studies are used to quantitatively evaluate the steady-state availability and identify the most important parameters influencing system availability for the different deployment configurations.**

***Index Terms*—NFV-MANO, Availability, Software aging, Software rejuvenation, SAN models, Containers.**

I. Introduction

NETWORK Function Virtualization (NFV) empowers an innovative transformation of today's network architectures. At the core of the paradigm lies the separation of the network functions from the underlying hardware platforms. Network-based services can be realized through virtualized software entities, commonly referred to as Virtualized Network Functions (VNFs), which can be executed in general purpose hardware rather than requiring specialized purpose-built platforms. They can embody network functions such as Routers (vRouter), Firewalls (vFW), and Load Balancers (vLB) [1], and can be chained together to provide advanced full-scale network services [2], [3].

As defined by the European Telecommunications Standards Institute (ETSI), the standard high-level architecture of NFV incorporates three main blocks that are the NFV infrastructure (NFVI), the VNFs, and a logically centralized Management and Orchestration (MANO) entity [4]. The NFVI provides a virtualization environment for the deployment and execution of VNFs, including virtual compute, storage and networking resources. VNFs are software implementations of network functions which should be able to interact with other VNFs for providing composed network services. The MANO performs life-cycle management of VNFs and NFs, and the orchestration of infrastructure resources supporting their execution.

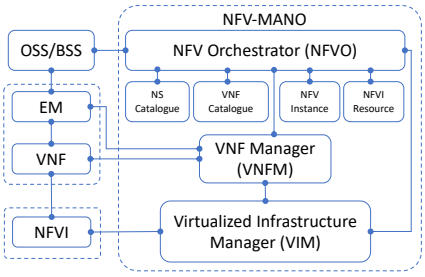


Fig. 1. The NFV-MANO high-level framework (adapted from [4]).

Removing the dependency between the network function software and the hardware infrastructure is expected to bring a variety of advantages in how networks are operated and managed [5], [6]. Nonetheless, it also brings additional implications on the network management systems that need to be extended beyond traditional FCAPS (Fault, Configuration, Accounting, Performance, Security) management services in order to provide life-cycle management of a new set of entities such as the VNFs, network services (NSs), and the virtualized infrastructure [7]. In addition, the operators need to ensure that service lifecycle is adequately orchestrated and managed such that service needs and requirements are met. To this aim, ETSI has defined a specific NFV-Management and Orchestration (NFV-MANO) framework [4], in the remainder simply referred to as MANO. Fig. 1 presents the high-level architectural view of the MANO framework which consists of the following functional blocks:

NFV Orchestrator (NFVO): It is the primary responsible for the orchestration and management of the NFV infrastructure (NFVI) resources across multiple virtualized infrastructure managers (VIMs) and the lifecycle of the network services including operations like on-boarding, instantiating, scaling, or terminating network services. It also interacts with the operation and business support system (OSS/BSS), through which customers/operators perform service operations including instantiating, updating, or terminating a service.

VNF Manager (VNFM): It is the block in charge of the configuration and lifecycle management of one or more VNFs.

The VNFM receives from the NFVO management instructions for VNFs (e.g. deploy, configure, and terminate) and executes them through its interfaces with the VNFs. The NFVO and VNFM jointly work to ensure that the network services and their corresponding VNFs meet the service quality requirements (e.g. reliability, latency or throughput).

Virtualized Infrastructure Manager (VIM): It manages and orchestrates the physical resources, i.e., compute, storage, and networking, upon which the VNFs are executed.

In addition to the three main blocks, a set of catalogs represent the repositories of on-boarded NS, VNF packages and the relative instances. Moreover, another repository holds information regarding available/consumed NFVI resources, as abstracted by the VIM.

An important end-user expectation is the high-availability level that NFV-enabled services will deliver. This is because several of the envisioned NFV service use cases fall into the telecom domain in which carrier-grade quality of service is a strict requirement, i.e., 5-nines availability [8], [9]. Moreover, NFV is foreseen to be a main pillar of future 5-th generation (5G) networks where stringent delay and availability demands (5-nines or more, i.e., less than 5 minutes of yearly downtime) are expected [10]. However, ensuring high-availability levels can be an arduous challenge that network operators need to cope with since service outages, induced by various component failures, are inevitable events. High availability is typically achieved by providing fault-tolerance capabilities through the allocation of redundant elements [11] over which the system switches upon the failure of primary components. To this end, a robust management and orchestration system featuring resiliency facets is mandatory for conducting correct and timely counter-actions to such events [12], [13]. Moreover, failures of the MANO itself could jeopardize the overall functionality of the network and potentially impact the service delivery by causing severe outages, which sometimes may be hard to deal with [14], [15]. It is thus of an utmost importance to ensure that a logically-centralized management and orchestration system is highly dependable and able to ensure *service continuity* [8]. To highlight the importance of a dependable MANO system, ETSI has published guidelines and requirements regarding the MANO resiliency capabilities [16].

Cloud-native application engineering is a consolidated approach in designing, building, and running applications that can fully exploit cloud computing benefits. An important pattern of cloud-native applications is that they are composed of microservices where each of these small services can operate independently of each other, provide a specific service, and communicate through well-defined mechanisms [17]. Moreover, cloud-native applications are packaged as a set of lightweight containers (e.g., Docker [18] or LXC [19]) aiming at providing context isolation, highly accessible, scalable and portable virtual environments. This way, service provisioning becomes more flexible, agile, and reliable [20]. Driven by such benefits, there is an increasing trend in adopting cloud-native design patterns also for virtualized network functions through deploying and running networking code as containerized software [21]–[23]. This trend has been embraced also by some of the most prominent open-source MANO projects which leverage a microservice architecture in deploying and operating MANO components through lightweight containers [24]–[26].

In this paper, we take a model-driven approach for predicting the availability of container-based MANO implementations and evaluating the impact that variations of critical failure and repair parameters have on the overall system availability. We adopt Stochastic Activity Networks (SANs) modeling formalism and perform a quantitative assessment of various deployment configurations enriched with fault-tolerance on both software and hosting infrastructure. An extensive sensitivity analysis allows us to localize bottleneck parameters for each of the deployment setups. The main contributions of this article introduce:

(i). Modeling abstractions for containerized MANO implementations, integrated with software rejuvenation and deployed in different redundant configurations, which are inspired by practices adopting cloud-native designs.
(ii). An experimental campaign on a containerized MANO platform aiming at retrieving realistic system recovery parameters.
(iii). A characterization of failure dynamics and an extensive sensitivity analysis targeting dependability metrics for both centralized and distributed MANO deployments.
(iv). Computational results that characterize failure dynamics, and sensitivity analysis that identifies critical parameters and rejuvenation policies for maximizing the steady-state availability (SSA).

The remainder is organized as follows. Section II presents the related work and highlights the key novelties. Section III presents the case study MANO architecture and the mapping of the components to the ETSI framework. The different deployment configurations that considered in this study are illustrated in Section IV. Section V introduces the software aging phenomenon and the mechanisms to cope with its related effects. The availability models resembling the different configurations are presented in Section VI. In Section VIII, we show the results of the analysis and conclude the paper by highlighting the most important insights in Section IX.

II. Related Work

NFV dependability is an important challenge and a significant research effort has been put on addressing this challenge. ETSI has promulgated various NFV specifications in regard to requirements, capabilities, and models for assessing reliability, availability, and service continuity [8], [16], [27], [28].

Most of the model-based studies evaluating NFV availability focus on network service availability modeling and quantification without considering the potential impact that the MANO may have on the end-to-end service availability. These studies either focus on specific NFV use cases such as virtualization of the evolved packet core (EPC) system [29] and the virtualization of the IP multimedia subsystem (IMS) [30], or model and analyze generic network services provided through NFV-enabled infrastructures [31], [32], without regarding the effect that a faulty MANO may have on the overall service availability. However, as emphasized by ETSI, the MANO plays a crucial role in fault management [16] and it may have a huge

impact on the NFV-enabled network service performance [14], [15]. As a result, a study of its failure dynamics and availability analysis can be an important contribution for predicting and identifying MANO availability bottlenecks.

In [29], the authors present an availability model of a virtualized EPC by using stochastic activity networks. The study assesses the system availability through discrete-event simulation and identifies the most relevant criteria to account for by service providers in order to meet a certain availability level. The proposed model includes also the MANO system but no analysis is performed.

A two-level hierarchical availability model of a network service in NFV architectures has been proposed in [31]. By aggregating non-state space (Reliability Block diagrams) and state-space models (Stochastic Reward Nets), the authors quantify the SSA and perform a sensitivity analysis to determine the most critical parameters influencing the network service availability. Similarly, in [32], they extend such analysis by including the VIM functionality, as the entity responsible for the management of the physical infrastructure resources, into the reliability block diagram (RBD). Their main findings indicate that a relatively small increment of hypervisor or VNF software failure intensity has a marginal effect on the service availability. In addition, they identify the most appropriate redundancy configuration in terms of additional replicas for providing fine-nines availability. The same authors model and assess the availability of an NFV-oriented IP multimedia subsystem (IMS) [30]. Exploiting the same modeling techniques, they assess the availability of a containerized IMS and perform a sensitivity analysis on failure and repair rate of some of the IMS components. In addition, they identify the best k-out-of-n redundancy configuration for each IMS element such that a five-nine availability is reached.

In [33], the authors propose a hierarchical availability model of an NFV service by adopting stochastic activity networks. Each VNF, composing the network service, is considered as a load-sharing cluster and specific separate models abstracting different redundancy mechanisms, called Availability Modes, are constructed. The study performs a sensitivity analysis on various critical parameters and also investigates the impact that a faulty orchestrator has on the service availability. Differently, in this paper we focus on the MANO system rather than the NFV-service and propose availability models derived from current microservice based implementations. Moreover, our study provides insights on the most critical parameters specifically affecting the MANO availability for different deployment options and under software proactive maintenance.

Even though different from a model-based investigation, the authors of [34] propose centralized and distributed mechanisms for providing a reliable and fault-tolerant microservice-based MANO. The mechanisms exploit load balancing and state sharing and include some tunnable parameters which can help an operator optimise the trade-offs between reliability and the associated costs in terms of resource usage. The proposed setup allows the definition of a cost function which can help the operator determine the best configuration among the centralized and distributed MANO deployments.

One of the first studies to carry out an availability assessment of containerized systems is [35]. The authors propose availability models for different configurations and compare various container deployments. Through both analytic and simulation computational results they investigate the k-out-of-N redundancy configuration and evaluate the availability sensitivity to different failure parameters. In [36], the same authors present the development of a software tool called ContAv which can perform the evaluation of containerized systems' availability. Through the use of both non-state and state-space models, designed by the authors, the tool assesses the system availability for different configurations and allows a system architect to easily parametrize and perform sensitivity analysis. However, both works assume that container restarts are sufficient for recovering the containerized application. This can be an oversimplified assumption since the application source code, built in the container image, can also be subject to failures which require a software fix or patch [37]–[39]. Moreover, the work disregards the hardware infrastructure which can also be a dependability bottleneck despite the container instances are provided with instance redundancy. The models presented in our work relax these assumptions. In addition, we investigate also the impact that both aging and non-aging related bugs have on the system availability, where software rejuvenation is considered as a countermeasure.

Built on our previous attempt to characterize failure and recovery behavior of the MANO system [40], the present work extends the investigation in several aspects. One is more truthful modeling abstractions for MANO implementations. Another is a model for distributed MANO deployments which encompasses redundancy on both software and hosting infrastructure. In addition, a component-wise MANO model is introduced. In all these, the impact of software proactive maintenance, in the form of software rejuvenation, is particularly factored in. Moreover, we exploit fault-injection techniques and perform experimental trials on a realistic testbed based on which some key model parameters are retrieved for use in numerical analysis.

III. Case Study

There are currently several open-source MANO framework implementations, such as OSM [24], SONATA [26], and ONAP [41]. To restrain the nonconforming development of MANO architectures with incompatible APIs, ETSI has provided several guidelines of the different MANO architectural options [6], [42], which are currently widely accepted within the sector. Despite the various options, an ETSI-compliant architecture should adhere to the streamlined specifications and include the main functional blocks, which should provide an end-to-end network service management and orchestration.

In this paper we extrapolate the deployment options of OSM, a well-established architecture supported by ETSI and led by a large community of network operators and research institutions [24]. OSM claims to be closely aligned with ETSI NFV information models and consists in a production-quality and VIM-independent software stack. Eight releases have been distributed up to now and Release 8 is currently the latest release. It includes different installation methods

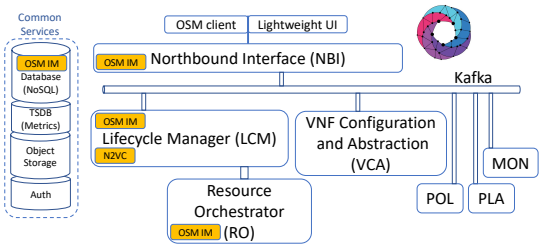


Fig. 2. OSM architectural view (adapted from [24]).

where the MANO components can be deployed as *dockerized* instances [18] into a hypervisor-based virtualized environment, a public hosting infrastructure, or directly into a proprietary commodity hardware. The latter represents a common way of deploying and running the OSM stack.

Fig. 2 illustrates the architectural view of OSM with the specific names of the stack components. The LCM module stands for Lifecycle Manager and plays the role of the NFVO in the ETSI MANO framework. The VCA assumes the role of the VNFM and exploits a Juju controller [43], deployed in a Linux Container (LXC) [19], for performing the VNFs configuration and management. The VIM, despite being formally part of the MANO framework, is typically bundled with the NFVI and thus is not present in the OSM stack. However, the interaction with the VIM is realized through a specific driver called resource orchestrator (RO). Note that this is also common for other MANO implementations, see for example OpenBaton [25] and Tacker [44]. A set of additional integrated components enable VNF placement, policy, fault and performance management. Specifically, the PLA component explores an optimization engine which defines the placement of VNFs into the available NFVI infrastructure, e.g., subject to resource constraints, cost, and utilization. The MON module performs monitoring by collecting VNF metrics from the VIM and VCA, storing them in a time-series database (TSDB), and reporting alarms related to these metrics. Policy management is accomplished by the POL component and regards tasks such as configuring auto-scaling groups for VNFs, listening for MON alarms, and reporting scaling/alarm messages to LCM when scaling/alarm conditions are met. In addition, there is also a set of common services such as data stores, authentication, and monitoring tools which are used by other components for accomplishing their tasks. For example, Prometheus [45] realizes the TSDB which is used to scrap and store time-series data related to VNF metrics collected by the MON module. Finally, the communication among the different components is executed through a unified distributed Apache Kafka message bus for asynchronous communication [46]. Apache Kafka is a fault-tolerant message queuing system that uses a publish-subscribe model for streaming messages like a data pipeline.

Typical operations that a standard-compliant MANO is expected to perform fall into five major categories [6]: i) VNF package-related operations such as on-boarding, enabling, disabling, updating, querying, and deleting VNF packages; ii) VNF-related operations such as feasibility check, instantiation, scaling (both expansion and contraction), terminating, and fault management; iii) NS descriptor (NSD) operations such as on-boarding, disabling, enabling, updating, querying, and deleting NSDs; iv) NS-related operations such as instantiation, scaling (scale-in and scale-out), updating, and terminating NSs; and v) VNF forwarding graph (FG), i.e., VNF chaining, lifecycle operations such as creating, updating, querying, and deleting VNF FGs.

Executing the aforementioned operations requires the cooperation of multiple functional blocks of the MANO framework. For example, the VNF scaling operations envision the coordination and exchange of control flows among the NFVO, the VNFM, and also the VIM [6]. This is also reflected in the OSM architecture since similar operations involve interaction of several components. As a mere example, the automated VNF scaling procedure relies on alarms, raised from VNF and VIM collected metrics, that trigger a scaling process for which also the MON, POL and TSDB components interact with the LCM, Juju and RO modules. Henceforth, from a dependability perspective, ensuring the complete functionality of the MANO requires that all components are able to provide their services. As a result, it is reasonable to assume the OSM software as a single entity since the failure of even a single component will prevent the system from providing its agreed function(s). This assumption is (to a certain extent) also validated in the experiments reported in Section VII and used in the analysis in Section VIII.

IV. Deployment Configurations

In this section we illustrate the different deployment cases which are the focus of this study.

A. Docker Swarm deployment

Docker is a widely used container technology and an application running on Docker is constituted by a container manager (also called engine or daemon), which manages images, volumes, networks, and container instances. A container instance is build from a container image which is typically stored in an image repository. It is common that for a given image, several container instances are spawned, forming a cluster, for purposes like load balancing, high-availability or scalability.

The OSM Docker swarm installation deploys 14 docker containers running in *swarm mode* with each component having one single replica. Docker *swarm mode* is a native feature of Docker for managing and orchestrating a cluster of Docker engines forming a so called *swarm*. It entails several cluster management characteristics such as: i) decentralized configuration of cluster nodes at runtime, ii) automatic scaling, iii) automatic cluster state reconciliation, and iv) integrated load balancing. A swarm is a cluster of Docker nodes which can act as managers, who manage the swarm membership and delegate tasks, and workers which run swarm services.

A Docker node can be a manager, worker, or both. A service is the definition of the tasks that shall be executed by the swarm through either standalone managers or worker nodes. When defining a service, the optimal state of it is defined by specifying features like number of replicas, network and

Fig. 3. Illustration of *Manager* (left) and *Manager-Worker* (right) deployment configurations and experimental testbed.

storage resources attached to it, and the ports the service exposes etc. It is the responsibility of the Docker manager to maintain the swarm state in case one of the worker nodes becomes unavailable by re-scheduling its tasks to other nodes.

A *swarm* may consist in only one node, which by default will simultaneously act as a manger and worker, but it cannot be only a worker without a manager. We refer to this setup as the *Manager* configuration. To be noted that this kind of deployment does not provide sufficient protection in terms of faulty physical host and supporting software like the operating system. Therefore, though not specifically recommended by the OSM community, we consider the case where an additional node joins the swarm for acting as a manager node and the service workload is only processed in the worker node. This is also a Docker recommendation in case a limited number of physical hosts is available [47]. In this case, the swarm cluster is composed of worker and manager nodes and we refer to it as *Manager-Worker* configuration. Fig. 3 depicts the key differences between the two deployment options.

One of the key features of a swam is the automatic cluster state reconciliation. This is an important feature in terms of fault management policies. In case one of the services of the cluster is down, the swarm state changes and the manager immediately respawns the failed container/containers on other available nodes (e.g., in the Manager node in a *Manager-Worker* setup) and the service stack becomes healthy again. Moreover, also in case events such as daemon, OS, and hardware failures are experienced on the worker node, all containers are respawned in the other node and the service is recovered.

B. Kubernetes deployment

Kubernetes, also known as K8s, is a container orchestration platform, alternative to Docker swam, created by Google and currently being managed by the Cloud Native Computing Foundation [48]. It was created with orchestration in mind and is supported by a much greater community compared to Docker swarm. In Release 8, OSM has evolved into supporting Kubernetes both as the infrastructure to run OSM as well as the infrastructure to deploy Kubernetes-based network functions.

Kubernetes is specifically designed for managing clusters of containerized applications. A K8s cluster consists of a set of worker machines, called nodes, and a container orchestration layer, called control plane. A worker node hosts the pods, which are the set of running applications executing the workload, and the control plane manages the worker nodes and the pods running in them. The control plane includes four components; the frontend K8s API server `kube-apiserver`, the key-value data store `etcd`, `kube-controller-manager` process(es), and the task `scheduler`. Particularly important is the `etdc` system which is a strongly consistent and distributed key-value store for reliably storing data in a distributed system. It uses Raft consensus algorithm [49] for leader election and for ensuring that cluster internal state is consistently replicated among the members. For an N members cluster, the quorum, i.e., majority, is lost when more than $(N-1)/2$ members fail. For more details on how the Raft protocol operates, the reader may refer to [49].

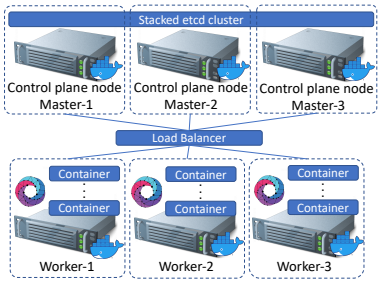


Fig. 4. Illustration of a highly available *Multi-master* cluster deployment.

A recent OSM feature is the ability to deploy OSM in a K8s highly-available (HA) cluster. In this deployment option, the OSM pods, i.e., OSM software stack components, are replicated into three distinct virtual machines running in the same physical hosts. In addition, also the control plane, called Master, is deployed in a separate machine and runs in the same host. This configuration aims at providing fault tolerance by actively running three OSM pods in a load-sharing configuration. In case any of the pods fails, the master will reschedule incoming requests on the remaining ones. However, fault tolerance is only on the OSM software level since the physical host is a single point of failure. Moreover, the failure of the the single Master would destabilize the cluster state and it would prevent the system from accepting and processing incoming requests although the pods would still be up and running.

To overcome this limitation, and driven by Kubernetes recommendations for deploying highly available clusters [50], we consider another topology, called *Multi-master cluster*, where worker and master nodes are distributed in multiple physical hosting nodes. The cluster is composed of three OSM pods which are deployed in separate physical hosts and there are also three Masters, forming the cluster control plane, with each of them also running in a separate physical node. Fig. 4 illustrates this K8s-inspired cluster topology. Each of the three Masters, hosts an `etcd` member and they together form an `etcd` cluster that enables maintaining a strongly consistent internal state and ensures that the lost of one of the members, i.e., Masters, can be tolerated. Note that only the Masters participate in the `etcd` cluster. This way, the failure of one single Master would not compromise the quorum and the cluster would still be able to elect a leader for managing the overall cluster.

V. Software Aging and Rejuvenation

Past studies of software engineering classify software faults into two main categories, Bohrbugs and Mandelbugs [37]. Bohrbugs, otherwise called deterministic, are software faults that typically can be easily reproduced since they tend to manifest themselves consistently under the same conditions. They often may lead to a software crash or process hanging and the bugs need to be identified and resolved. It is possible that accurate test and validation efforts can identify and correct this kind of bugs. Mandelbugs are bugs whose activation and error propagation are more complex in nature. They are difficult to isolate and as a result, they are hard to reproduce. Their manifestation is transient in nature and are usually caused by timing and synchronization issues resulting in race conditions. A retry operation or software restart may often resolve the issue [51].

Software aging is a well-known phenomenon associated with software systems [52]. The general characteristic of software aging is the fact that as the software execution time period increases, the associated failure intensity also increases. A successive activation of relative aging-related software faults causes software errors, which have not yet caused a software failure, to accumulate in the internal system state. It is due to this accumulation that aging-related errors may propagate to a system failure. This system state is also called the erroneous or failure probable state. It has been shown that all aging-related bugs are Mandelbugs [37], [52], hence further classifying Mandelbugs into two categories; aging-related and non-aging related Mandelbugs. Typical faults in IT software systems caused by aging effects include resource leakages, numerical errors, or data corruption accumulation.

The time to aging-related failure defines the time period from the moment of the software startup time to the observation of an aging-related failure. Its probability distribution is mostly influenced by the running lifetime period and the software workload quantity. The aging effect is not reversible without an external interventions and a proactive fault management method to deal with software aging is software rejuvenation. The rejuvenation aim is to clean up the internal system state and thus prevent the occurrence of more severe failures. Common methods of rejuvenation techniques consist of a system restart and/or reboot procedure [38]. Any rejuvenation will typically incur to some overhead, i.e., downtime due to safe restarts, but the goal is to prevent more severe crash failures that may be difficult to recover. As a result, an important problem is to optimize the rejuvenation schedule. Analytic-based models have been widely adopted to find the optimal tradeoff for a variety of software systems including virtualized servers [53]–[55], service function chains [56], and software-defined controllers [39], [57]. Common to all these efforts is the adoption of Petri-net based formalisms and the characterization of aging dynamics with the objective of identifying the optimal rejuvenation schedule such that the system SSA is maximized.

In similar lines, the scope of this work is not limited to characterizing MANO software-dependability dynamics impacted by the aging phenomenon but also assesses non-aging related faults' impact on the SSA. Henceforth, on the software level, we consider both aging and non-aging related Mandelbugs, while assuming that correct testing and validation has removed the Bohrbugs prior to deployment.

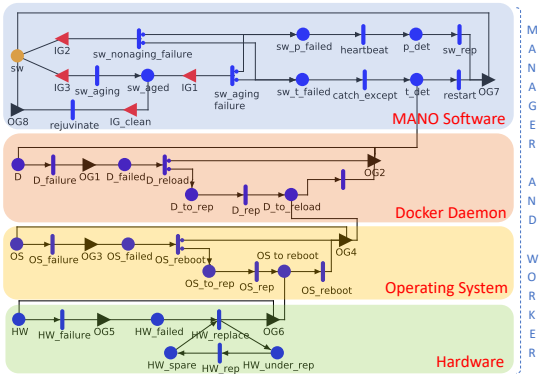
VI. Availability Models

A SAN is a modeling formalism with which detailed performance, dependability, or performability models can be implemented in a comprehensive manner [58]. SANs are stochastic extentions of Petri Nets consisting of four primitives: *places*, *activities*, *input gates*, and *output gates*. Places are graphically represented as circles and contain a certain number of tokens which represent the *marking* of the place. The marking of each place in the model represents the state of the system. Activities are actions that take a certain amount of time to fire and move tokens from one place to another. They impact the system performance and can be *timed* (thick vertical lines) or *instantaneous* (thin vertical lines). A timed activity has a distribution function associated with its duration and can have distribution case probabilities used to model uncertainty associated with activity completion. The case probabilities are graphically represented as small circles on the right of the activities. Upon completion, an activity fires and enables token movements from places connected by incoming arcs to places connected by outgoing arcs. This way a system state update occurs and tokens are moved from one place to another by redefining the places' markings. Input and output gates define marking changes that occur when an activity completes. Different from output gates, the input gates are also able to control the enabling of activity completion, i.e., firing. All the models are constructed using the Möbius software tool [59].

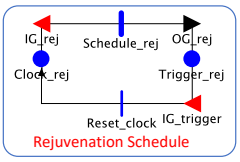
In the following, we illustrate the proposed abstraction models for the different MANO configurations.

A. Manager Configuration

Fig. 5 illustrates the SAN model of the *Manager* configuration. It abstracts the deployment of the MANO containerized software into one physical node, which acts as both manager and worker for the service tasks. Note that in the figure, we have treated the software deployment of both worker and manager together for illustration simplicity. Making the "manager" part more explicitly can be done similarly as in Fig. 6 for the *Manger-Worker* configuration. The model includes the MANO software (i.e., all MANO components), Docker daemon, OS, and hardware layers, and a similarly structured model may also apply to other containerized system. The places *D*, *OS*, and *HW* are initialized with 1 token each, indicating working Docker daemon, OS, and hardware components, respectively. The place *sw* is an extended place and allows the representation of structures or arrays. Specifically, we consider the tokens in *sw* to be a structure containing two fields, one representing the `operational units`, initialized with one token, and the other one representing the potential number of software `aging-related faults`, initialized with *N* tokens. Similarly to previous works (see [29]–[32]), it is assumed that all the timed activities follow a negative exponential distribution unless otherwise specified.



(a) SAN availability model of the MANO.



(b) Rejuvenation schedule model.

Fig. 5. SAN availability model of the *Manager* MANO configuration with software rejuvenation.

In [40] and some other studies (see for example [39], [57], [60]), software aging is modeled with a "one-shot" representation where a token is fired, following a certain distribution, from an up place to an error-prone place and the same token can be subject to a consequent firing due to a software aging-related failure. Nevertheless, this representation fails to capture the very essence of software aging, which is the continuous accumulation of software aging errors and the consequent increase of the failure rate. In this paper, we adjust this drawback by representing a more realistic aging behavior. Specifically, aging is represented through a timed activity *sw_aging*, with rate $\lambda_{\text{sw}_{\text{ag}}}$. The firing of *sw_aging* is enabled by the input gate *IG3*, which verifies that the system is operational, i.e., there is one token in the *sw* field `operational units`, and there is at least one token in the field software `aging-related faults`. For every *sw_aging* firing, there is a token removal from the N tokens, present in `aging-related faults`, and placed in *sw_aged*, which in turn represents the error-prone state. This way, the model allows the accumulation of aging errors in *sw_aged* and the *sw_aging_failure*, which represents the aging failure event, is directly proportional to the number of accumulated tokens in *sw_aged*. This way, the more accumulated aging errors, the higher is the failure intensity due to aging.

For the non aging-related Mandelbugs, the timed activity *sw_nonaging_failure* represents the non-aging related software failure event with rate $\lambda_{\text{sw-fail}_{\text{nag}}}$. When *sw_nonaging_failure* fires, the token representing the operational unit is removed from the place *sw* indicating that a MANO software failure has been experienced and the system is in a failed state.

For both software failure events, we differentiate between two types of failures based on their recovery process. We make use of case probabilities associated to the timed activities where C_{nag} defines the probability that a non-aging related failure event is recovered with a software restart and with probability $1 - C_{\text{nag}}$, the failure recovery requires a manual intervention for executing a software repair. Similarly, C_{ag} defines the probability that an aging related failure is recovered with a software restart and with $1-C_{\text{ag}}$ with a software repair.

Once a software failure is experienced, a token is placed in either *sw_p_failed* or *sw_t_failed*, which define the recovery process that the software will undergo. *heartbeat* and *catch-exception* symbolize the detection of failures and are defined with deterministic times μ_h and μ_c. *sw_rep* and *restart* represent the repair (including any eventual reboot or upgrade of software) and restart events of the software with rate $\mu_{\text{sw}_{\text{rep}}}$ and $\mu_{\text{sw}_{\text{res}}}$, respectively. On the docker engine level, i.e., daemon, *D_failure* and *D_restart* model the failure and recovery events of the daemon with rates λ_{D} and $\mu_{\text{D}_{\text{r}}}$, respectively. The recovery entails a daemon restart where with probability C_{D} a daemon restart recovers the failure and with $1-C_{\text{D}}$ a hard repair is needed. The latter is defined through the activity *D_rep* with rate $\mu_{\text{D}_{\text{rep}}}$. Once the daemon is repaired, an additional restart is performed to fully recover it. Similarly to the daemon, the operating system level is modeled with the same dynamics having specific failure and repair parameters which we introduce in Section VIII. On the hardware level, *HW_failure* and *HW_replace* represent the failure and recovery with rates λ_{HW} and $\mu_{\text{HW}_{\text{rep}}}$, respectively. The place *HW_spare* indicates the spare hardware equipment used to replace the failed hardware and is initialized with 1 token.

A novel contribution compared to our earlier work [40] is the adoption of software rejuvenation, as a proactive software maintenance mechanism. We apply a time-based rejuvenation where in specific time intervals, called rejuvenation intervals, the system undergoes a graceful software restart. To model this mechanism, we introduce a model (Fig. 10(a)) that defines the rejuvenation scheduling, and an additional timed activity *rejuvenate* models the time it takes the system to restart. More specifically, the place *Clock_rej* holds one token and the deterministic time activity *Schedule_rej*, which defines the rejuvenation interval, upon firing moves the token from *Clock_rej* to *Trigger_rej*, where the latter represents the state that the rejuvenation can be triggered. This movement is enabled by the *IG_rej* port which verifies that the system is operational, there is at least one token in *sw_aged*, and there is one token in *Trigger_rej*. If these conditions are satisfied, the rejuvenation is performed and *rejuvenate* fires a token. At the same time, *IG_clean* removes all the accumulated tokens in *sw_aged* by setting them to zero and sets the `operational units` field in *sw* to zero, indicating that the system is undergoing a downtime due to rejuvenation. The *Schedule_rej* and *rejuvenate* activities are defined with deterministic times μ_{Sched} and μ_{rej}, respectively. Once the rejuvenation is completed, the token is moved from *Trigger_rej* and placed into *Clock_rej* by the firing of the instantaneous activity *Reset_clock*, and the output gate *OG*8 resets the *sw* fields `operational units` and `aging-related faults` equal to one and N tokens, respectively. The output gate *OG*7

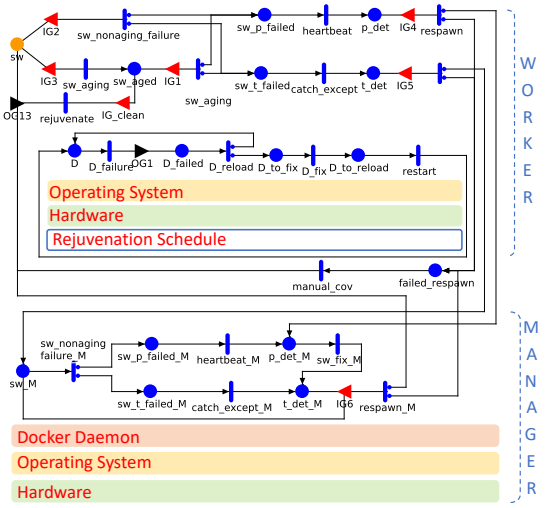


Fig. 6. SAN availability model of the MANO deployed in a *Manager-Worker* configuration.

operates similarly to *OG*8 except that in this case the system has gone through a software recovery procedure. Note that rejuvenation can be performed only when it is scheduled to happen and the system is operational.

Finally, the following output gates define the token marking movements among lower-level places: *OG1/OG3/OG5* manage the failure events of the daemon, OS, and hardware levels, respectively. When their related timed activities fire, connected to their incoming arcs, the output gate places one token in the respective failed position and sets to zero the upper-level places. This is because a failure of the physical hardware will cause a failure of the OS which in turn impacts the operational state of the daemon and MANO software as well; *OG2/OG4/OG6* places 1 token in their relative working place, i.e., *D/OS/HW*, and the relative upper-level places to which they are connected by outgoing arcs. For example, a recovery from a daemon failure brings the daemon in the up state but requires a restart of the MANO software for a fully working system. The system is fully operational when the `operational units` field of *sw* place holds one token.

B. Manager-Worker Configuration

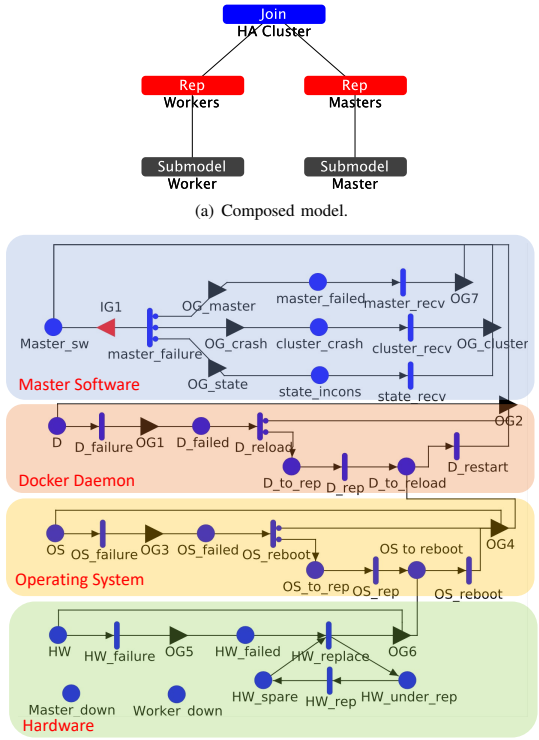
The *Manager-Worker* configuration consists of two separate nodes forming a cluster and the OSM stack is deployed on the worker node, with the latter being responsible for workload processing. Fig. 6 depicts the *Manager-Worker* SAN model. To distinguish the models of the two entities, we add a suffix *_M* for all the places and activities regarding the manager part. The system is fully working if there is a token in either of the *sw*, *sw_aged*, or *sw_M* places.

On the worker node, the MANO software component is similar to the *Manager* configuration except for the recovery phase where once a failure is detected, the containers running the software are respawned, through the timed activity *respawn*, in the manager node. We distinguish two cases: when a software repair is needed, the token is moved from *p_det* of the worker node to *p_det_M* of the manager. In the other case, the token is moved from *t_det* to *sw_M* indicating that a respawn, i.e., container restart, is sufficient to recover the system. However, for both cases, we consider the eventuality of a respawn process that fails. To this end, we consider two case probabilities associated with the timed activities. With probability C_{respawn}, the container respawn is successful and $1 - C_{\text{respawn}}$ it fails. In the latter, there is a need for a manual coverage, represented by *manual_cov*, and the token is placed back in place *sw*. In order for the respawn to instantiate, the hosting manager node needs to be operational and this is controlled by the enabling gates *IG1/IG2* which enable the respawn only if the daemon, OS and hardware of the manager are working, i.e., their respective places *D_M*, *OS_M*, and *HW_M* contain each 1 token. In addition, differently to the *Manager* setup, once the daemon fails, there is just the recovery of the daemon since the MANO software is immediately respawned in the manager node. The rest of the model is similar to the *Manager* configuration and due to space limitations we use colored bars with component names to indicate the relative parts of the model and omit illustrating.

On the manager node, once a token is deposited in *sw_M*, the system is again operational. While the software is running in this node, we assume that it is subject to only non-aging software related failures. This is because *swarm mode* best practices suggest that the worker node should be the dedicated node for handling task requests in a 'normal' condition. Therefore, we limit the hosting of the MANO software to the manager node only for the period the worker node is failed. To this end, the input gate *IG6* enables a respawn of the software containers from the manager node to the worker node once the worker node is up and running again and ready to accommodate the containers. As a result, the manager node will host the containers for a relatively short time compared to the software aging time, hence making the assumption of only non-aging failure events on the manager node a reasonable assumption. The rest of the manager components, i.e., daemon, OS, and hardware are similar to the *Manager* configuration which for lack of space have been represented through colored bars, hence we omit further illustrating.

C. Multi-master Cluster Configuration

For abstracting the Multi-master cluster system, we exploit a Rep/Join model composition formalism which is integrated in Möbius. The formalism exploits system symmetries and generates lumped state spaces which are smaller compared to systems that do not exploit symmetries. This is particularly useful for large systems whose model nets generate complex stochastic processes [61]. The formalism enables the composition of a model in the form of a tree, where each leaf node represents a system submodel and each non-leaf node can be a Join or Replicate node. A Join node is a state-sharing node used to compose two or more submodels, whereas a Replicate node is used to compose a model consisting of a number of identical submodel replicas and can also enable



(a) Composed model.

(b) Master SAN model.

Fig. 7. SAN availability model of the MANO deployed in the *Multi-master* cluster with software rejuvenation.

state-sharing among its replicated submodels. The replicated submodels behave independently of each other and the root node represents the complete cluster model.

The *Multi-master* cluster we consider is not part of a deployment option or an enhancement feature of OSM and is primarily driven by Kubernetes recommendations for the deployment of 'truly' highly available clusters [50]. For the scope of our investigation, we make some reasonable assumptions that limit the system complexity, yet do not impact system performance, as they can be deployment options that an operator can arbitrarily choose. First, we assume that the cluster components fail independently. This can be a reasonable assumption in case components are geographically distributed; therefore, minimizing the likelihood that events can simultaneously affect two or more nodes. In addition, we assume that the load balancer is failure free and uniformly distributes the workload among nodes (refer to Fig. 4). Moreover, we also consider that the OSM pods are not deployed in virtual machines but directly on standard hardware running an operating system. We also assume that each worker node of the cluster runs a Docker runtime engine, i.e. daemon.

The cluster is modeled through the Rep/Join formalism by replicating three times both the Master, i.e., control plane, and the Worker submodels, as illustrated in Fig. 7(a).

The Worker submodel is similar to the *Manager* configuration model except for the presence of two shared places called *Worker_down* and *Master_down*. These two places are also present in the Master submodel and are used to keep track of the availability of Workers and Masters for the overall composed model, i.e., the *Multi-master* cluster. Every time a Worker or Master fails, a token is placed in the respective place and removed when they are recovered.

The Master submodel, illustrated in Fig. 7(b), is similar to the previous configurations on the hardware, OS, and Docker daemon levels. On the software level, we consider failure events that can affect either singularly the Master nodes or the overall cluster. Several studies have shown that distributed applications experience a variety of issues due to their distributed implementation. Some of the most typical issues that can cause cluster-wide failures concern state inconsistencies, leader election, defective fault management, or scalability issues [39], [62]–[64]. We account for these failure modes by assuming that each of the Master replicas may experience software failures (e.g., failures of the API server, scheduler, or etcd members) causing a single replica failure, cluster-wide crash, or cluster state inconsistencies. We use state distributions to characterize these events with probability C_{master}, C_{crash}, and $1-C_{\text{master}}-C_{\text{crash}}$, respectively. The Master software fails with rate λ_{Master} (transition *master_failure*), and this event is enabled through the input gate *IG1* only if less than three Masters are down, i.e. less than three tokens in the shared place *Master_down*. In case a cluster-wide crash or state inconsistency is observed, the respective output gates *OG_crash* and *OG_state* place three tokens in *Master_down*. On the recovery of such failures, the gate *OG_cluster* removes three tokens from *Master_down* and places them in *Master_sw* to indicate that a cluster-wide failure has been recovered.

The overall *Multi-master* cluster is considered unavailable when three tokens are present in place *Worker_down* and more than one token is present in place *Master_down*. Note that also other failure/recovery events on the other components (daemon, OS, and hardware) for both submodels place/remove one token in *Master_down* or *Worker_down* depending on the submodel.

D. Component-wise MANO Model

The approach taken so far in this paper is to consider the MANO software as a single component on the software level. On the one hand, decomposing the MANO software model into specific components would allow characterizing the various components in a finer grain in regard to their failure/repair dynamics. This can be of particular interest for cases where some software components are developed, tested, and validated by 'external' developing teams which may follow different practices, as is the case of the Juju VCA component which is developed and maintained by Canonical rather than the OSM community. Nevertheless, abstracting realistic MANO solutions is still subject to the actual architecture since the various solutions significantly differ in terms of architecture and implementations [65]. To illustrate, modeling the OSM software would require abstracting 14 software components,

and even more for ONAP because it comprises 20 functional modules [65]. As a result, it is hard to employ a generalized model which is capable of a fine grain modeling of realistic implementations.

On the other hand, at a high level, all solutions should adhere to the ETSI standards, where the three main functional blocks, i.e, NFVO, VNFM, and VIM, must be part of a compliant architecture. This requirement can be reflected in a functionality-wise generalized model. This modeling approach could be suitable in cases where failure/repair dynamics of individual components differ significantly, though the lack of detailed studies in this matter, and ultimately failure and repair parameters of individual components, may discourage the pursue of this modeling approach. In the rest of this section, such a component-wise modeling attempt is introduced. In the next section, we also introduce experimental trials to retrieve key parameters regarding recovery times of individual components which can be used for a preliminary investigation.

Fig. 8 depicts the adopted model of a high-level MANO with separate components. In particular, the model includes separate NFVO, VNFM, and VIM software elements, deployed in the same hosting node, and their relative rejuvenation policies. The same software layer model utilized in the *Manager* configuration is used to abstract each of the components and due to lack of space they are represented through colored bars instead of SAN primitives. The three components are assumed to fail independently and the failure of just one of them would lead to an unavailable MANO. This assumption is according to the expectations of an ETSI-standardized NFV-MANO system as all main functional blocks are expected to be fully operational in order to be able to orchestrate and manage NFV services [16]. Due to the lack of failure data regarding MANO solutions, let alone single software components, we assume that each of the components is characterized by similar failure times which together exemplify the total intensity of the MANO software adopted in the other models. Regarding the repair process, we retrieved individual recovery parameters through experimental trials on the OSM solution by injecting faults on the software level targeting individual components, i.e., LCM, Juju VCA, and RO. On the Docker daemon, OS, and hardware level, the same submodels utilized in the *Manager* configuration are also used here, and the failure of any of these levels demands a restart of each of the MANO components once the level is restored such that the system can be deemed operational.

The rejuvenation process is separate for each of the three components and is subject to the individual utilization and software aging rates. For example, an operator could reduce the rejuvenation frequency for less utilized components and vice-versa. However, for simplicity, and also due to lack of knowledge regarding individual failure characteristics, we assume that the same rejuvenation process governs the individual rejuvenation policies. This can also be beneficial since a fully synchronized rejuvenation process will lead to the minimum downtime overhead introduced by rejuvenation. In addition, the rejuvenation duration is equal to the highest amount of time required to restart the single components. The model is solved by feeding the individual recovery times of the LCM,

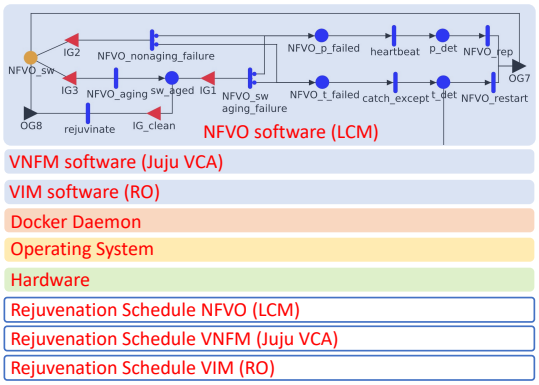


Fig. 8. SAN availability model of the MANO with separate software components.

Juju VCA, and RO components, while maintaining the total failure intensities. We compare the two approaches in terms of SSA and perform a sensitivity analysis on the impact that variations of the rejuvenation interval, software aging, and software-aging induced failures have on the SSA.

VII. Experimental Testbed

Model-driven availability assessment relies on model parameters regarding failure and repair processes. However, the lack of failure and repair data is a common issue for novel technologies and projects. To partially tackle this issue, we performed an experimental campaign aiming at retrieving realistic recovery times of the system components by adopting fault-injection techniques. Our testbed consists of hardware and software technologies that are commonly used in cloud computing infrastructures in which the OSM software stack is deployed from scratch. Specifically, OSM Release 8 is deployed in *swarm mode* option, i.e., with Docker swarm orchestrator, into a Linux-based operating system (server version with kernel 5.15) with Docker engine (version 20.10.5) running on a 56-core Intel® Xeon® @ 1.70GHz machine with 128GB RAM, two 10-Gbps and two 1-Gbps Intel Ethernet NICs, and four 1-TB SATA hard drives. In this deployment option, the single machine will act as both manager and worker node, i.e., *Manager* deployment.

In order to perform measurements of the *Manager-Worker* deployment in case the worker node experiences failure on the host level, namely respawn times, we join to the OSM swarm deployment another host machine by using the standard `docker swarm join` command. The latter node is equipped with the same software and hardware technologies of the previous one and acts as a worker node. The host machines are connected to each other by their 10-Gbps NICs through a 5-Gbps Ethernet network switch and the OSM swarm is deployed in the worker node. Fig. 3 depicts the testbed adopted for the experimental campaign. This way, we emulate the two Docker swarm deployments and the testbeds are ready for fault injections on the different system components.

Starting with the *Manager* deployment, we inject the following fault types:

Software faults: responsible for software crashes and process hanging of the OSM software layer. Such faults can be varied in terms of manifestation nature including time and synchronization issues resulting in race conditions, resource leakage due to software aging errors, and error handling faults [37], [51]. Several of these software faults are also reported in the Bugzilla bug tracker platform utilized by the OSM community [66]. To emulate the occurrence of these faults, we forcefully terminate each of the OSM containers, and measure the time it takes for the stack to return in a running state. Precisely, we kill all containers of the stack and continuously (every second) interrogate each of the tasks, i.e., containers, until they reach a running state (the `.CurrentState` of the task). The interval between the time the fault is injected and the time the last task is running defines the overall time that will parametrize the mean time to perform an OSM software restart (i.e., the μ_{restart} activity on the model).

Docker engine faults: similarly to OSM containers, also the Docker engine can be subject to software faults. [67] reports faults affecting the Docker engine caused by software aging phenomenon. This component is particularly critical as a failure of the daemon causes the simultaneous failure of all running containers, networks, and mounted volumes. We mirror the fault on this layer by abruptly halting the container management process, i.e., `dockerd` process, and record the time it takes to restart, i.e., be running again. The measurements will define the rate of the *D_restart* activity.

Operating system faults: also the operating system is affected by software faults and several studies present recurring faults including OS exceptions, error codes, OS panics, or hangs [68]–[70]. Needless to say, the failure of the OS results in the termination of all the software layers running on top. To mimic this type of faults we force an immediate OS reboot without terminating any process or unmounting any file systems, i.e., hard reboot. The experiment executes the reboot command and records the time the command is issued. Upon system boot, we retrieve the time it takes for the kernel to reach the default runlevel (5 in the machines) and compute the time difference. The assessment will determine the mean time to perform an OS reboot, defined as μ_{OS_r} in the models.

Swarm node faults: these are faults that trigger a respawn of the containers in another node in case events such as daemon, OS, and hardware failures are experienced on the node that hosts the swarm services. We emulate this kind of faults by using standard docker commands that drain the availability of the node to host the containers and this triggers the automatic re-instantiation of the whole stack into another node. Specifically, we run `docker node update --availability drain <NODE-ID>` on the *Worker* node, which disables the *Worker* ability to host swarm tasks, and measure the time it takes for all containers to reach a running state in the *Manager* node.

The considered form of injection is focused on failure modes as effect of faults occurring in the components that can affect the system. Although from a terminology viewpoint, this form of injection in some cases is referred to as *error/failure injection*, it is also common to refer to this form as *fault injection* since failures of a component can be regarded as faults from the perspective of the system that incorporates the component [71].

We performed 50 controlled experiments for each fault type, resulting in 200 experiments in total. For each fault type, we develop ad-hoc shell scripts that inject the fault, trigger the recovery and measure the recovery time, wait for a reasonable amount of time such that the targeted system reaches a stable state, and re-run the fault injection. It is worth noting that we consider these kinds of faults as events that cause a soft failure of the targeted system for which a restart/reboot of the system is sufficient to recover it. In addition, we also performed 50 fault-injection trials individually on three software components; the LCM module, the Juju VCA, and the RO component. These individual mean recovery times are used in the assessment of the *Component-wise* model.

While running the experimental trials, we made several observations. At first, through an inspection of each of the containers, using `docker inspect` command, we observe that, while each of the containers is created within seconds from the fault injection time, the times for them to reach a running state significantly differ from each other. Some tasks reach a running state within a few seconds, e.g., the Database and the AUTH components, while others require a few tens of seconds, e.g., the RO, POL, and TSDB. Other components require even a few minutes to reach a running state, hence clearly showing a significant difference compared to recovery times reported in studies regarding containerized applications (i.e., recovery within hundreds of milliseconds) [35], [36].

Secondly, we observe a consistent behavior when inspecting the faults that cause a restart of the whole OSM stack, e.g., *Docker engine faults*. The LCM and the MON containers are always the last to reach a running state, with LCM reaching the desired state before MON. Although they are started multiple times, they fail to reach a running state until the rest of the components are running. Such observation is different when the single components of LCM, RO, and Juju are restarted individually. The times in these cases are smaller, refer to Table I, and consequently lead to an intuition that there should be some software dependencies among the components such that only when other containers are running, and consequently exposing services, others may reach a running state. However we are not able to identify the level of dependency for each of the running containers without a detailed knowledge of the software architectural design. This observation further supports the consideration that treating the OSM stack as a single entity may be more reasonable than treating its individual elements separately. Finally, during the *Swarm node* fault-injection measurements we observed that upon the node availability draining, all the containers were quickly respawned in the other node except Grafana and Prometheus. This behavior led to swarm instability, and hence we applied a workaround by quickly rolling back the node availability so that these two components can be restarted in the same node. This is likely due to some dependency among these components and the host node where they are initially launched. Clearly this does not represent the considered scenario, i.e., *Manager-Worker*, but we assume that their respawn times are similar, although respawned in the same node. We measured the respawn times similarly to the *Manager* case by adopting the workaround.

The mean recovery times described above, together with the relative standard deviation, are reported in Table I. We notice that some of the components such as the Juju, the RO and the Docker daemon have a rather fair stability in their mean time to recover since they present a limited spread of time values. As expected, the restart of the OSM software on both the same or another node, i.e, *Swarm node* faults, presents very similar values. This is because the services are managed by the swarm and spawning containers in another node, with the same processing capability, involves the same process, i.e., the docker engine spins up the same tasks using the already pulled container images.

VIII. Numerical Analysis

In this section, we present a numerical (evaluation) study. The proposed models are defined in the Möbius software tool [59] and they are solved using discrete-event simulation, integrated in the tool, with 99% confidence interval and 10^{-5} width of relative confidence interval. The SAN model parameters are in part retrieved from previous literature [39], [57], in part from experimental measurements, and the rest are estimated guesses based on our empirical experience. They are illustrated in Table I, and they represent the baseline parameters.

A. Sensitivity Analysis

Given the baseline parameters, the achieved MANO availability for every model is presented in Table II, together with the relative availability when an optimized rejuvenation policy is applied. We observe that for all deployments there is a meaningful improvement in terms of downtime reduction when an optimal rejuvenation policy is applied. The gain is more pronounced for the *Manager-Worker* and *Multi-master* case studies, achieving a 39% and 61% of downtime reduction relative to the system downtime without rejuvenation.

The sensitivity analysis is performed by varying failure and recovery parameters with one order of magnitude, i.e., $\times 10$ and $\times 10^{-1}$, from their baseline values, and retrieving the SSA in case no rejuvenation is employed. The sensitivity to these parameters, for all the case studies, separated into failure and recovery events, is presented in Fig. 9. We have adopted a modified logarithmic scale on the availability axis in order to obtain a better visualization of the high availability numbers.

For the *Manager* case, the most impactful failure parameters are software non-aging failure rate followed by hardware, software aging, and software aging failure rate. In particular, reducing the software non-aging related failure rate decreases the availability to 0.9913. Among these failure parameters, software aging rate brings the highest improvement on the SSA, reaching 0.9984. Concerning recovery parameters, software repair, followed by the hardware replace rate, has the highest impact on the system availability by reducing it from 0.99723 to almost 0.989. At the same time, the highest improvement, reaching 0.99932, is achieved for a software repair rate increase, i.e. lower software repair time.

The same analysis for the *Manager-Worker* deployment reports a considerable reduction of the critical parameters.

TABLE I
Availability model parameters
(°from experiments, ‡from literature [39], [57]).

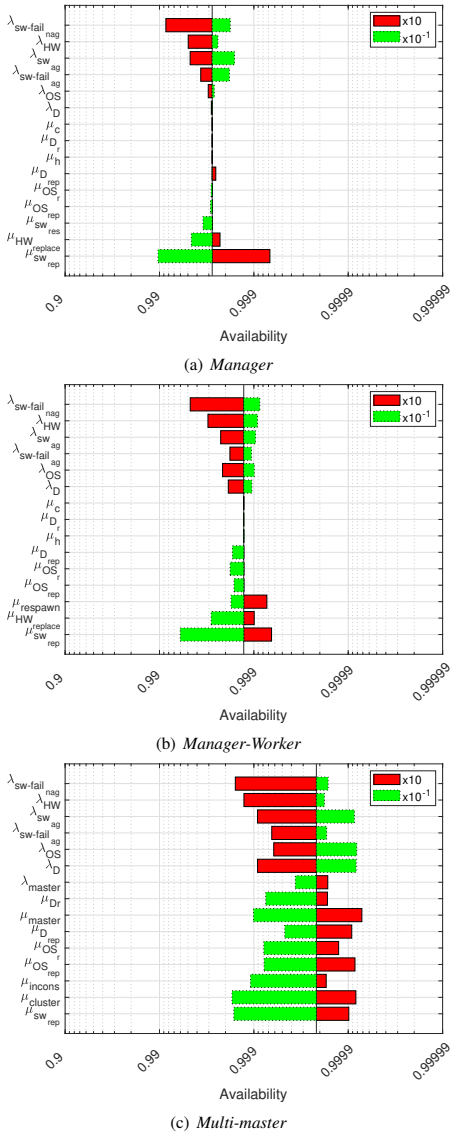
Intensity	Time	Description [Mean time to]
$\lambda_{\text{sw}_{\text{ag}}}^{-1} = 1$	week	MANO software aging‡
$\lambda_{\text{sw}-\text{fail}_{\text{ag}}}^{-1} = 3$	days	next MANO software failure after aging‡
$\lambda_{\text{sw}-\text{fail}_{\text{nag}}}^{-1} = 1$	month	next MANO non-aging software failure‡
$\mu_{\text{sw}_{\text{rep}}}^{-1} = 1$	hour	MANO software repair‡
$\mu_{\text{sw}_{\text{res}}}^{-1} = 185\ (\pm 15.6)$	seconds	MANO software restart (OSM stack)°
$\mu_{\text{NFVO}_{\text{res}}}^{-1} = 32\ (\pm 3.1)$	seconds	NFVO container restart (LCM)°
$\mu_{\text{VNFM}_{\text{res}}}^{-1} = 8.5\ (\pm 0.6)$	seconds	VNFM container restart (Juju VCA)°
$\mu_{\text{VIM}_{\text{res}}}^{-1} = 19.5\ (\pm 0.7)$	seconds	VIM driver container restart (RO)°
$\mu_{\text{h}}^{-1} = 10$	seconds	heartbeat*‡
$\mu_{\text{c}}^{-10} = 1$	millisecond	catch exception*‡
$\lambda_{\text{D}}^{-1} = 4$	months	next daemon failure‡
$\mu_{\text{D}_{\text{rep}}}^{-1} = 1$	hour	daemon repair‡
$\mu_{\text{D}_{\text{r}}}^{-1} = 30\ (\pm 1.8)$	seconds	Docker daemon restart°
$\lambda_{\text{OS}}^{-1} = 4$	months	next OS failure‡
$\mu_{\text{OS}_{\text{rep}}}^{-1} = 1$	hour	OS repair‡
$\mu_{\text{OS}_{\text{r}}}^{-1} = 249\ (\pm 21.4)$	seconds	OS reboot°
$\lambda_{\text{HW}}^{-1} = 6$	months	next hardware failure‡
$\mu_{\text{HW}_{\text{rep}}}^{-1} = 24$	hours	hardware repair‡
$\mu_{\text{HW}_{\text{replace}}}^{-1} = 1$	hour	hardware replace‡
$\mu_{\text{rej}}^{-1} = 3$	minutes	rejuvenation duration°
$C_{\text{nag}} = 0.3$		prob. for non-aging transient failures‡
$C_{\text{ag}} = 0.7$		prob. for aging transient failures‡
$C_{\text{D}} = 0.9$		daemon restart coverage factor‡
$C_{\text{OS}} = 0.9$		OS reboot coverage factor‡
$N_{\text{spare}} = 1$		Number of spare hardware‡
$N = 60$		Number of potential software aging faults‡
$\mu_{\text{respawn}}^{-1} = 189\ (\pm 21.6)$	seconds	respawn MANO software containers°
$\mu_{\text{cov}}^{-1} = 1$	hour	manual coverage
$C_{\text{respawn}} = 0.9$		respawn coverage factor‡
$\lambda_{\text{Master}}^{-1} = 4$	months	next master failure
$\mu_{\text{master}}^{-1} = 5$	minutes	recover master failure
$\mu_{\text{cluster}}^{-1} = 15$	minutes	recover cluster crash
$\mu_{\text{incons}}^{-1} = 3$	minutes	recover state inconsistencies‡
$C_{\text{master}} = 0.55$		probability of Master software failure‡
$C_{\text{crash}} = 0.05$		probability of cluster-wide crash‡

*Deterministic time

TABLE II
Steady-state availability of the different models without and with optimal rejuvenation policy.

	Manager	*Manager-Worker*	*Multi-master*	*Component-wise*
MANO w/o rej.	0.99723	0.99871	0.999782	0.99723
MANO opt. rej.	0.99762	0.999215	0.999915	0.99794
Downtime reduction	14%	39%	61%	25%

Besides the overall availability gain introduced by the fault-tolerance on the host level (refer to Section VI-B), the negative impacts of both software non-aging failure rate and software repair rate are markedly reduced compared to the *Manager* case. To illustrate, for the *Manager* case, a ten-fold increase of software non-aging related failure rate decreases the SSA from 0.99723 to 0.99143, which corresponds to an increase of 50.84 hours of yearly downtime (from 24.28 to 75.12). For the same parameter reduction, the *Manager-Worker* SSA is reduced from 0.99871 to 0.99526 corresponding to a 30.76 hours of additional yearly downtime. However, there is a more evident impact of parameters related to the physical host. We notice that an increase of either the OS or daemon failure rate has a more pronounced effect on availability reduction. This is because more frequent failures of the Manager OS or Docker daemon would prevent the Manager from hosting the MANO software in case a failure is observed on the Worker side.



(a) *Manager*

(b) *Manager-Worker*

(c) *Multi-master*

Fig. 9. Sensitivity analysis for the different MANO deployments without rejuvenation.

Moreover, a higher respawn rate (μ_{respawn}) can considerably improve the SSA up to 0.99937.

The *Multi-master* configuration increases further the system availability, reaching 0.999782. However, the same critical parameters identified in previous case studies continue to be critical. In particular, similarly to the *Manager-Worker* setup, deterioration of host level failure intensities has a non-negligible impact. This is because such failures influence more the availability of the Master nodes compared to the Worker nodes due to the fact that failure of more than one Master limits the overall cluster availability, as opposed to the Worker nodes where failure of all the three replicas is needed to cause a service outage. On the other hand, a ten-fold improvement, whether failure rate reduction or recovery rate increase, brings more substantial benefits in the SSA. For several failure and recovery parameters, SSA values exceed four nines availability, i.e., less than 52 minutes of yearly downtime. Observing individual Master parameters, we notice that the recovery times of events that can cause a cluster-wide failure such as μ_{cluster} and μ_{incons} greatly impact the system SSA. In particular, a ten-fold change of the time to recover cluster failures can affect the SSA significantly. This is because recovering a cluster crash requires a larger amount of time compared to the events that cause inconsistent states. On the other hand, a ten-fold reduction in μ_{incons} also causes a comparable reduction which can be explained by the higher frequency that such events occur. These findings confirm past model-based assessments of distributed control-plane implementations which report the impact of cluster-wide failures [39].

B. Software Rejuvenation Impact

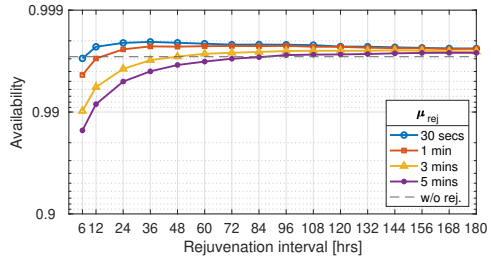
It is obvious that applying frequent rejuvenation does prevent the accumulation of aging errors, yet a frequent maintenance may lead to useless downtime caused by software restart. In order to fully profit from rejuvenation, an operator needs to find a balance between the deliberate downtime and the avoiding of more severe outages due to the error accumulation. Therefore, an operator should determine the optimal policy for scheduling the rejuvenation process.

The impact of different rejuvenation policies, i.e., μ_{Sched} and μ_{rej}, for all the cases under study are illustrated in Fig. 10 and Table III summarizes the optimal values. We use the same modified logarithmic scale to better illustrate the computed results. As expected, and common to all models, the results show that that enabling a shorter rejuvenation duration, brings significant benefits in availability. This is more evident when early rejuvenation schedules are applied.

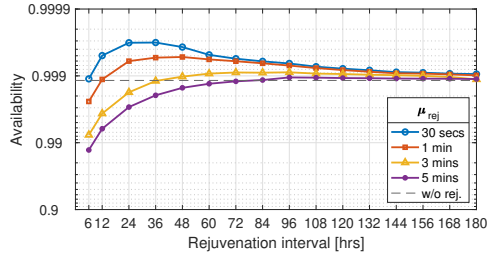
TABLE III
STEADY-STATE AVAILABILITY WITH OPTIMAL REJUVENATION POLICY WHEN VARYING REJUVENATION DURATION.

Rejuvenation duration	**Availability [Optimal rejuvenation interval in hrs]**			
	Manager	***Manager-Worker***	***Multi-master***	***Component-wise***
30 secs	0.99795 [36]	0.999678 [36]	0.9999362 [48]	0.99794 [48]
1 min	0.99785 [72]	0.999428 [48]	0.9999320 [60]	0.99772 [72]
3 mins	0.99762 [132]	0.999215 [96]	0.999915 [72]	0.99715 [180]
5 mins	0.99737 [168]	0.999144 [108]	0.999894 [84]	0.99705 [180]

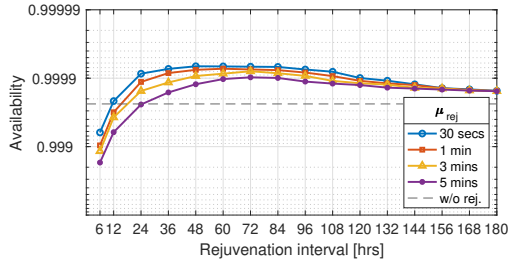
Concerning the *Manager* scenario for baseline parameters, i.e., a rejuvenation duration of 3 minutes, the maximum achievable availability is 0.99762 with an optimal rejuvenation interval of 132 hours. In case the software maintenance lasts longer, i.e., 5 minutes, the optimal rejuvenation trigger time is 168 hours, yet the availability gain is almost negligible compared to the non rejuvenated case. A similar trend is observed for the *Manager-Worker* implementation. The maximum availability that can be achieved with baseline parameters is 0.999215 for a maintenance interval of 96 hours, i.e., a safe software restart every four days.



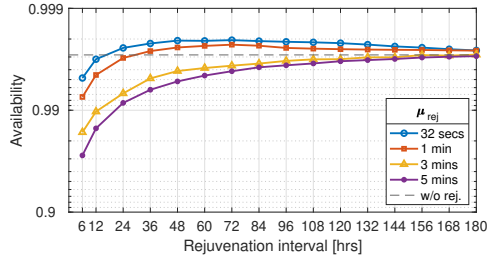
(a) *Manager*



(b) *Manager-Worker*



(c) *Multi-master*



(d) *Component-wise*

Fig. 10. Impact of rejuvenation policies on system availability for the different MANO deployments and for a varying rejuvenation duration.

In the *Multi-master* case, the overall availabilities are much higher and the maximum availability is achieved with a software restart every two days reaching 0.999915 with baseline parameters. Different to the other models, for short rejuvenation intervals, the difference between the rejuvenation durations (μ_{rej}) is less pronounced. This is because *Multi-master* entails a load-sharing cluster composed of three replicas of the MANO software which provides adequate protection even in cases where rejuvenation duration takes longer, i.e., longer downtime of one replica due to rejuvenation.

TABLE IV
STEADY-STATE AVAILABILITY WITH OPTIMAL REJUVENATION POLICY WHEN VARYING SOFTWARE AGING RATE.

Mean time to software aging	**Availability [Optimal rejuvenation interval in hrs]**			
	Manager	***Manager-Worker***	***Multi-master***	***Component-wise***
1 day	0.99665 [24]	0.99840 [36]	0.999626 [48]	0.99670 [36]
3 days	0.99743 [36]	0.99890 [48]	0.999816 [60]	0.99753 [36]
7 days	0.99762 [132]	0.999215 [96]	0.999915 [72]	0.99795 [132]
10 days	0.99791 [144]	0.999439 [108]	0.999956 [84]	0.99815 [156]

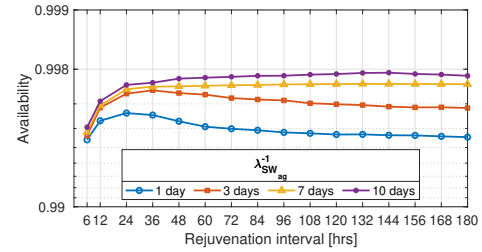
The results of the *Component-wise* model analysis show system performances that are much like the *Manager* model where for the baseline parameters the maximum achievable SSA differ at most $3.2 \cdot 10^{-4}$ compared to the *Manager* representation (0.99762 vs. 0.99794). Note that the rejuvenation schedules of the individual components are fully synchronized and the baseline duration equals 32 seconds, which is the highest amount of time required to restart the single components, i.e., LCM.

C. Software Aging Impact

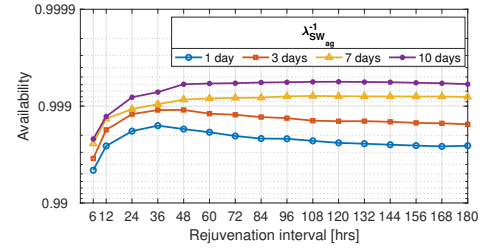
Software aging is an unpredictable parameter since it depends on several factors that may be out of developer's control such as software utilization rate, i.e., system load, operational profile and infrastructure, or software implementation. However, it has been shown that under high system workload, the software aging rate tends to increase, hence more aging errors are accumulated [67], [72], [73]. Consequently, the aging-related failure intensity increases.

We carry out a numerical analysis for a varying rejuvenation interval and assuming four software aging intensities representing high, medium, moderate, and low software utilization rates, i.e., 1, 3, 7 and 10 days mean time to software aging intensity. Fig. 11 illustrates the results for all the models. In addition, Table IV highlights the maximum availability figures for each of the deployments. Results reveal that for low to moderate software utilization, i.e., 7-10 days, the availability figures are closer compared to medium-high utilization. Such tendency is more evident as the rejuvenation interval decreases. Moreover, the difference between medium and high utilization is decreased when more fault-tolerance is introduced in the system, i.e., comparing the different MANO deployments. Again, the *Component-wise* system representation exhibits results almost identical to the *Manager*. For baseline parameters, the maximum SSA difference between the two models is within $3.3 \cdot 10^{-4}$ (0.99762 vs. 0.99795). Finally, for each of the case studies, the highest uptime gain is achieved for high software utilization indicating that a system under high workload can benefit more from rejuvenation.

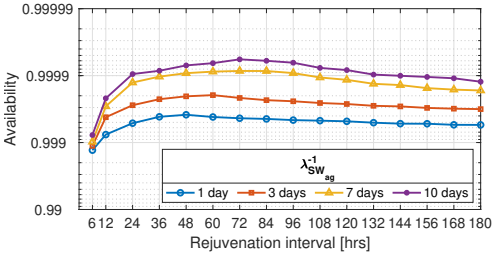
The sensitivity analysis revealed that aging failure rate may have a considerable impact on the availability of the MANO. We explore a range of software aging parameters by varying the aging rate and the aging failure rate between 1 and 10 days. Fig. 12 depicts the MANO availability for different parameter combinations, for each of the studied cases. In the *Manager* deployment case, it can be seen that the impact of the aging failure rates greatly depends on the



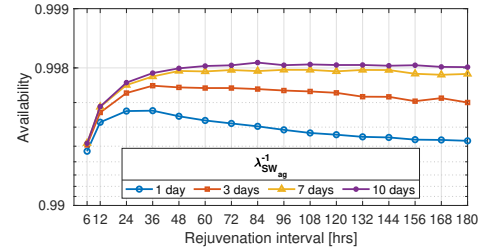
(a) *Manager*



(b) *Manager-Worker*

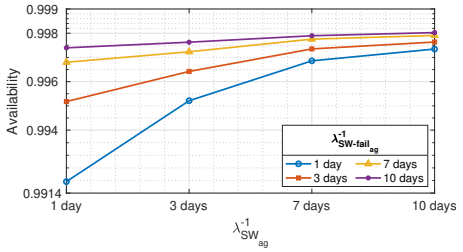


(c) *Multi-master*

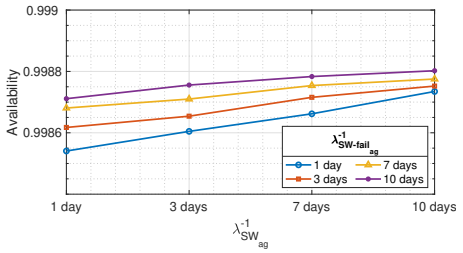


(d) *Component-wise*

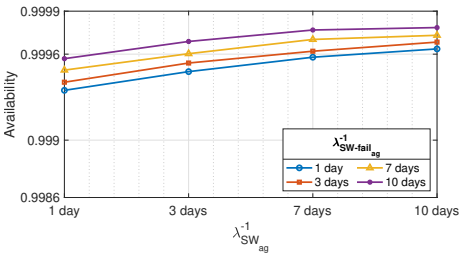
Fig. 11. System availability when varying rejuvenation interval and software aging rate.



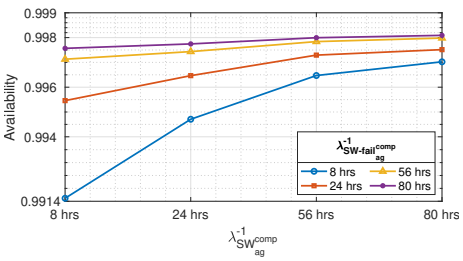
(a) *Manager*



(b) *Manager-Worker*



(c) *Multi-master*



(d) *Component-wise*

Fig. 12. Impact of software aging vs. aging-related failure.

rate of aging. For a short software aging time ($\lambda^{-1}_{\text{SW}_{\text{ag}}}$), i.e, lower than 7 days, an increase of the aging-related failure rate, i.e., lower time for software failure due to aging, can have a significant impact on the MANO availability. On the contrary, for a low to moderate software utilization, i.e., high times for the software to age, variations of the aging-related failure rates have a much lower impact. A similar trend, yet much less marked, is observed also for the *Manager-Worker* deployment. This tendency becomes negligible for the *Multi-master* deployment, therefore supporting the previous finding that adequate protection is needed on both host and software levels for neutralizing variations of critical parameters such as software aging and aging-induced failure rates.

Regarding the *Component-wise* model, Fig. 12(d) shows the sensitivity analysis for single components having three times lower failure intensities on the component's level compared to (such that the overall failure intensity of treating them together is the same as) the single MANO software model, i.e., *Manager*. Also for this analysis, the results show a trend very similar to the *Manager* model. The SSA of both models differs at most $5.05 \cdot 10^{-4}$. Moreover, increments or reductions of software aging parameters produce very much alike impacts

for both models. Such observations, together with the previous insights, show that modeling of the MANO software as a single component yields a reasonable representative analysis of the system's steady state availability.

D. Threats to Validity

A possible limitation of this study concerns the precision in our numerical investigation. This is due to the accuracy of baseline parameters, which is, in general, common to many model-based studies. Although the majority of model parameters have been retrieved from related studies, we acknowledge that the choice of parameters may skew the analytic results. In particular, due to a lack of publicly available data regarding failure and recovery dynamics of MANO systems, we have made reasonable assumptions, based on studies regarding software of similar complexity. To lift this limitation a bit, we have performed experimental trials on a realistic MANO deployment aiming at retrieving recovery parameters' values of MANO software. Nevertheless, the very scope of the sensitivity analysis is to shed light onto the uncertainty related to these parameters, and two-orders of magnitude variation range is, in our opinion, sufficient to capture to a wide extent the uncertainties. An additional threat to the validity of our results is related to some assumptions regarding deployment configurations. In the *Multi-master* deployment, we assume that the load balancer is failure free. This is not the case for realistic deployments. However, from a deployment perspective, a service operator can limit the impact of this threat by using external load balancers which can provide a sufficient level of reliability. In addition, we also assume that while being hosted in the *Manager* node, regardless of the type of fault affecting the Worker node, the MANO software is only subject to non-aging related failures. While this does not reflect a realistic behavior, it is reasonable for those events that require a relatively short time to recover compared to the software aging rate. Overall, the goal of this work is to propose a methodology and model abstractions for assessing MANO implementations which can be used by system operators that have access to empirical data and can extract parameter values for use in the models.

IX. Conclusion

This paper presents four comprehensive availability models for a containerized NFV-MANO architecture encompassing various redundancy configurations. The models incorporate diverse failure modes and the corresponding recovery behaviors, regarding both hardware and software components. The models also include software aging effects and software rejuvenation, as proactive maintenance, aiming at mitigating aging effects. We performed an experimental campaign on real-life MANO system aiming at retrieving realistic system recovery parameters. We carried out an exhaustive sensitivity analysis from which we assess and quantify the steady-state availability and identified the impact that critical parameters have. The investigation showed that non-aging-related software failures and software repair rates stand out as key deteriorating failure and repair parameters, respectively. However, employing clustering mechanisms such as Kubernetes with redundancy on both host and software levels further boosts the NFV-MANO availability. Moreover, software aging can have a considerable impact on the MANO availability and we observed that a correct tuning of the rejuvenation policy can be beneficial and is particularly well-suited in cases where a high software utilization is experienced.

References

[1] ETSI, GS NFV, "ETSI GS NFV 001 v1. 1.1 Network Functions Virtualisation," *Use Cases. sl: ETSI*, 2013.

[2] G. Brown and H. Reading, "Service chaining in carrier networks," *Heavy Reading*, 2015.

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

[4] ETSI, GS NFV, "Network Functions Virtualisation (NFV): Architectural Framework," *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.

[5] ETSI, "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action," *White Paper*, no. 1, pp. 1–16, 2012.

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

[7] R. Mijumbi *et al.*, "Management and orchestration challenges in network functions virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, 2016.

[8] ETSI, GS NFV, "ETSI GS NFV-REL 001 v1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.

[9] 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.

[10] B. Blanco *et al.*, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[11] K. S. Trivedi and A. Bobbio, *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge University Press, 2017.

[12] G. Arfaoui, J. M. Sanchez Vilchez, and J. Wary, "Security and Resilience in 5G: Current Challenges and Future Directions," in *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 1010–1015.

[13] N. F. S. De Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, vol. 142, pp. 69–94, 2019.

[14] A. J. Gonzalez *et al.*, "Dependability of the NFV orchestrator: State of the art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3307 – 3329, 2018.

[15] G. Nencioni *et al.*, "Orchestration and control in software-defined 5G networks: Research challenges," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[16] ETSI, GS NFV, "ETSI GR NFV-REL 007 v1.1.2: Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities," 2017.

[17] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *Ieee Software*, vol. 33, no. 3, pp. 42–52, 2016.

[18] Docker Website. Accessed: 2020-11-15. [Online]. Available: "https://www.docker.com/"

[19] Linux Containers (LXC). Accessed: 2020-11-15. [Online]. Available: "https://linuxcontainers.org/"

[20] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," in *Present and ulterior software engineering*. Springer, 2017, pp. 195–216.

[21] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5G mobile systems," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 483–496, 2016.

[22] S. Sharma, R. Miller, and A. Francini, "A Cloud-Native Approach to 5G Network Slicing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 120–127, 2017.

[23] 5G-PPP Software Network Working Group *et al.*, "From webscale to telco, the cloud native journey," *Editor: Bessem Sayadi, July*, 2018.

[24] Open Source MANO (OSM). Accessed: 2020-11-15. [Online]. Available: "https://osm.etsi.org"

[25] OpenBaton. Accessed: 2020-11-15. [Online]. Available: "https://openbaton.github.io"

[26] SONATA website. Accessed: 2020-11-15. [Online]. Available: "https://www.sonata-nfv.eu/"

[27] ETSI, GS NFV, "ETSI GS NFV REL 004 v1.1.1: Network Functions Virtualisation (NFV); Assurance; Report on Active Monitoring and Failure Detection," 2016.

[28] ——, "Reliability; Report on Models and Features for End-to-End Reliability," no. GS REL 003 v1.1.2, 2016-07.

[29] A. Gonzalez *et al.*, "Service availability in the NFV virtualized evolved packet core," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[30] M. Di Mauro *et al.*, "IP multimedia subsystem in an NFV environment: availability evaluation and sensitivity analysis," in *2018 IEEE NFV-SDN*. IEEE, 2018, pp. 1–6.

[31] ——, "Service function chaining deployed in an NFV environment: An availability modeling," in *IEEE CSCN*. IEEE, 2017, pp. 42–47.

[32] ——, "Availability modeling and evaluation of a network service deployed via NFV," in *International Tyrrhenian Workshop on Digital Communication*. Springer, 2017, pp. 31–44.

[33] B. Tola, G. Nencioni, B. E. Helvik, and Y. Jiang, "Modeling and evaluating NFV-enabled network services under different availability modes," in *2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020*. IEEE, March 2019.

[34] T. Soenen *et al.*, "Optimising microservice-based reliable NFV management and orchestration architectures," in *The 9th International Workshop on Resilient Networks Design and Modeling*, Sep. 2017, pp. 1–7.

[35] S. Sebastio, R. Ghosh, and T. Mukherjee, "An availability analysis approach for deployment configurations of containers," *IEEE Transactions on Services Computing*, pp. 1–1, 2017.

[36] S. Sebastio, R. Ghosh, A. Gupta, and T. Mukherjee, "Contav: A tool to assess availability of container-based systems," in *2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA)*, 2018, pp. 25–32.

[37] M. Grottke and K. S. Trivedi, "Software faults, software aging and software rejuvenation," *The Journal of Reliability Engineering Association of Japan*, vol. 27, no. 7, pp. 425–438, 2005.

[38] K. S. Trivedi *et al.*, "Recovery from failures due to Mandelbugs in IT systems," *Proceedings of IEEE PRDC*, pp. 224–233, 2011.

[39] P. Vizarreta, K. Trivedi, V. Mendiratta, W. Kellerer, and C. Mas-Machuca, "DASON: Dependability Assessment Framework for Imperfect Distributed SDN Implementations," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 652–667, 2020.

[40] B. Tola, Y. Jiang, and B. E. Helvik, "On the Resilience of the NFV-MANO: An Availability Model of a Cloud-native Architecture," in *2020 16th International Conference on the Design of Reliable Communication Networks DRCN 2020*, 2020, pp. 1–7.

[41] L. Foundation. Open Network Automation Platform. Accessed: 2020-11-15. [Online]. Available: "https://www.onap.org/"

[42] G. N. ETSI, "Network functions virtualisation (NFV); management and orchestration; report on architectural options," vol. 1, p. v0, 2016.

[43] Juju Documentation. Accessed: 2020-11-15. [Online]. Available: "https://juju.is/docs"

[44] OpenStack Tacker. Accessed: 2020-11-15. [Online]. Available: "https://wiki.openstack.org/wiki/Tacker"

[45] Prometheus - Monitoring system and Time-series database. Accessed: 2020-02-26. [Online]. Available: "https://prometheus.io/docs/introduction/overview/"

[46] A. S. Foundation, "Apache Kafka," accessed: 2020-11-15. [Online]. Available: "\url{https://kafka.apache.org/intro}"

[47] Docker Documentation. Accessed: 2020-11-15. [Online]. Available: "https://docs.docker.com/engine/swarm/"

[48] Kubernetes Website. Accessed: 2020-11-15. [Online]. Available: "https://kubernetes.io/"

[49] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 USENIX Annual Technical Conference (USENIX-ATC 14)*, 2014, pp. 305–319.

[50] Creating Highly Available clusters with kubeadm. Accessed: 2020-11-15. [Online]. Available: "https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/"

[51] M. Grottke and K. S. Trivedi, "Fighting bugs: Remove, retry, replicate, and rejuvenate," *Computer*, vol. 40, no. 2, pp. 107–109, 2007.

[52] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *2008 IEEE ISSRE Wksp*, Nov 2008, pp. 1–6.

[53] F. Machida, D. S. Kim, and K. S. Trivedi, "Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration," *Performance Evaluation*, vol. 70, no. 3, pp. 212–230, 2013.

[54] M. Escheikh, Z. Tayachi, and K. Barkaoui, "Workload-dependent software aging impact on performance and energy consumption in server virtualized systems," in *2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2016, pp. 111–118.

[55] M. Torquato and M. Vieira, "Interacting SRN models for availability evaluation of VM migration as rejuvenation on a system under varying workload," in *2018 IEEE International symposium on software reliability engineering workshops (ISSREW)*. IEEE, 2018, pp. 300–307.

[56] E. Guedes and P. Maciel, "Stochastic model for availability analysis of service function chains using rejuvenation and live migration," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019, pp. 211–217.

[57] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. M. Machuca, "Characterization of failure dynamics in SDN controllers," in *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2017, pp. 1–7.

[58] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *School organized by the European Educational Forum*. Springer, 2000, pp. 315–343.

[59] Möbius: Model-based environment for validation of system reliability, availability, security and performance. Accessed: 2020-11-15. [Online]. Available: "https://www.mobius.illinois.edu"

[60] T. A. Nguyen, D. S. Kim, and J. S. Park, "A comprehensive availability modeling and analysis of a virtualized servers system using stochastic reward nets," *The Scientific World Journal*, vol. 2014, 2014.

[61] W. H. Sanders and J. F. Meyer, "Reduced base model construction methods for stochastic activity networks," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 1, pp. 25–36, 1991.

[62] D. Yuan, Y. Luo, X. Zhuang, G. R. Rodrigues, X. Zhao, Y. Zhang, P. U. Jain, and M. Stumm, "Simple testing can prevent most critical failures: An analysis of production failures in distributed data-intensive systems," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 249–265.

[63] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from Googles network infrastructure," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 58–72.

[64] R. Hanmer, L. Jagadeesan, V. Mendiratta, and H. Zhang, "Friend or foe: Strong consistency vs. overload in high-availability distributed systems and SDN," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2018, pp. 59–64.

[65] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, "Benchmarking open source NFV MANO systems: OSM and ONAP," *Computer Communications*, vol. 161, pp. 86–98, 2020.

[66] OSM Bugzilla bug tracking system. Accessed: 2020-11-15. [Online]. Available: "https://osm.etsi.org/bugzilla/describecomponents.cgi"

[67] M. Torquato and M. Vieira, "An experimental study of software aging and rejuvenation in dockerd," in *2019 15th European Dependable Computing Conference (EDCC)*, 2019, pp. 1–6.

[68] A. Chou, J. Yang, B. Chelf, S. Hallem, and D. Engler, "An Empirical Study of Operating Systems Errors," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, p. 73–88, Oct. 2001. [Online]. Available: https://doi.org/10.1145/502059.502042

[69] K. Kanoun and Y. Crouzet, "Dependability benchmarks for operating systems," *International Journal of Performability Engineering*, vol. 2, pp. 275–287, 2006.

[70] D. Cotroneo, R. Natella, R. Pietrantuono, and S. Russo, "Software aging analysis of the linux operating system," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*, 2010, pp. 71–80.

[71] D. Cotroneo, A. K. Iannillo, R. Natella, and S. Rosiello, "Dependability assessment of the Android OS through fault injection," *IEEE Transactions on Reliability*, 2019.

[72] R. Matos, J. Araujo, V. Alves, and P. Maciel, "Characterization of software aging effects in elastic storage mechanisms for private clouds," in *2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*, 2012, pp. 293–298.

[73] D. Cotroneo, F. Fucci, A. K. Iannillo, R. Natella, and R. Pietrantuono, "Software aging analysis of the android mobile OS," in *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, 2016, pp. 478–489.

Besmir Tola received the M.Sc. degree in Electronics and Telecommunication Engineering from the University of Siena (Italy) in 2014. In autumn 2015, he joined the IIK department at the Norwegian University of Science and Technology (NTNU) as a Ph.D. candidate in Information Security and Communication Technology. From spring 2020, he holds an Assistant Professor position within the same department. In 2016 and 2018, he was a visiting researcher at the Nokia Bell Labs in Stuttgart (Germany), and UNINETT (Norwegian National Research and Education Network Operator), respectively, where he worked on dependability modeling and analysis of cloud computing infrastructures and services. His research interests include performance and dependability analysis of cloud computing, SDN, and NFV architectures.

Yuming Jiang received the B.Sc. degree from Peking University and the Ph.D. degree from the National University of Singapore. He has been a Professor with the Norwegian University of Science and Technology, Trondheim, Norway, since 2005. From 1996 to 1997, he was with Motorola, Beijing, China, and from 2001 to 2003, he was with the Institute for Infocomm Research (I2R), Singapore. He visited Northwestern University from 2009 to 2010, and Columbia University from 2015 to 2016. His research interests are the provision, analysis, and management of quality of service guarantees, with a focus on (stochastic) network calculus and its applications. He has authored the book entitled Stochastic Network Calculus. He was a Co-Chair of IEEE Globecom 2005 - General Conference Symposium, a TPC Co-Chair of 67th IEEE Vehicular Technology Conference (VTC) 2008, the General Chair of IFIP Networking 2014 Conference, the Chair of the 2018 International Workshop on Network Calculus and Applications, and a TPC Co-Chair of the 32nd International Teletraffic Congress (ITC32), 2020.

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), the Department of Telematics and Department of information Security and Communication Technology. In the period 2009 – 2017, he has been Vice Dean with responsibility for research at the Faculty of Information Technology and Electrical Engineering at NTNU. 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. During 2003 - 2012 Principal investigator at the Norwegian Centre of Excellence Q2S - the Centre for Quantifiable Quality of Service and is since 2020 Principal investigator at the Centre for Research based Innovation NORCICS - Norwegian Center for Cybersecurity in Critical Sectors. 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 is on ensuring dependability in services provided by multi-domain, virtualised ICT systems, with activities focusing on 5G and SmartGrids.

PAPER D

Network-Aware Availability Modeling of an End-to-End NFV-Enabled Service

Besmir Tola, Gianfranco Nencioni, and Bjarne E. Helvik

IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1389-1403, Dec. 2019
DOI: 10.1109/TNSM.2019.2948725

Network-Aware Availability Modeling of an End-to-End NFV-enabled Service

Besmir Tola, *Member, IEEE,* Gianfranco Nencioni, and Bjarne E. Helvik, *Life Senior Member, IEEE*

***Abstract*—Network Function Virtualization (NFV) represents a key shift in nowadays network service provisioning by entailing higher flexibility, elasticity, and programmability of network services. Dependability is one of the main aspects that need to be investigated and tackled in order to profitably use NFV in the future. The main objective of this paper is to propose a comprehensive approach to estimate the end-to-end NFV-deployed service availability and present a quantitative assessment of the network factors that affect the availability of the service provided by an NFV architecture. To achieve this goal, we adopted a two-level availability model where i) the low level considers the network topology structure and NFV connectivity requirements through the definition of the system structure function based on minimal-cut sets and ii) the higher level examines dynamics and failure modes of network and NFV elements through stochastic activity networks. By using the proposed model, we have carried out an extensive sensitivity analysis to identify the impact on the service availability of the different service elements involved in the delivery, and their deployment across the network. The results highlight the significant impact that network nodes have on the end-to-end network service. Less robust network nodes may reduce the availability of an NFV-enabled service by more than one order of magnitude even though NFV elements like VNFs or MANO are provided with redundancy. Moreover, the results show that adopting an SDN-integrated network degrades the service availability and increases the vulnerability of the network service to SDN controllers unless adequately protected.**

***Index Terms*—NFV, Software-defined Networking, Service Function Chaining, Availability Modeling, SAN Models.**

I. Introduction

NETWORK Function Virtualisation (NFV) has drained significant attention from the research community due to its promising benefits in network manageability, cost efficiency, and reduced time to market of new and more specialized network services. Through the use of virtualization and paradigms like cloud computing, it decouples network function software from expensive purpose-built hardware and runs them as software deployed on Commercial Off-The-Shelf (COTS) hardware [1]. As such, NFV provides the necessary flexibility to enable agile, cost-effective, and on-demand service delivery model in conjunction with automated management.

According to the European Telecommunications Standards Institute (ETSI) [1], the high-level NFV architectural framework consists of three main blocks which include: i) Virtualised Network Functions (VNFs), ii) NFV Infrastructure (NFVI) and iii) NFV Management and Orchestration (MANO) block. The latter comprises the NFV Orchestrator (NFVO), VNF Manager (VNFM) and Virtualised Infrastructure Manager (VIM) where the communication among the functional blocks is enabled through well-defined reference points.

The VNF is the software implementation of a network function and it is executed on the NFVI, which encompasses a set of diverse physical resources and their virtualization software. The NFVI may be distributed on geographically distinct locations, called NFVI Point of Presences (NFVI-PoPs), and the related resources (e.g. compute, storage and network) are managed and controlled by one or more VIMs. The VNFM is the entity responsible for the lifecycle management (e.g. instantiation, scaling, termination, healing , and monitoring) of one or more VNF instances. Moreover, the NFVO is in charge of the orchestration and management of NFVI resources across multiple VIMs and the lifecycle management of network services. The NFVO and VNFM work jointly to ensure that the network services and their corresponding VNFs meet the service quality requirements specified in a Service Level Agreement (SLA), e.g., throughput, latency and reliability [2].

In order to be fully beneficial, the success of NFV is tightly coupled with several challenges that need to be addressed, where service *dependability*, as the ability to deliver a service that can justifiably be trusted [3], represents a major concern [4], [5], [6]. In addition, the upcoming 5G cellular system, for which NFV represents an essential enabling technology [4], envisions very demanding usage scenarios like Ultra Reliable and Low Latency Communications (URLLC). A URLLC service expects that the underlying infrastructure is able to provide more than fine-nines availability being translated into less than 5 minutes of downtime per year. Therefore, it becomes important to assess and quantify the dependability of NFV-enabled services.

Evaluation of system dependability (reliability, availability, etc.) is commonly achieved through analytic and numerical methods [7]. In its specification regarding end-to-end reliability [2], ETSI provides several guidelines for modeling and estimating NFV service reliability and availability. They stress out that a correct reliability/availability estimation should incorporate all the service elements and components involved in the end-to-end delivery. The supporting infrastructure, both computing and transport network, and the inter-dependencies with the software providing the service, i.e., VNFs, are required to be taken into account when estimating the reliability or availability of the service. On the other hand, they present rather simple models consisting of series and/or parallel combinations of reliability block diagrams, hence, failing to capture failure/repair dynamics of service elements and their constituent components.

A number of previous works have quantified the availability of NFV-oriented services, either in "general" terms or by selecting specific NFV service use cases [8], [9],

[10]. Nevertheless, none of these works have performed an exhaustive assessment of NFV service availability since they lack key service elements like physical network links or forwarding/routing devices which are essential networking elements inter-connecting VNFs composing a service chain. Thus, as emphasized by ETSI as well, we found that incorporating the network and the topological dependencies remains a preliminary endeavor for a correct and complete end-to-end NFV service dependability assessment. This served as primary motivation for our contribution in this paper. In addition, NFV and Software-defined networking (SDN) are increasingly becoming co-dependent since the later brings the necessary flexibility in managing network resources for composing network functions into higher-level services [11]. Therefore, it is important to assess the network service dependability also for SDN-integrated NFV-based services. This further motivates our investigation and research contribution.

Availability, as the probability that service will be provided when needed, is regarded as the most important dependability attribute in networks [12]. As specified in [12], service availability is considered of major importance to end users and it has to be defined in a clear and concise way in the SLA. Thus, in this work we focus on the availability of end-to-end NFV-enabled services. To this end, the objective of this paper is to provide an approach for a more accurate prediction of the availability of NFV-based services than the current state of the art by both taking into account the structural properties of the underlying physical network, computing and storage infrastructure, and the dynamic behavior of network elements and functions.

In this paper, we present a two-level availability model where i) the lower level consists of the structural analysis based on minimal-cut sets which are derived by the network connectivity requirements for ensuring an end-to-end network service, and ii) the higher level is composed of the availability models, based on stochastic activity network (SAN), of the network and NFV elements that are needed to provide an NFV-based service. The two levels are merged by applying the *inclusion-exclusion principle*. Moreover, we perform a quantitative assessment and sensitivity analysis from which we are able to identify the main critical parameters in the deployment of the NFV elements that influence the overall service robustness. By identifying such parameters, we gain insights that could be exploited for designing and operating an NFV-based network service such that high-grade availability requirements are to be met.

The remainder of the paper is organized as follows. In Section II, we discuss the relevant studies regarding NFV dependability. Section III introduces the service elements composing an end-to-end NFV-based service and the related dependability challenges. In Section IV, a representative network topology is introduced together with a set of VNF, NFVI-PoP, and MANO configuration cases. The objective of this is twofold, to give a reference for the discussion of structural modelling in the next section and to serve as a basis for the numerical studies at the end of the paper. As indicated, in Section V, the two-level model used to evaluate the end-to-end service availability is presented. Discussion of the numerical results of the sensitivity analysis in regard to the most critical parameters is presented in Section VI. Finally, Section VII summarises the paper by highlighting the most important conclusions.

II. Related Work

There are several methodologies that dependability studies have used to develop analytic models for quantifying system dependability. A thorough introduction may be found in [7]. For a better understanding of the different techniques utilized in the related work, we briefly summarize the most common methodologies.

Analytic dependability models typically fall into three categories: i) Non-state-space models, ii) State-space models, and iii) Hierarchical models.

Typical non-state-space models include Reliability Block Diagrams (RBD), Fault-trees (FT), and Reliability Graphs (RG). RBDs and FTs are used to represent the logical structure of a system, with respect to how availability of system components impacts the overall system availability.

State-space models are used to model complex interactions and behaviors within a system. A variety of state-space modeling techniques have been used in previous works. They span from Markov-based models like discrete/continuous-time Markov chains (D/CTMC) to semi-Markov Processes. When a reward function is associated with the chain, for the evaluation of a certain metric, they are known as Markov reward models (MRM). Other representatives of state-space models, which are more human intuitive, include Petri-net (PN)-based models like stochastic-Petri nets (SPN) and generalized-SPN (GSPN). When a reward rate is associated with the net, it is a stochastic reward net (SRN). An additional of PNs are stochastic activity networks (SANs).

Hierarchical models are multi-level models where higher levels are frequently non-state space models and lower levels are typically state-space models which are more suitable for capturing individual complex behavior. A common feature of multi-level models, which makes them more useful in comparison to state space models, is the limitation of state-space explosion when dealing with large and complex systems.

Server virtualization represents a key enabling technology in NFV [13]. The authors of [14], [15] laid the groundwork of availability modeling involving virtualized systems. They use a two-level hierarchical model, composed of CTMC and FT, to represent and compare virtualized and non-virtualized server systems. Through a parametic sensitivity analysis, they were able to identify the parameters deserving more attention for improving the availability and the capacity oriented availability, i.e., performability, of the system. However, due to the nature of CTMCs, complex systems may have to deal with a state space explosion which represents an important drawback. Kim *et al.* [16] exploits Stochastic Reward Nets, an extension of Petri nets, to overcome this drawback. They extend the work in [14] by proposing a scalable model which is able to incorporate more failure and recovery behaviors involved in virtualized server systems, and include features like virtual machine live migrations and high availability.

Surprisingly, only a few works propose and quantitatively assess an NFV-based network service availability.

In [8], the authors present an availability model of a virtualized Evolved Packet Core, as an NFV use case, by using SANs. They assess the system availability through discrete-event simulation and identify the most relevant criteria to account for by service providers in order to meet a certain availability level. In addition, they model events like catastrophic failures as such events may represent a serious threat to the overall system availability.

A two-level hierarchical availability model of a network service in NFV architectures has been proposed in [17]. By aggregating RBDs (higher level) and SRNs (lower level), they evaluate the steady-state availability and perform a sensitivity analysis to determine the most critical parameters influencing the network service availability. Similarly, in [18], they extend such analysis by including the VIM functionality, as the entity responsible for the management of the network service, into the RBD. Their main findings indicate that a relatively small increment of hypervisor or VNF software failure intensity has a marginal effect on the service availability. In addition, they identify the most appropriate redundancy configuration in terms of additional replicas for providing fine-nines availability. The same authors model and assess the availability of an NFV-oriented IP multimedia subsystem (IMS) [9]. Exploiting the same modeling technique, consisting of a hierarchical model composed of RBD and SRN, they assess the availability of a containerized IMS and perform a sensitivity analysis on failure and repair rate of some of the IMS components. In addition, they identify the best k-out-of-n redundancy configuration for each elements of the IMS such that a five-nine availability is reached.

In a more recent study [10], a composed availability model of an NFV service, based on SANs, is proposed. Each VNF, composing the network service, is considered as a load-sharing cluster and the authors propose separate models for various redundancy mechanisms called Availability Modes. Through a sensitivity analysis, they investigate the effects of cluster provisioning and recovery strategies for each mode aiming at finding the most appropriate configuration providing the highest level of service availability.

The contribution of this work compared to the related studies differs in several points that aim at filling the current gap when estimating end-to-end NFV-based service availability. None of the previous works has considered the effects of the underlying physical network and its intrinsic topological dependencies emerging from the network connectivity requirements. In addition, the related works provide insights regarding a limited set of failure parameters associated with NFV elements and do not consider the impact of the failure dynamics of networking devices on the service availability. Instead, in this proposed approach, the network structural analysis allows evaluating the impact of the network connectivity in provisioning a highly dependable network service. Moreover, the dynamic models of the NFV-based service elements permit to identify the critical failure parameters, within the network and NFV elements, that impact the end-to-end service availability. Furthermore, this contribution can be seen by service operators as a starting point for developing a decision support tool in designing and operating fault tolerance and redundancy strategies to fulfill the resilience requirements of carrier-grade services. To the best of our knowledge, this approach is the first model to incorporate the impact of the transport network in an NFV-oriented service.

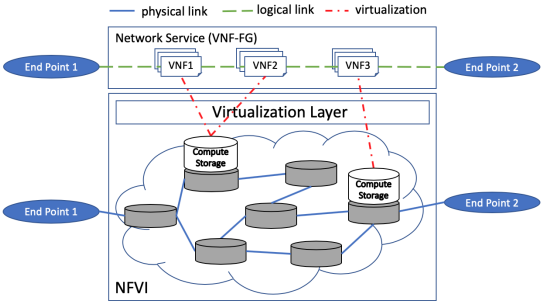


Fig. 1. Delivery of an end-to-end NFV-based service.

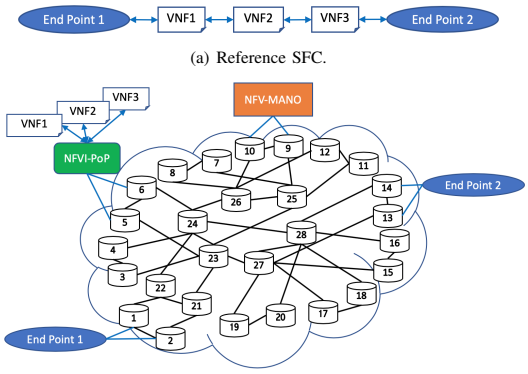
III. Dependability of an NFV-based Service

In NFV, a network service can be visualized architecturally as a forwarding graph of (virtual and physical) network functions supported and interconnected by the underlying network infrastructure. According to ETSI [1], a VNF Forwarding Graph (VNF-FG) defines the composition of VNFs, providing an NFV-enabled service and their relative sequence for traffic to traverse. Similarly, the Internet Engineering Task Force (IETF) specifies a Service Function Chaining (SFC) as "the definition and instantiation of an ordered set of service functions and subsequent steering of traffic through them" [19]. In the NFV context, both nomenclatures refer to the same thing, hence, hereafter we will refer to an SFC as the composition of an ordered set of VNFs providing a service. Thus, the delivery of an end-to-end service, illustrated in Figure 1, where both end points are customers of the NFV architecture, comprises several network functions, which are mutually connected in parallel or in series, to construct a network service graph in the form of a SFC. The service is implemented and operated through an interaction of the SFC, realizing the service, and the MANO, which acts as the manager of the service lifecycle.

The underlying network contributes to the behavior of the higher-level service which in turn can be regarded as a combination of the behavior of its constituent functional elements [1]. Thus, the delivery of a network service needs to be estimated based on the following functional elements:

- ingress and egress *end points*;
- physical and virtual *network functions* that constitute the SFC between the end points;
- *supporting infrastructure* (e.g., compute and storage nodes) that runs the VNFs;
- *networking devices* that allow the interconnection of the network functions.

From a dependability perspective, a network service could be potentially threatened by the failure of any of these elements. The transition to NFV deployments introduces additional challenges that service providers need to account for. As identified by ETSI [20], a typical challenge resides in



(a) Reference SFC.

(b) Reference NFV deployment.

Fig. 2. Network topology and NFV service deployment.

the dependency among VNFs, the virtualization layer, and the hardware infrastructure. By decoupling the software from hardware, the VNFs are not aware of the underlying hardware. Henceforth, a failure on the physical infrastructure may cause a service outage in case several VNFs share the same hardware, as opposed to physical network functions where the hardware is dedicated to a specific function. In addition, the virtualization layer introduces an additional failure source. The hypervisor itself may be prone to software failures which may affect a large part of the software infrastructure. Moreover, the NFVI will rely on extensive use of commercial off-the-shelf (COTS) servers which are usually more error-prone compared to specialized hardware implementing legacy network functions [5]. As a result, dependability may potentially represent a key threat to the success of NFV architectures and ETSI has streamlined specific reports in regard to reliability models, capabilities, and requirements [2], [20], [21].

IV. Network Topology and Case Studies

The reference SFC that will be considered in our assessment is depicted in Figure 2(a) and is composed of three VNFs. The SFC will be deployed in a real world-wide backbone network [22] which is composed of 28 nodes and 40 links, as illustrated in Figure 2(b). Note that only the network topology had been adopted from a real backbone network and the NFV deployment together with its relative redundancy configuration will be subject of investigation.

The location of the end points 1 and 2 will be fixed in all the evaluations, whereas the location and the redundancy of the NFV elements (VNF, NFVI-PoP, MANO) will change during the evaluations. Initially, the scenario where all the three VNFs are deployed into the same NFVI-PoP, referred to as the *Reference* case, is considered. In this scenario, both NFVI-PoP and MANO are placed in the edge part of the network. Afterward, the cases where the VNFs are deployed into two and three separate NFVI-PoPs (denoted *2 NFVI-PoPs* and *3 NFVI-PoPs*, respectively), placed in the edge, are investigated.

Note the representation of NFVI-PoPs and VNFs. The NFVI-PoP represents a physical entity and includes the physical resources and the software for managing the resources. The VNF represents the virtual resources and the software function that is using the resources. One or multiple VNFs are running on a NFVI-PoP. Given this assumption, the arrowed lines that connect the VNFs to the NFVI-PoP are virtual connections which we assume to be fault-free. Therefore, they are not considered as links in the structural analysis. In addition, we regard the SFC availability from the network operator's customer interface. Hence, we consider the end points and their connecting links outside the scope of the NFV-service availability evaluation. Lastly, we do not optimize the placement of NFVI-PoPs or VNFs across the network, since such problems fall outside the scope of this paper and regard challenges associated with resource allocation where service availability can be treated as an objective function or constraint, as investigated in works like [23], [24] and the references therein. Nonetheless, to acquire further insights, in addition to the *Reference* case, we evaluate the service unavailability even when the NFV elements are directly connected to the network nodes having a higher betweenness centrality, i.e., the core nodes of the backbone network. We refer to this deployment as the *Core* case and present the results of both redundant and non-redundant configurations in the numerical evaluation (Section VI-F).

Moreover, an integration with Software-Defined Networking (SDN) can be also considered. SDN consists in the separation of the control and data planes and the logical centralisation of the control plane in the SDN controller. In this case, several deployment strategies can be considered. As identified by [11], there are several use cases for SDN integration with NFV. Some of the Proof of Concepts (PoCs) regard the SDN controller merged within the VIM functionality as part of the MANO entity, whereas others consider the SDN controller as part of the NFVI or as a virtualised entity similar to a VNF. In this paper, we assume that the SDN functionality is part of the VIM entity but their location placement are geographically separated, as would the case when the NFV-based service provider and the network operator are two distinct entities.

Furthermore, a *redundant deployment* can be considered in order to provide a resilient service. In this case, the MANO, which is a logically-centralized entity, can be physically split or duplicated in different geographical areas. The VNFs, which are logical entities running on geographically-distributed computing centers, can be split or duplicated in the same (local) computing center or in other (remote) computing centers. Similarly, when an SDN-integrated architecture is considered, the SDN controller can be duplicated into separate locations in order to provide redundancy.

Figure 3 depicts the case study when a redundant deployment is considered. When only the MANO is redundant, the *Reference* deployment is considered but the dash-dot MANO element represents the MANO redundant unit which is denoted as *MANO redundant*. Similarly, in case the VNFs (and the NFVI-PoPs) are the only elements having redundant units they are denoted as *VNF redundant*. In case all the NFV elements are redundant, the deployment, denoted as *All redundant*, represent the case of fully redundant NFV service. When an SDN-integrated network is assumed, the SC node denotes the SDN controller and the relative dash-dot element represent the

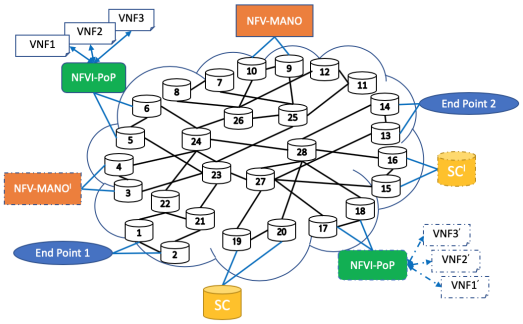


Fig. 3. SDN-integrated NFV redundant deployment.

redundant unit.

V. NFV-based Service Availability Modelling

In this section, we introduce the two-level model used to evaluate the availability of an NFV-based network service. Specifically, we regard the availability in terms of the steady-state availability, hereafter simply referred to as availability. The modeling approach consists of two levels:

- *Structural* model of the network topology and NFV deployment;
- *Dynamic* models of NFV-based service elements.

The two-level approach seeks to depict a large-scale NFV infrastructure that is deployed on top of network and computing infrastructures. The structural model assesses the network connectivity required to deliver an end-to-end NFV-based service by means of an SFC where the VNFs are running on computing centers distributed on the network infrastructure. For the structural model, reliability block diagram, fault trees, or structure functions expressed as minimal-cut or -path sets can be used (see Section V-A). The dynamic models characterize the potential failure causes of the elements needed to deliver an end-to-end NFV-based service. For the dynamic models, Markov model, Stochastic Petri nets, or extensions of the later can be used (see Section V-B).

In the following subsections, we introduce our approach through the case studies presented in Section IV which include the reference SFC that constitutes the NFV-based service. First, we present the connectivity requirements for providing an end-to-end NFV-enabled service and based on them the structure functions for each case study and minimal-cut sets are computed. Second, we introduce simple SAN models that characterize the failure dynamic behavior of the network and NFV elements. Finally, we show how to combine the two levels and evaluate the end-to-end service availability.

A. Structural Model

Structural models are an attractive technique for performing system dependability assessment [25]. Key dependability properties can be extracted from the structure function. Consider a system with n subsystems. Each subsystem can have two possible states: working and failed. As a result, the state of each i subsystem is given by a binary variable x_i, where $x_i = 1$ if the subsystem is working and $x_i = 0$ if the subsystem is failed. Hence, the state vector of the overall system is:

$$\mathbf{x} = (x_1, x_2, ..., x_n)$$

and the system operational mode can be described by the following binary function:

$$\Phi(\mathbf{x}) = \Phi(x_1, x_2, ..., x_n)$$

which is defined as the structure function and corresponds to a logical Boolean function that expresses the system mode, i.e., working or not. As a boolean function, it can be represented in one of the two canonical forms, the *Minimal sum-of-products form* (I^{st}-canonical form) or *Minimal product-of-sums form* (II^{nd}-canonical form). From these forms, we can extract dependability properties namely path and cut sets. The definition of the connectivity requirements will determine the most critical elements involved in an end-to-end network service and by means of the structural analysis, either based on *minimal-path sets* or *minimal-cut sets* [25], we are able to identify such elements. In this paper, we make use of *minimal-cut sets* and the following definitions apply:

Definition 1 (Cut set): A set of structure components that by failing ensures that the structure is failed.

Definition 2 (Minimal-cut set): A cut set of a structure that cannot be reduced without loosing status as a cut set.

Definition 3 (Structure function): Each max-term of the structure function expressed in a minimal product-of-sum form corresponds to a minimal-cut set.

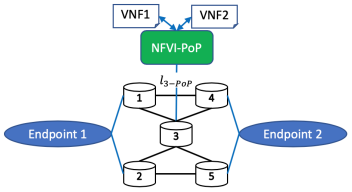


Fig. 4. Showcase for the structural analysis.

To better illustrate, Figure 4 depicts a small system structure with five network nodes and a chain of two VNFs deployed in one NFVI-PoP. For simplicity, let us assume that the links connecting the network nodes do not fail. Let us consider a working service as a "flow" moving from endpoint 1, receive service from the VNFs, to endpoint 2. Note that the requirement of the flow being able to receive service from the VNFs defines a specific connectivity requirement that will influence the structure function. If the system has failed, the flow is prevented from being served and reaching the destination. The system is considered to be working if there exists a set of functioning components that permits the flow to be served by the VNFs and reach the destination.

From *Definition 1*, the cut sets of the structure are all the possible combinations of the components such that their simultaneous failure ensures that the system is in a failed state. Such cut sets are $\{VNF_1\}$, $\{VNF_2\}$, $\{NFVI\text{-}PoP\}$, $\{l_{3-PoP}\}$, $\{3\}, \{1,2\}, \{4,5\}, \{1,3,5\}, \{2,3,4\}, \{1, l_{3-PoP}, 4\}, \{1,2,VNF_1\}$, etc. Applying *Definition 2*, we can identify those sets that are strictly required to fail, i.e., minimal, such that the system

is failed. The statement “cannot be reduced” implies that if we remove one or more components from a minimal cut set, the set is no longer a cut set. Henceforth, the minimal-cut sets are only $\{l_{3-PoP}\}$, $\{VNF_1\}$, $\{VNF_2\}$, $\{NFVI\text{-}PoP\}$, $\{3\}, \{1,2\}, \{4,5\}$ and the structure function, in the form of *minimal product-of-sums*, is defined as:

$$\Phi(\mathbf{x}) = x_{VNF_1} \cdot x_{VNF_2} \cdot x_{NFVI-PoP} \cdot x_{l_{3-PoP}} \cdot x_3 \cdot (x_1 + x_2) \cdot (x_4 + x_5)$$

which aligns with *Definition 3*. In other words, the structure function identifies those system elements that being unavailable cause a system unavailability.

The adoption of an NFV architecture will change the way network services are provisioned compared to legacy networks by including more flexibility, automation, and agile orchestration. The key features of the new service delivery paradigm are the following: "centralisation" of the control logic into the MANO; "remotisation" of the network functions; "sharing" of the computing resource; geographical "distribution" of the computing centers. These features lead to an increase in the network connectivity requirements for provisioning a network service that can be summarized as follows:

- *MANO – end points connectivity:* The end point must be able to connect with the MANO in order to trigger the service provisioning.
- *MANO – VNF connectivity:* The MANO must be able to connect with the VNFs composing the SFC in order to orchestrate and manage the lifecycle of the VNFs.
- *SFC connectivity:* The ordered connectivity of the VNFs (and the end points) composing the SFC must be assured.

The first two connectivity requirements are related to the *control plane* in NFV and concern the necessary requirements of service request acceptance and management and orchestration of VNFs. Whereas, the last requirement regards the *data plane* layer and the correct service composition.

In case an SDN integrated network is considered, further connectivity requirements need to be included.

- **MANO – SDN controller connectivity**
  The peer-to-peer communication between the MANO and the SDN controller must be guaranteed in order to allow the request of the network resources for composing the SFC.
- **SDN controller – network nodes connectivity**
  The SDN controller must be able to connect with the network nodes that compose the paths among the elements in the SFC.

Furthermore, for a redundant deployment, the above connectivity requirements need to be modified accordingly, e.g., the requirement can be relaxed by ensuring the connectivity to at least one of the redundant elements.

For all the examined NFV deployments, their connectivity requirements are very important in establishing, through the structure function, the most critical elements in the delivery of a network service. For example, the requirement of ensuring an ordered connectivity of the VNFs, i.e., the SFC, is reflected in the structure function by imposing this condition when finding all the paths that include an ordered sequence of the VNFs. Accordingly, for each NFV deployment, this requirement will be embedded into the structure function from which we derive the relative minimal-cut sets. For further details on the structure function analysis, the reader may refer to [7], [25].

B. Dynamic Models

The second part of the two-level model consists of the dynamic models of network and NFV elements. To establish these models, Stochastic Activity Network (SAN) formalism is used. This enables detailed performance, dependability, or performability models to be defined in a comprehensive manner [26].

SANs are stochastic extentions of Petri Nets consisting of four primitives: *places*, *activities*, *input gates*, and *output gates*. Places are graphically represented as circles and contain a certain number of tokens which represent the *marking* of the place. The set of all place markings represent the state of the modeled system. Activities are action that take a certain amount of time to complete. They impact the system performance and can be *timed* (thick vertical lines) or *instantaneous* (thin vertical lines). A timed activity has a distribution function associated with its duration and can have distribution case probabilities used to model uncertainty associated with activity completion. The case probabilities are graphically represented as small circles on the right of the activities. Upon completion, an activity fires and enables token movements from places connected by incoming arcs to places connected by outgoing arcs. This way a system state update occurs and tokens are moved from one place to another by redefining the places markings. Input and output gates define marking changes that occur when an activity completes. Different from output gates, the input gates are also able to control the enabling of activity completion, i.e., firing. The models presented below are defined in the Möbius software tool [27].

Dynamic models are defined for the following elements:

- Network elements:
  - Connecting links;
  - IP router (*traditional network case*);
  - SDN switch (*SDN case*);
  - SDN controller (*SDN case*);
- NFV elements:
  - NFVI-PoP;
  - VNF;
  - MANO.

It is an objective that these models should be simple, yet sufficient. More complex and comprehensive models can be realized, but in this paper, we preferred to use models that enable us to apprehend the essential features of the system and emphasize the necessary details of the elements while keeping the complexity low since our focus is to evaluate the impact of networking on NFV-based service provisioning.

SAN models of network elements (for both SDN and traditional network) have been already proposed [28] and we will use the same models.

The NFVI comprises several geographical locations, and the transport network providing connectivity between these locations is considered as part of the whole infrastructure. A specific geographic location is where an NFVI-PoP (e.g., a

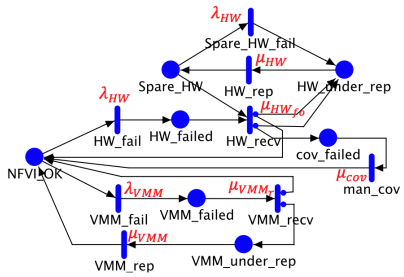


Fig. 5. SAN model of an NFVI-PoP.

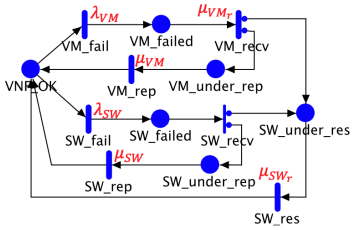


Fig. 6. SAN model of a VNF.

data center) is located and where a number of NFVI-Nodes reside. NFVI-Nodes are a group of physical devices that provide the necessary (computing, storage, and networking) resources needed by the VNF execution environment. Without any loss of generality and to keep a low complexity, we will consider NFVI-PoP and NFVI-Node as a single entity.

In modeling the VNF system, the choice of the virtualization technology used, i.e., hypervisor- or container-based, can determine the model. We believe that from a dependability perspective, the hypervisor-based technology represents a more advantageous choice due to, among others, stronger isolation between virtual and the physical machine or a higher fault detection coverage compared to containers, as shown by studies like [29]. Hence, in our model we assume a hypervisor-based technology and from a VNF perspective and depending on the deployment strategy, the VNF itself may have different failure sources. For example, when two or more VNFs are deployed in a single NFVI-PoP, the failure of the physical or hypervisor level represent a common cause failure for the different VNFs deployed on the same node. As such, we split the failure causes of the VNFs into those related to the underlying infrastructure which may represent a common failure mode for several VNFs, i.e., NFVI-PoP, and those representing the failure of the VNF itself which include the Virtual Machine (VM) and the VNF software.

1) NFVI-PoP: The SAN model of the NFVI-PoP is depicted in Figure 5. In the model we focus on the two main components that constitute the NFVI-Node which may cause a failure on the physical level, i.e., hardware and the Virtualisation-layer software infrastructure, otherwise called Virtual Machine Manager (VMM) or hypervisor. The model is composed of the following places:

- *NFVI_OK* corresponds to the fully working state of the system and is initialized with 1 token;
- *HW_failed* is populated with one token in case a failure of hardware level (memory, disk, I/O, storage etc.) is experienced, 0 otherwise;
- *HW_under_rep* represents the state where the failed hardware undergoes a repair process;
- *Spare_HW* represents the redundant hardware infrastructure ready to take over in case a hardware failure is experienced and it is initialised with one token;
- *cov_failed* represents the state where the hardware failover is unsuccessful and thus, manual intervention is required to bring the hardware up;
- *VMM_failed* represents the state when the virtualization software is failed.
- *VMM_under_rep* represents the state where the VMM undergoes a hard repair process, i.e., applying a fix/patch or software update;

Similarly to many related work and studies performing availability modeling and analysis, see for example [8], [9], [17], [18], we assume that timed activities follow an exponential distribution. The places in the model are connected by means of the following timed activities:

- *HW_fail* and *HW_repair* represent the hardware failure and recovery events with rates λ_{HW} and μ_{HW}, respectively;
- *Spare_HW_fail* represents the redundant hardware failure event with rate λ_{HW};
- *HW_recv* represents the hardware failover event with rate and $\mu_{\text{HW}_{\text{fo}}}$. There are two cases, with probability C_{fo} the failover procedure is successful where one token, fetched from *Spare_HW*, is moved to *NFVI_OK* and another one is placed in *HW_under_repair* in order to repair the failed hardware unit. Whereas with probability $1 - C_{\text{fo}}$ the failover is unsuccessful and one token is placed in *HW_under_repair* and another is moved back to *HW_failed* for a new failover procedure;
- *man_cov* represents a manual coverage intervention executing a hard recovery, with rate μ_{cov}, when an unsuccessful hardware failover is experienced;
- *VMM_recv* represents the recovery process of the virtualization software with rate μ_{VMM_r}. It consists in a simple software reboot process and there are two cases, with probability C_{vmm} a simple reboot successfully recovers the failure and with probability $1 - C_{\text{vmm}}$ the reboot is not successful therefore a hard repair is needed. In both cases, a token is moved from *VMM_failed* to *NFVI_OK* or *VMM_under_rep*, respectively.
- *VMM_fail* and *VMM_rep* represent the failure and hard repair process of the visualization software with rate λ_{VMM} and μ_{VMM}, accordingly.

2) VNF: Figure 6 illustrates the SAN model of a VNF. The model considers failures on the VM and VNF software components. Once a VM failure is evidenced, the recovery undergoes a simple restart where with probability C_{VM} the restart successfully recovers the failure and with probability $1 - C_{\text{VM}}$ a hard repair (patching or fixing) is needed. If the VM restart is successful, the system undergoes a VNF software restart (*SW_res*) to fully recover. Similarly, if a VNF software

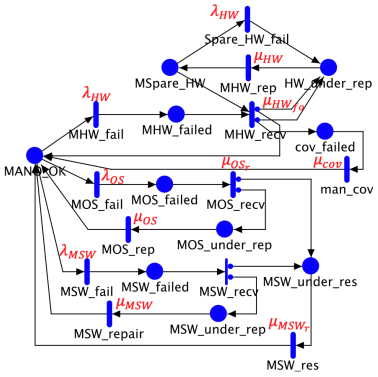


Fig. 7. SAN model of a MANO.

is experienced, with probability C_{SW} the VNF software restart successfully recovers the failure and with probability $1-C_{VM}$ a software fixing is needed (*SW_rep*).

The model is composed of the following places:

- *VNF_OK* represents the fully working state of the system and is initialized with one token;
- *VM_failed* and *SW_failed* correspond to the states in which the VM or VNF software are failed. They are populated with one token in case a failure is experienced, 0 otherwise;
- *VM_under_rep* and *SW_under_rep* represent the states where the VM and VNF software undergoes a hard repair process, accordingly.
- *SW_under_res* corresponds to the state in which the VNF software undergoes a simple software restart action.

The VNF is failed if there are no tokens in *VNF_OK*. The following negative exponentially distributed timed activities connect the places of the model:

- *VM_fail* and *VM_rep* represent the VM failure and hard repair events with rates λ_{VM} and μ_{VM}, respectively;
- *SW_fail* and *SW_rep* represents the failure and hard repair events of the VNF software with rate λ_{SW} and μ_{SW}, respectively.s
- *VM_recv* represents the recovery process of the VM with rate μ_{VM_r}. It consists in a simple VM reset process and there are two cases, with probability C_{vm} a simple reset successfully recovers the failure and with probability $1-C_{vm}$ the reset is not successful therefore a hard repair is needed. In both cases, a token is moved from *VM_failed* to *SW_under_res* or *VM_under_rep*, respectively. Note that, in case the VM reset is successful there is a need to perform a VNF software restart to bring the system up. With nowadays technologies, these action times are comparable thus the need to include a VNF software restart becomes significant.
- *SW_recv* is an instantaneous activity which only models the software simple restart coverage. With probability C_{sw}, a simple software restart recovers the software failure and with $1-C_{sw}$ a hard software repair is needed.

TABLE I
MODEL PARAMETERS FOR THE NFVI-POP, VNF AND MANO WITH THEIR RESPECTIVE NUMERICAL VALUES USED IN THE CASE STUDIES.

Intensity	Time	Description [Mean time to]
$1/\lambda_{HW} = 6$	months	next hardware failure
$1/\mu_{HW} = 2$	hours	hardware repair
$1/\mu_{HW_{fo}} = 3$	minutes	hardware failover
$1/\mu_{cov} = 30$	minutes	manual coverage
$1/\lambda_{VMM} = 4$	months	next VMM failure
$1/\mu_{VMM} = 1$	hour	VMM hard repair
$1/\mu_{VMM_r} = 1$	minute	VMM reboot
$1/\lambda_{VM} = 3$	months	next VM failure
$1/\mu_{VM} = 1$	hour	VM hard repair
$1/\mu_{VM_r} = 30$	seconds	VM reset
$1/\lambda_{OS} = 2$	months	OS failure
$1/\mu_{OS} = 1$	hour	OS hard repair
$1/\mu_{OS_r} = 1$	min	OS reboot
$1/\lambda_{SW} = 2$	weeks	next VNF software failure
$1/\mu_{SW} = 30$	minutes	VNF software hard repair
$1/\mu_{SW_r} = 15$	seconds	VNF software restart
$1/\lambda_{MSW} = 1$	month	next MANO software failure
$1/\mu_{MSW} = 30$	minutes	MANO software hard repair
$1/\mu_{MSW_r} = 30$	seconds	MANO software restart
$C_{fo} = 0.95$		failover coverage factor
$C_{VMM} = 0.9$		VMM reboot coverage factor
$C_{VM} = 0.9$		VM reset coverage factor
$C_{OS} = 0.9$		OS reboot coverage factor
$C_{SW} = 0.8$		VNF software restart coverage factor
$C_{MSW} = 0.85$		MANO software restart coverage factor

3) MANO: There are several differing MANO designs and the authors of [30] review some of them. We decided to represent a high-level architecture of a widely referenced open source solution, namely Open Baton [31]. A common deployment involves a high volume server running its own Operating System (OS), e.g., Linux based kernel OS, and the installation of the various MANO components software packages. However, for simplicity and with no loss of generality, we consider the MANO software as a single entity where the failure of any of its subcomponents causes a system failure.

As depicted in Figure 7, on the hardware level, the MANO model is identical to the NFVI-PoP. On the software level, the model is similar to the VNF model having the OS and the MANO software components instead of the VM and the VNF software, respectively. The MANO is considered unavailable when there are no tokens in *MANO_OK* place. Due to these similarities, a detailed description is omitted.

A set of numerical values regarding failure and repair intensities and coverage probabilities, retrieved from previous literature [9], [15], [16], [28], are presented in Table I. These are hereafter referred to as baseline parameters.

C. End-to-end Service Availability by Level Merging

The remaining step is to evaluate the end-to-end service availability by merging the structure function and minimal-cut sets from Section V-A with the individual elements availability computed using the SAN models in Section V-B. In particular, since we make use of minimal-cut sets, we consider system unavailability.

Imposing the connectivity requirements for a correct service delivery, identified in Section V-A, and expressing the structure function in the form of *minimal product-of-sums* we obtain all the possible sets of service elements (network and

NFV), i.e., minimal-cut sets, who's failure will generate a service outage. As a result, if at least one of these sets is unavailable, the service will be unavailable. Therefore, the service unavailability will be given by the probability of the union of these sets. Note that the structure function does not regard any particular routing mechanism since it considers all the available paths satisfying the connectivity requirements. In addition, even though the logical service chains are the same for the different case studies, they represent different physical topologies of the chain. Such differences are reflected by having a distinct structure function for each of the case studies we investigate.

In order to merge the two levels, we make use of the *inclusion-exclusion principle*, which is a probabilistic technique to obtain the elements in a union of finite sets. Using the inclusion-exclusion principle on the structure function we can define the service unavailability as the probability of the union of all minimal-cut sets.

$$U_{NS} = P\left(\bigcup_{i=1}^{n} C_i\right) = \sum_{k=1}^{n}(-1)^{k-1} \sum_{0\neq I\subseteq[n],|I|=k} P\left(\bigcap_{i\in I} C_i\right)$$

where $C_1, C_2, ..., C_n$ are the minimal-cut sets and $P(C_i)$ is the probability of set C_i.

To compute the probability of the intersection of minimal-cut sets we just need to know the unavailability of the individual elements composing the minimal-cut set, since in the structural analysis we assume that the failures of these elements are independent. As a result, the probability of the intersection is given by the product of the probabilities of minimal-cut sets which in turn are given by the product of the probabilities of the single elements belonging to the set. In our case, such probabilities represent the elements unavailability and we compute them by using the proposed SAN models defined in Section V-B.

For assessing the service unavailability of each case study, we select the minimal-cut sets with cardinality lower than five as principal-cut sets, because the probability of the intersection of minimal-cut sets with higher cardinality becomes negligible in comparison to the principle-cut sets. This is because almost all the probability mass is in the principle sets when elements unavailabilities are relatively small, i.e., order of 10^{-3} or smaller, as shown in our investigation (refer to Section VI). In this case, $P(C_1) \sim 10^{-3}$, $P(C_2) \sim 10^{-6}$, $P(C_3) \sim 10^{-9}$, and so forth. Therefore, the probabilities of the intersection of minimal-cut sets with cardinality higher than five will have a negligible effect. In addition, also the probability of intersection of higher cardinality minimal-cut sets with the probability of the principle-cut sets will be much smaller than the probability of the principle-cut sets.

Table II presents the distribution of the principal-cut sets for each case study. Observing the first three case studies, i.e., deploying the VNFs into different NFVI-PoPs, there is an increase of the principal-cut sets for each cardinality when spreading the VNF deployment into multiple NFVI-PoPs. In addition, for the same deployments, when an SDN-integrated network is considered, there is a further increase of the cut sets. On the other hand, the addition of redundancy decreases the number of minimal-cut sets for the smaller cardinalities, i.e., C_1 and C_2, and increases those with cardinality 3 and 4. We explore the impact of this increase in more details in the following analysis.

TABLE II
DISTRIBUTION OF MINIMAL-CUT SET FOR THE FIRST FOUR CARDINALITIES OVER THE DIFFERENT NFV DEPLOYMENTS.

	C_1	C_2	C_3	C_4	Sum (Total*)
Reference	5	63	16	0	84 (18,097,984)
2 NFVI-PoPs	6	74	20	0	100 (23,969,350)
3 NFVI-PoPs	7	85	24	0	116 (29,957,966)
SDN Reference	6	74	20	0	100 (19,727,900)
SDN 2 NFVI-PoPs	7	85	24	0	116 (24,947,306)
SDN 3 NFVI-PoPs	8	96	28	0	132 (30,557,922)
MANO redundant	4	45	50	161	260 (24,017,754)
VNF redundant	1	55	122	261	439 (73,600,881)
All redundant	0	35	122	414	571 (107,254,823)
SDN All redundant	0	43	122	415	580 (122,878,786)

*Over all C_i

VI. NUMERICAL EVALUATION

In this section, we present the numerical analysis that has been carried out to evaluate the NFV deployment across the network for different scenarios, i.e., VNF deployment locations, and the different levels of redundancy adopted by the NFV elements. The goal of our analysis is to investigate the effects of varying both elements unavailability and element's component failure intensities on the end-to-end NFV service, given the various NFV deployment case studies, NFV and network elements, and the variation of elements unavailability and element's component failure intensities. First, we identify the critical elements, involved in the service delivery, that mainly affect the end-to-end service availability. Afterward, we delve into the element's components aiming at identifying the critical ones which mostly impact the service unavailability.

Möbius [27] is a powerful software tool for system modeling and analysis as it offers formalism-independent solvers for the system evaluation of certain measures of interest, e.g. element unavailability. One type of solver integrated in the tool is a Discrete-Event Simulator (DES) [32]. The simulator allows the modeler to choose a variety of simulation execution parameters such as type of random generator, random seed, maximum/minimum batches, or simulation result accuracy through confidence intervals etc. In addition, it offers high flexibility in running multiple simulations at once which are very useful in case a multitude of scenarios are investigated. We use this simulator to derive the element's unavailability by solving the element's SAN models presented in Section V-B.

In this study, each element's baseline unavailability, presented in Table III, is derived through simulations of the individual *dynamic* SAN models with 95% confidence interval by utilizing the baseline parameters. As previously specified, we have assumed that the timed activities, having mean rates presented in Table I, follow an exponential distribution. In fact, as soon as the repair process is extremely short compared to the mean time between failures, their mean will dominate the impact on the element availability and the effects of the actual recovery distributions are marginal. We verified this "insensitivity" by evaluating the NFV elements with

TABLE III
ELEMENT'S BASELINE AVAILABILITY.

	Availability	Unavailability	95% Confidence Interval
Link	0.999911	$8.89 \cdot 10^{-5}$	+/- $1.34 \cdot 10^{-5}$
IP Router	0.9924	$7.55 \cdot 10^{-3}$	+/- $5.06 \cdot 10^{-4}$
SDN Switch	0.9970	$2.98 \cdot 10^{-3}$	+/- $5.33 \cdot 10^{-4}$
SDN Controller	0.99897	$1.02 \cdot 10^{-3}$	+/- $7.57 \cdot 10^{-4}$
VNF	0.99950	$4.94 \cdot 10^{-4}$	+/- $6.37 \cdot 10^{-4}$
MANO	0.99983	$1.68 \cdot 10^{-4}$	+/- $3.46 \cdot 10^{-5}$
NFVI-PoP	0.999951	$4.84 \cdot 10^{-5}$	+/- $1.85 \cdot 10^{-5}$

deterministic recovery processes and the their unavailability variation is almost none compared to the exponential case.

To evaluate the impact that variation of a certain element unavailability has on the end-to-end service unavailability, we use a scaling factor α_x for $x \in \{$Link, Router, MANO, NFVI-PoP, VNF, Switch, and SDN controller$\}$, which affects the baseline unavailability of the elements. Simulations have been carried out by considering a scaling factor α_x that varies within a range spanning: $\alpha_x \in \{10^{-i}\}$ for $i = -3, ..., 1$. For each simulation, we vary α_x while keeping the rest of the element's unavailability equal to their baseline values. To illustrate, for $\alpha_x = 1$ the x element unavailability equals its baseline unavailability and when $\alpha_x = 10$, the unavailability is increased by one order of magnitude, and vice-versa for $10^{-1}, 10^{-2}, 10^{-3}$. $\alpha_x = 1$ is what we consider the most likely value of these parameters which are computed by solving the relative SANs with failure and repair parameters retrieved from previous literature (refer to Table I). However, since there is an ongoing evolution of both hardware and software technologies, it is important to study the effects on the sensitivity of these parameters with the used potential range due to changes in technology. Therefore, the scaling factor range is introduced to capture this evolution and is intended to represent the foreseeable changes in the near years to come.

For presenting the results, we are looking at a 4-dimensional problem where one dimension is represented by the NFV deployments (see Table II), another one identifies the elements (network and NFV elements), another determines the range of the scaling factor, and the last one expresses the end-to-end service unavailability as a function of the previous three. Therefore, a compact and easily comparable representation of this is achieved by using pie-like polar plots which are divided into different sectors representing the various deployments. In each sector, the angle and radius show the service elements and service unavailability due to element's unavailability/component failure intensity variation imposed by the scaling factor, respectively.

A. Impact of element's availability

In this subsection, the effects of varying the unavailability of the network and VNF elements on the end-to-end network service are investigated. In addition, we compare the unavailability of an NFV-based service in the case of assuming a fault-free network.

Figure 8 shows the end-to-end network service unavailability when varying the scaling factor α_x for the cases when the SFC is deployed into a single, multiple or separate NFVI-PoPs,

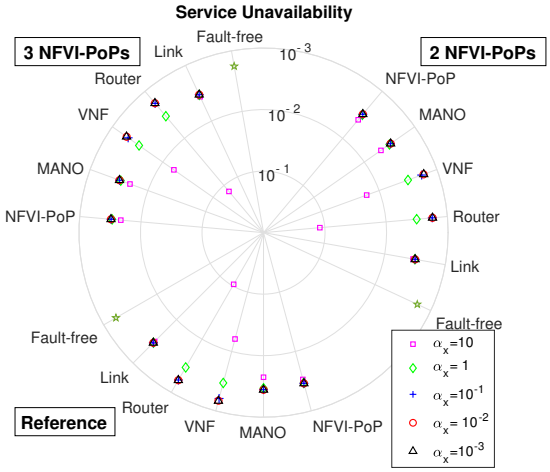


Fig. 8. Service unavailability of the three NFV deployments when varying element unavailability factor α_x.

and for the case when both links and IP routers are fault-free. Note that in this case, we consider a traditional network and not yet an SDN-integrated network. In the following, unless otherwise specified, all the case studies refer to a traditional network (TN).

An immediate observation is that the elements unavailability variation produces the same trends for all the three deployment cases. For the *Reference* deployment, given the baseline unavailabilities, the service unavailability reaches $2.9 \cdot 10^{-3}$. Any variation of link unavailability, either decreasing or increasing, does not significantly affect the service unavailability. On the contrary, the router unavailability may greatly impact the service unavailability. In particular, we observe that when the routers become less robust, i.e., $\alpha_{Router} = 10$, the service unavailability increases by more than one order of magnitude. On the other hand, when the router unavailability is reduced even by just one order of magnitude, the service unavailability is reduced to an extent that it approaches the fault-free network service unavailability ($1.71 \cdot 10^{-3}$ vs. $1.69 \cdot 10^{-3}$).

Regarding the NFV elements, the first observation we make is that for the MANO and NFVI-PoP, a decrease of their unavailability does not produce a noteworthy reduction of the service unavailability. The opposite is valid for the VNF where its unavailability reduction halves the service unavailability, i.e., from $2.9 \cdot 10^{-3}$ to $1.4 \cdot 10^{-3}$. In addition, we note that increasing the VNF unavailability by one order of magnitude, is accompanied with five times higher service unavailability. This can be explained by the fact that VNFs are three critical elements where the failure of any one of them produces a service outage. As a result, we can deduct that the VNF may play an important role in achieving both higher or lower service availability. Common to both network and NFV elements, decreasing their availability further, i.e., from 10^{-1} to 10^{-3}, does not bring an additional service unavailability reduction. In summary, the IP routers and VNFs represent the most critical network and NFV elements, respectively.

B. Impact of number of NFVI-PoPs

Deploying the VNFs, composing the SFC, into multiple or even separate NFVI-PoPs would definitively increase the path carrying service flows as they need to traverse more network elements. Accordingly, there would be an increase in the likelihood that more element's failures may impact the service availability. As a result, the system will be more vulnerable to failure events as highlighted by the increase of the principal-cut sets, presented in Table II, when the number of NFVI-PoPs hosting the SFC increases. Therefore, one can expect that service availability may be significantly deteriorated if for any reason the VNFs need to be geographically distributed. Surprisingly, spreading the VNFs into more or even completely separate NFVI-PoPs is followed with a very slight unavailability deterioration (in the order of 10^{-4}). More specifically, for the baseline element availabilities, employing two and three NFVI-PoPs results in a service unavailability of $3.17307 \cdot 10^{-3}$ and $3.39255 \cdot 10^{-3}$, respectively, versus $2.95355 \cdot 10^{-3}$ of the *Reference* case. The same difference is evidenced when varying the element's availabilities. The rationale behind is that despite the distribution of the VNFs into separate PoPs increases the low cardinality sets, the service availability is relatively insensitive to the VNF distribution in multiple NFVI-PoPs because in this case there is a higher number of available paths connecting the VNFs. The low cardinality sets are important but the high connectivity captured by the structure function and the associated flexibility in routing makes the placement effect insignificant. However, the outcome represent a good input to network administrators, as in cases an operator has to distribute the VNFs due to specific needs like resource shortages, the service availability will not be significantly affected. Note that there is an implicit premise that the network elements are homogeneous, i.e., have the same availability, and the presented outcome is also subject to the specific setting and network topology. In case a sparser network is considered the outcome may be otherwise.

To sum up, the splitting of the service chain into multiple NFVI-PoPs has a small effect on the unavailability due to an increase of the available paths connecting the splitted VNFs.

C. Impact of redundancy

In this subsection, we evaluate the impact of the redundancy of the NFV elements. To this end, we investigate the cases when only the MANO, the VNFs and when all the NFV elements are redundant, respectively.

In Figure 9, we illustrate the sensitivity analysis only for $\alpha_x = \{10^{-1}, 1, 10\}$, as for lower values there is not a significant variation. Deploying a redundant MANO decreases the service unavailability but the decrease is not significant (order of 10^{-4}). However, a redundant MANO provides adequate protection when the MANO unavailability increases, as opposed to the *Reference* case. Since the VNFs and routers are not protected with redundancy, an increase of their unavailability greatly affects the service by one and two orders of magnitude, respectively. In case only the VNFs are provided with redundancy, the service unavailability is further decreased reaching $1.1 \cdot 10^{-3}$ and it is sufficiently shielded against VNF

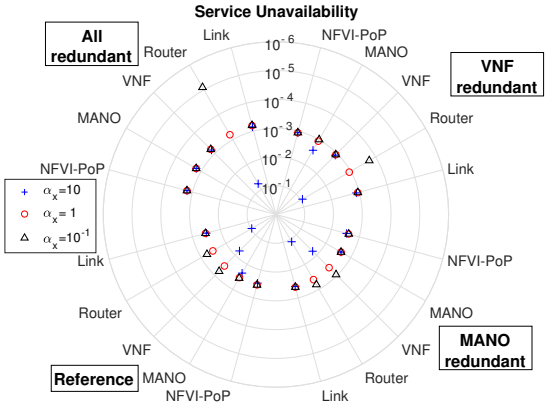


Fig. 9. Service unavailability for varying element unavailability factor α_x when considering NFV redundant elements.

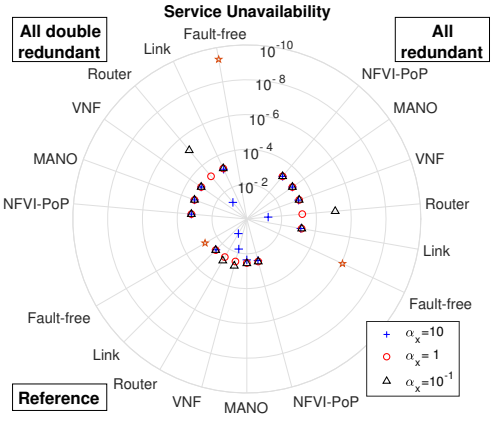


Fig. 10. Service unavailability for varying element unavailability factor α_x when considering single and double redundant NFV elements.

unavailability increments. Similarly, when all NFV elements are redundant, the service unavailability is further reduced compared to the previous two cases reaching a value of $6.3 \cdot 10^{-4}$. In this case, an increase of the VNF, NFVI-PoP or MANO unavailability does not impact the service unavailability as the redundant units provide an adequate protection. However, their unavailability reduction gives no effect at all.

Interestingly, the router may both greatly increase and reduce the end-to-end unavailability. A more robust IP router allows achieving a $7.09 \cdot 10^{-6}$ unavailability which represents target values expected by highly available NFV services, i.e., 5-nines availability [2], [5]. Moreover, we evaluate the case even when double redundancy, i.e., double VNFs, NFVI-PoPs and MANO, is deployed. Figure 10 shows the comparison of the sensitivity analysis for this deployment. We evidence that the additional unavailability reduction is rather negligible when a double redundant deployment is considered, i.e., an order of 10^{-5}. Curiously, very low service unavailability values are achieved only when the network elements are fault-free.

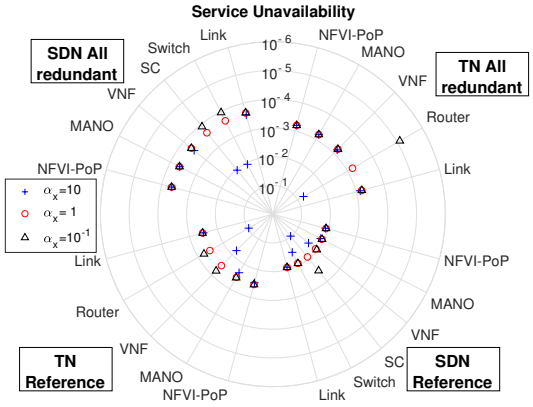


Fig. 11. Service unavailability of the traditional vs. SDN-integrated network for varying element unavailability factor α_x.

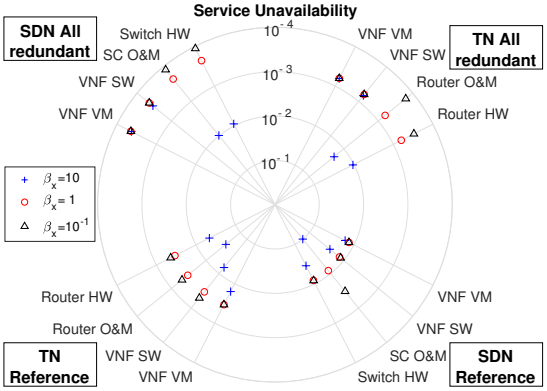


Fig. 12. Service unavailability of the traditional vs. SDN integrated networks for varying element's component failure intensity factor β_x.

Therefore, employing double redundant NFV elements does not produce compelling benefits unless the network elements are 'perfect'.

To summarize, for achieving five-nines availability, in addition to replicated NFV elements, the routers resiliency needs to be better than the nominal values used in this study.

D. Impact of SDN

When integrating an SDN network, there is an increase in the network connectivity requirements, presented previously in Section V-A, which is translated in an increase of the principle minimal cut-sets (refer to Table II). By having more principal-cut set, the SDN-integrated NFV service is expected to be more vulnerable in terms of service unavailability.

Figure 11 shows a comparison of the traditional and SDN-integrated network for the *Reference* and redundant deployments. As expected, the SDN-integrated service unavailability is higher compared to the traditional deployment. Specifically, for the *Reference* deployment, the SDN service unavailability reaches $1.2 \cdot 10^{-2}$ vs. $2.9 \cdot 10^{-3}$ of the TN case. This result is primarily due to the increased connectivity requirements imposed by the control plane of the SDN network.

Another observation regards the impact of the network nodes, i.e., routers or switches. For all the deployments, the robustness of the router is more relevant for the TN case than the switch for the SDN deployment. In the SDN case, it is the SC that has an impact magnitude similar to the routers for the TN case, thus representing the most crucial elements in an SDN-integrated network. Specifically, the increase/decrease of the scaling factor for the SC is accompanied with an increase/decrease of almost one order of magnitude of the service unavailability.

Surprisingly, when a redundant deployment is considered, the baseline service unavailability is three times less than the TN case. This result might look unexpected as it is the opposite compared to the non-redundant deployments, however, it is explained by the fact that the SC, being a critical component, is provided with redundancy which further decreases the baseline service unavailability. Nevertheless, an increasing SC unavailability may seriously degrade the service unavailability despite it makes use of a redundant unit. As a result, adopting a less robust SC may hinder the advantages created by the redundancy. Moreover, a similar trend is observed for the switches. An increasing switch unavailability is accompanied with more than one order of magnitude of service availability reduction. Differently, their unavailability reduction brings only a small service unavailability reduction. The opposite happens with the TN case, as a router unavailability reduction contributes to up to two orders of magnitude service unavailability drop. In brief, the SDN controller represents a critical element which may deteriorate the end-to-end service availability.

E. Impact of element's component failure intensity

In addition to the impact of the element's unavailability, we investigate the impact of each element components on the overall service unavailability. To this end, we investigate the impact of their relative failure intensities, presented in Table I. We use a scaling factor β_x for $x \in \{\text{HW}, \text{SW}, \text{O\&M}, \text{etc.}\}$, which affects the intensities of the relative element components, e.g., hardware, β_{HW}, software, β_{SW} or operation and management (O&M) etc. Simulations have been carried out by considering a scaling factor β_x that varies within a range spanning: $\beta_x \in \{10^{-i}\}$ for $i = -1, 0, 1$. For each simulation, we vary β_x while keeping the rest of the parameters as defined in Table I. Note that intensity variations are done one at a time.

Driven by the previous results, we present the sensitivity analysis of only the noteworthy components of the most critical elements, i.e., IP routers, VNFs, SDN switches, and SDN controller. Figure 12 shows the end-to-end service unavailability when varying the scaling factor of the most relevant failure intensities of those elements, for the TN and SDN *Reference* deployments with and without redundant elements.

For the non-redundant deployments, the largest impact on the service unavailability is due to the router and SC O&M failure intensity increments and such impact is similar for both TN and SDN deployments. On the other hand, when the O&M failure intensity decreases, a much larger relative gain is obtained by the SC compared to routers. The VNF software

TABLE IV
DISTRIBUTION OF PRINCIPAL-CUT SETS FOR THE DIFFERENT NFV ELEMENT PLACEMENTS AND THEIR RELATIVE SERVICE UNAVAILABILITY.

	C_1	C_2	C_3	C_4	Service Unavailability*	Reduction %
Reference	5	63	16	0	$2.953 \cdot 10^{-3}$	
Core	5	39	8	0	$2.443 \cdot 10^{-3}$	17.27%
SDN Reference	6	74	20	0	$1.218 \cdot 10^{-2}$	
SDN Core	6	50	8	0	$1.210 \cdot 10^{-2}$	0.65%
All redundant	0	35	122	414	$6.332 \cdot 10^{-4}$	
All redundant Core	0	35	73	97	$6.310 \cdot 10^{-4}$	0.34%
SDN All redundant	0	43	122	415	$2.264 \cdot 10^{-4}$	
SDN All redundant Core	0	43	69	122	$2.260 \cdot 10^{-4}$	0.17%

*Calculated with the element's baseline unavailabilities

presents a larger impact compared to the VNF VM component and such gain is slightly more pronounced for the TN case. This result is somehow expected since the VNF software failure intensity is much smaller than the VM intensity, while a reduction of the software intensity, i.e., $\beta_{\text{VNF}_{\text{SW}}} = 10^{-1}$, does not give a significant effect.

Regarding the redundant deployments, similarly to the previous outcomes, any increase on the VNF components failure intensity is suppressed by the redundancy protection. On the other hand, despite the SC is provided with redundancy, a higher O&M failure intensity may considerably degrade the service unavailability by more than one order of magnitude. Similarly, the SDN switch hardware system may play an important role in the overall service availability.

To summarize, for the traditional network, the hardware and O&M systems of routers represent critical components that may greatly impact the service availability. In an SDN network, the SC O&M software and switch hardware may have the largest impact on the end-to-end service availability.

F. Impact of NFV element placement

So far we have considered a presumably worst-case deployment where the NFV elements are placed on the edge of the backbone network. However, one might argue that the placement of the NFVI-PoPs, MANO and SDN controller, may significantly impact the service availability. To shed light on this, we examine the case where NFV elements, with and without redundancy, are deployed in the network nodes having the highest betweenness centrality [33]. These nodes are $\{23, \ldots, 28\}$ and represent the set of nodes that have the highest number of times they appear in the shortest path of any two other nodes. Figure 13 illustrates the TN and SDN-enabled NFV deployments for both redundant and non-redundant cases. A similar placement may be driven by the need of an operator to limit the service delay and/or the eventual additional path stretch due to the failover on the redundant element. The same notation, representing the previous use cases, followed by *Core* is used to identify the cases where the NFV elements are placed in the core nodes. To illustrate, *Core* represents the case of a traditional network with no redundant NFV elements and VNFs are running in the same NFVI-PoP. The MANO and the NFVI-PoP are connected to central nodes as depicted in Figure 13 (solid contour).

Table IV presents the distribution of the principal-cut sets for both the *Reference* and *Core* deployments together with their respective service unavailabilities. Observing the

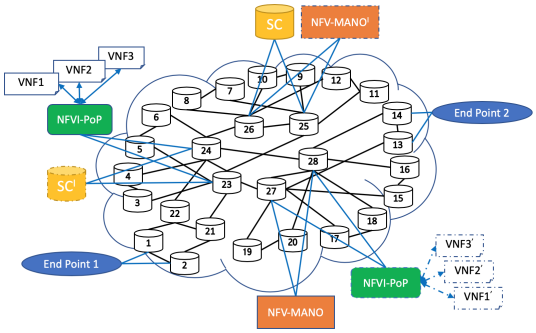


Fig. 13. SDN-integrated NFV deployments with NFVI-PoPs, MANO and SDN controller placed in the nodes with the highest betweenness centrality.

principal-cut sets for the non-redundant configurations, the *Core* deployments present a significant decrease in the number of minimal-cut sets of high cardinality suggesting that the service will be less vulnerable compared to the *Reference* cases. Despite this reduction, a minor decrease is achieved only for the TN deployment where the service unavailability is 17% less than the *Reference* deployment. This is because, given the element's baseline availabilities, on the *inclusion-exclusion principle*, the most impactful principal-cut sets, i.e., C_1, are not changed and the contributions from the other cardinalities are much smaller. In the SDN-enabled case, the service reduction is almost none and this can be explained by the fact that the SDN controller, being a crucial element, is still present in first cardinality sets which mostly impact the service unavailability. A similar trend is evidenced for the redundant cases where despite the principal-cut sets are more than halved, the service unavailability reduction is rather insignificant as a result of the fact that the lower cardinality sets C_2 remain unchanged. In addition to the *Core* deployment, we examined also the case where the MANO, PoP and SC are attached to the same two networking nodes having the highest betweenness centrality, i.e., nodes 23 and 24. We noticed that even in this deployment, the availability increase is not significant. Specifically, the unavailability is $1.208 \cdot 10^{-2}$ vs. $1.21 \cdot 10^{-2}$ of the *SDN Core*. This result is further evidence that it is the SC which brings a significant effect on the service availability regardless of the placement. To conclude, the placement of the NFVI-PoPs, MANO and SDN controller has a minimal effect on the overall service availability for the non-redundant architecture and almost none for the redundant architecture.

VII. CONCLUDING REMARKS

A comprehensive approach for the evaluation of end-to-end NFV-based service availability has been proposed. Through the formalized *two-level* availability model, we are able to capture both network topology structural dependencies and failure dynamics of the individual elements involved in the end-to-end service delivery. In addition, an extensive sensitivity analysis, for several case studies including traditional and SDN-integrated networks, aiming at identifying the main

critical elements has been carried out. The main outcomes include the following:

- in case a traditional network is employed, the most impactful elements are represented by the IP routers and VNFs composing the SFC. Adopting less robust routers and VNFs, compared to their baseline availabilities, may reduce the end-to-end service availability up to two orders of magnitude. Despite a small gain is obtained for more available routers and VNFs, adopting much more available routers and VNFs does not gain accordingly;
- deploying the VNFs into multiple or separate NFVI-PoPs does not significantly affect the service unavailability. In addition, the placement of the NFVI-PoPs, MANO and SDN controller does not reflect a remarkable impact;
- applying redundancy to NFV elements further decreases the service unavailability and brings adequate protection to any eventual increase of their unavailabilities. In addition, when such elements are redundant, making use of more robust router devices allows the service to reach target values like 5-nines availability;
- compared to a traditional network, an SDN-integrated solution brings additional challenges reflected in lower service availability. In an SDN network, the SDN controller is the most critical element which could even inhibit the advantages brought by the redundancy of the NFV elements. On the other hand, adopting a redundant SDN controller further decreases the service unavailability compared to a traditional network with redundant NFV elements;
- from an element's component perspective, the service is mostly affected by the router hardware and O&M failure intensity variations for both redundant and non-redundant NFV element deployments. Similarly, for an SDN-integrated network, high intensity of SC O&M software and switch hardware failures may significantly degrade the service unavailability.

To summarize, deploying redundant NFV elements like VNFs, MANO, and NFVI-PoPs contributes in lower service unavailability but network elements like IP routers may either severely degrade or significantly increase the overall service availability. Therefore, if 5-nines target figures are to be expected, in addition to NFV redundant elements, more reliable router hardware and O&M software architectures need to be employed.

REFERENCES

[1] G. N. ETSI, "Network Functions Virtualisation (NFV): Architectural Framework," *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.

[2] ETSI, "Reliability; Report on Models and Features for E2E Reliability," ETSI, Tech. Rep. GS REL 003 v1.1.2, 2016-07.

[3] 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, no. 1, pp. 11–33, 2004.

[4] N.-I. ETSI, "Network Functions Virtualisation (NFV); Network Operator Perspectives on NFV priorities for 5G," Tech. Rep., 2017.

[5] 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.

[6] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[7] K. S. Trivedi and A. Bobbio, *Reliability and availability engineering: modeling, analysis, and applications*. Cambridge Univ. Press, 2017.

[8] A. Gonzalez *et al.*, "Service availability in the NFV virtualized evolved packet core," in *GLOBECOM, 2015 IEEE*. IEEE, 2015, pp. 1–6.

[9] M. Di Mauro *et al.*, "Ip multimedia subsystem in an nfv environment: availability evaluation and sensitivity analysis," in *2018 IEEE NFV-SDN*. IEEE, 2018, pp. 1–6.

[10] B. Tola, G. Nencioni, B. E. Helvik, and Y. Jiang, "Modeling and evaluating nfv-enabled network services under different availability modes," in *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)*, March 2019, pp. 1–5.

[11] N. ETSI, "Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework," Tech. Rep., 2015.

[12] ITU-T E.860 (06/02), "Framework of a service level agreement," 2002.

[13] ETSI, "Network Functions Virtualisation, An Introduction, Benefits, Enablers, Challenges & Call for Action," *White Paper*, no. 1, pp. 1–16, 2012.

[14] D. S. Kim, F. Machida, and K. S. Trivedi, "Availability modeling and analysis of a virtualized system," in *Dependable Computing, 2009. PRDC'09. 15th IEEE Pacific Rim International Symposium on*. IEEE, 2009, pp. 365–371.

[15] R. d. S. Matos *et al.*, "Sensitivity analysis of server virtualized system availability," *IEEE Transactions on Reliability*, vol. 61, no. 4, pp. 994–1006, 2012.

[16] D. S. Kim *et al.*, "Availability modeling and analysis of a virtualized system using stochastic reward nets," in *Computer and Information Technology (CIT), 2016 IEEE International Conference on*. IEEE, 2016, pp. 210–218.

[17] M. Di Mauro *et al.*, "Service function chaining deployed in an NFV environment: An availability modeling," in *2017 IEEE CSCN*. IEEE, 2017, pp. 42–47.

[18] ——, "Availability modeling and evaluation of a network service deployed via NFV," in *International Tyrrhenian Workshop on Digital Communication*. Springer, 2017, pp. 31–44.

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

[20] I. N. ETSI, "ETSI GS NFV-REL 001 v1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.

[21] ——, "ETSI GR NFV-REL 007 v1.1.2: Network Function Virtualisation (NFV); Reliability; Report on the resilience of NFV-MANO critical capabilities," 2017.

[22] G. Nencioni *et al.*, "Availability modelling of software-defined backbone networks," in *DNS Workshop, 2016 46th Annual IEEE/IFIP International Conference on*. IEEE, 2016, pp. 105–112.

[23] P. Vizarreta *et al.*, "Qos-driven function placement reducing expenditures in NFV deployments," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

[24] H. Zhu and C. Huang, "Availability-aware mobile edge application placement in 5G networks," in *IEEE GLOBECOM*, 2017, pp. 1–6.

[25] M. Rausand and A. Høyland, *System reliability theory: models, statistical methods, and applications*. John Wiley & Sons, 2004, vol. 396.

[26] W. H. Sanders and J. F. Meyer, "Stochastic activity networks: Formal definitions and concepts," in *Lectures on Formal Methods and Performance Analysis*, ser. Lecture Notes in Computer Science, vol. 2090. Springer, 2001, pp. 315–343.

[27] "Möbius: Model-based environment for validation of system reliability, availability, security and performance," "https://www.mobius.illinois.edu", Accessed: 2019-03-31.

[28] G. Nencioni, B. E. Helvik, and P. E. Heegaard, "Including failure correlation in availability modeling of a software-defined backbone network," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1032–1045, Dec 2017.

[29] D. Cotroneo, L. De Simone, and R. Natella, "NFV-bench: A dependability benchmark for network function virtualization systems," *IEEE Trans. on Network and Service Management*, vol. 14, no. 4, pp. 934–948, 2017.

[30] N. F. S. de Sousa, D. A. L. Perez, R. V. Rosa, M. A. Santos, and C. E. Rothenberg, "Network service orchestration: A survey," *Computer Communications*, vol. 142-143, pp. 69 – 94, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366418309502

[31] "Open Baton: An open source reference implementation of the ETSI NFV MANO specification," "http://openbaton.github.io", Accessed: 2019-03-31.

[32] A. L. Williamson, "Discrete event simulation in the mobius modeling framework," Master's thesis, University of Illinois at Urbana-Champaign, 1998.

[33] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977. [Online]. Available: http://www.jstor.org/stable/3033543

Besmir Tola received the M.Sc. degree in Electronics and Telecommunication Engineering from the University of Siena (Italy) in 2014. In autumn 2015, he joined the IIK department at the Norwegian University of Science and Technology (NTNU) as a Ph.D. candidate in Telematics. In 2016 and 2018, he was a visiting researcher at the Nokia Bell Labs in Stuttgart (Germany), and UNINETT (Norwegian National Research and Education Network Operator), respectively, where he worked on dependability modeling and failure data processing, and analysis of cloud computing infrastructures and services. His current research interests include performance and dependability analysis of Cloud Computing, SDN, and NFV architectures.

Gianfranco Nencioni received the M.Sc. degree in Telecommunication Engineering and the Ph.D. degree in Information Engineering from the University of Pisa, Italy, in 2008 and 2012, respectively. In 2011, he was a visiting Ph.D. student with the Computer Laboratory, University of Cambridge, U.K. He was a Post-Doctoral Fellow with the University of Pisa from 2012 to 2015 and the Norwegian University of Science and Technology, Norway, from 2015 to 2018. He is an Associate Professor with the University of Stavanger, Norway, from 2018. His research activity regards modelling and optimization in emerging networking technologies (e.g., SDN, NFV, 5G, Network Slicing). His past research activity has been focused on energy-aware routing and design in both wired and wireless networks and on dependability of SDN and NFV.

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), the Department of Telematics and Department of information Security and Communication Technology. In the period 2009 – 2017, he has been Vice Dean with responsibility for research at the Faculty of Information Technology and Electrical Engineering at NTNU. 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 is on ensuring dependability in services provided by multi-domain, virtualised ICT systems, with activities focusing on 5G and SmartGrids.

PAPER E

Towards Carrier-Grade Service Provisioning in NFV

Yordanos T. Woldeyohannes, Besmir Tola, and Yuming Jiang

IEEE Proceeding of 15th International Conference on the Design of Reliable Communication Networks (DRCN)

Coimbra, Portugal, 2019

Towards Carrier-Grade Service Provisioning in NFV

Yordanos Tibebu Woldeyohannes, Besmir Tola, and Yuming Jiang
NTNU-Norwegian University of Science and Technology, Norway

***Abstract*—Network Function Virtualization (NFV) is an emerging technology that reduces cost and brings flexibility in the provisioning of services. NFV-based networks are expected to be able to provide carrier-grade services, which require high availability. One of the challenges for achieving high availability is that the commodity servers used in NFV are more error prone than the purpose-built hardware. The "de-facto" technique for fault tolerance is redundancy. However, unless planned carefully, structural dependencies among network nodes could result in correlated node unavailabilities that undermine the effect of redundancy. In this paper, we address the challenge of developing a redundancy resource allocation scheme that takes into account correlated unavailabilities caused by network structural dependencies. The proposed scheme consist of two parts. In the *first part*, we propose an algorithm to identify nodes that can be highly affected by a node failure because of their network structural dependency with this node. The algorithm analyzes such dependencies using a recently proposed centrality measure called *dependency index*. In the *second part*, a redundancy resource allocation scheme that places backup network functions on nodes considering their dependency nature and assigns the instances to flows optimally is proposed. The results show that not considering the network structural dependency in backup placement may significantly affect the service availability to flows.The results also give insights into the trade-off between cost and performance.**

I. Introduction

Middleboxes or Network Functions (NFs) are widely utilized for various purposes such as improving security and network performance. The traditional approach of having a dedicated purpose-built hardware per NF has been shown to be inefficient and expensive [1]. Network Function Virtualization (NFV) alters this inflexible architecture by decoupling the software of NFs from the hardware and run the NFs on virtualized environment such as virtual machines (VMs) or containers. NF instances can then be created on the fly depending on the traffic and the network state [2]. The VMs and containers of the NFs are usually hosted on commercial off-the-shelf (COTS) hardware, which are comparatively less expensive than the purpose-built hardware. A service in NFV is typically composed of a set of NFs that are chained in some specific order, also known as service function chaining.

Carrier-grade services such as telecommunication services require high-level of availability reaching five-nines (99.999%) or more [3] [4]. Achieving this level of availability in NFV networks is challenging due to a number of reasons including: lower dependability of COTS servers, correlated failures or unavailabilities, and state management:

COTS availability: Legacy telecommunication networks have achieved carrier-grade service availability by using purpose-built hardware. In NFV, the purpose-built hardware is replaced by COTS hardware, which is usually more error-prone [4]. To achieve the same level of availability using COTS, NFV needs to build the resilience into software [5].

Correlated failures / unavailabilities: Although most literatures assume that failures are independent, correlated failures or unavailabilities are often common in real systems [6], [7]. For example, the failure of a node may result in the unavailability of other nodes because of their structural dependencies on this node [8]. The de-facto technique for boosting availability is redundancy. However, redundancy may become ineffective due to correlated failures or unavailabilities.

State management: A large number of NFs such as Deep Packet Inspection (DPI) and Network Address Translator (NAT) are stateful. Stateful NFs preserve service states, such as, TCP connection state and the mapping between IP addresses about ongoing connections [9]. Typically, NFs need to maintain 10-100s of state variables that are per-flow or shared across flows [10]. Backup instances of stateful NFs need to have updated state information to ensure successful failover and service continuity [11], [12].

In this paper, we make a step forward towards carrier-grade service provisioning in NFV, by proposing a novel redundancy resource allocation scheme where two crucial challenges are addressed. (1) One challenge is *how to factor the inherent network structural dependency among nodes into redundancy resource allocation*. (2) The other challenge is *how to efficiently place and allocate backup instances for service chain of flows.*

To tackle the first challenge, an algorithm that measures the dependency among network nodes and identifies nodes that have a high-level of structural correlation is proposed. The dependency among nodes of a network is quantified by using a centrality measure called *node dependency index* [13]. Here, there is an intuition, which is, a backup NF should not be placed at a node that may also become unavailable when the primary NF's node fails. For the second challenge, an optimization model that aims to efficiently place redundant NFs and assign backup NFs to service chains of flows is proposed. In this model, flows are assigned backup instances following the $1 : m$ active-standby redundancy mode, with which, every flow can tolerate the failure of any one of the NF instances on the NF chain [11]. In addition, following the intuition, redundant instances are not placed on backup nodes that are structurally correlated with the primary nodes. Furthermore, to efficiently utilize resources, backup NFs are shared by flows and only the required number of instances are created.

Most of the existing works on redundancy allocation in NFV

based networks focus on providing two-nines or three-nines service availability [14], [15]. only a few consider carrier-grade service availability [16], [17]. Both [16] and [17] allocate only on-site redundancies. However, to guarantee carrier-grade service availability it is important to also have backups distributed geographically [4]. In addition, [17] considers only the failure of the physical nodes while not the NF applications and assumes that all nodes have the same availability, and [16] focuses on the failure of VMs assuming similar availability and failure independence between VMs.

Our proposed scheme differs from the existing works in a number of ways. First, our scheme considers the effect of network structural dependency. Second, it takes into account both physical hardware failures and NF software failures with different availability values. Third, it can be used to allocate both on-site and off-site backups in an optimal way. Moreover, in our proposed scheme, in addition to availability, delay performance is also taken into consideration, such that the delay that flows experience after failover can be kept within the requirement of the flows.

The specific contributions in this paper include:

- An algorithm that identifies the set of nodes that have strong structural correlation using a centrality measure called node dependency index.
- A redundancy allocation scheme that finds the optimal number and placement of backup nodes and NF instances and assigns the instances to flows.

The paper is organized as follows. In Section II, the system model is described. Section III discusses in brief the node dependency index centrality measure. In Section IV, the algorithm proposed for identifying the nodes that have high-level of structural correlation is explained. The proposed redundancy allocation scheme is presented in Section V, followed by the results in Section VI. Finally, Section VII presents the concluding remarks.

II. System Model

The system considered is a network of nodes and links and is represented as a graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ represents the set of links. Nodes hosting NFs that are being utilized by the primary service chains of flows are called *primary nodes*. Nodes that can be used for backup are called *backup nodes*. $\mathcal{B}$ denotes the set of backup nodes and $\mathcal{P}$ the set of primary nodes. Backup NFs are hosted on backup nodes.

A node hosting backup instances can be a shared or dedicated backup node. A shared backup node is a node that is being used both as primary and backup node. This type of nodes reserve some resources to be used as backup while hosting NFs that are utilized by the primary service chains. Dedicated backup nodes are nodes that are used only to host backup instances.

Each flow f is defined by a source and destination node pair, which are respectively denoted as s_f and d_f. The service required by flow f is represented by a service chain, $\overrightarrow{S}_f = (S_f^1, S_f^2 \dots S_f^{g_f})$. The service chain is an ordered series of network functions, where S_f^1 is the first NF, S_f^2 is the second NF needed and so on. It is assumed that a flow is already assigned a primary service chain. The variable $p_{f,g}$ indicates the primary node p that is serving flow f's g^{th} service. A backup instance of an NF of type v requires k_v number of cores and can be a backup to up to C_v^b number of flows. For each flow f, there is an availability requirement on its service $\overrightarrow{S}_f$, denoted as A_f. A service $\overrightarrow{S}_f$ is considered available is either the primary or one of the backup service chains is available.

III. Structural Dependency Measures

The *node dependency index* $DI(i|n)$ measures the average level of dependency node i has on node n in connecting with the other nodes of the network [13]. $DI(i|n)$ is calculated from the path dependency index $DI(i \to j|n)$, which measures the dependency the path between nodes i and j has on node n. $DI(i \to j|n)$ is defined as:

$$DI(i \to j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } A_{ij}^{-n} = 1 \\ 1 & \text{if } A_{ij}^{-n} = 0, \end{cases} \tag{1}$$

where I_{ij} is an information measure, which is equal to the inverse of the shortest path distance hop counts, denoted as d_{ij}, between nodes i and j, i.e. $I_{ij} = 1/d_{ij}$. I_{ij}^{-n} is the information measure between nodes i and j after the deactivation of node n. The binary variable A_{ij}^{-n} measures the availability: $A_{ij}^{-n} = 1$ if node i can reach node j after the deactivation of node n and zero otherwise. The node dependency index is defined as:

$$DI(i|n) = \frac{1}{N-2} \sum_{j \in \mathcal{N}^{-n}/i \neq j} DI(i \to j|n). \tag{2}$$

$DI(i|n)$ measures the average dependency that node i has on node n. $DI(i|n) = 1$ if node i cannot connect with the other nodes, $DI(i|n) = 0$ if node i does not experience any connectivity problem and $0 < DI(i|n) < 1$, if node i experiences connectivity problem but is still able to connect to at least one other node, all after the failure of node n.

IV. Structurally Correlated Nodes

While failure independence is commonly assumed when studying availability, recent studies have demonstrated the existence of correlated failures and the pronounced effect of geographical adjacency [7] [18], [19]. Nevertheless, it has also been recognized that it is difficult to discover or predict dependencies among failures [6], [7] [18], [19]. To tackle this challenge, in this section, a novel approach is proposed to identify nodes that are inherently correlated due to the network structure. This information lays a foundation for the proposed redundancy allocation scheme that will be detailed in Section V.

A. Algorithm

The failure of a node may result in the unavailability of other nodes. For example, in a data-center network, the failure of a Top-of-Rack switch will result in the unavailability of all the servers located in the same rack. The proposed algorithm uses the node dependency index to measure the dependency among nodes and identify the nodes that have high structural correlation. From the definition of the node dependency index, if node i has high-level of dependency on node n, the failure of node n might result in the unavailability of node i.

Definition 1: *Critical nodes of node* i, denoted as $\mathcal{C}(i)$, are nodes that node i is highly dependent on, where node i is said to be highly dependent on node n if $DI(i|n)$ is above a given threshold t_{DI},

$$\mathcal{C}(i) = \{n | DI(i|n) > t_{DI}, n \in \mathcal{N}\}. \quad (3)$$

If $\mathcal{C}(i)$ is empty, node i is independent of the other nodes of the network so has no critical node. For example, in a fully mesh network, all nodes are independent of each other as the failure of one does not affect the connectivity of the others.

To find the set of nodes that have strong structural correlation with a primary node i, some intuitive observations are used which include:

First-level dependency

- Node i has a high probability of experiencing a correlated failure with its critical nodes in $\mathcal{C}(i)$ as the failure of these nodes might lead to the unavailability of node i. Thus, node i should not use the nodes in $\mathcal{C}(i)$ as a backup.
- Node i should not also use as a backup those nodes that depend on it highly. Since the failure of node i might also result in the unavailability of these nodes.

In brief, *a primary node i and its backup nodes should not depend on each other.* This can be regarded as the *first-level dependency* among nodes.

Second-level dependency

- Node i should not use as a backup nodes that depend heavily on its critical node. This is because if the unavailability of node i is due to the failure of its critical node, the other nodes that depend heavily on the critical node might also be unavailable.

The algorithm for finding a set of nodes, $\hat{\mathcal{B}}_i$, which are structurally correlated with a primary node i is shown in Algorithm 1. The algorithm starts by finding the critical nodes of a primary node. The critical nodes of node i, $\mathcal{C}(i)$, are inserted into the set $\hat{\mathcal{B}}_i$. Then, nodes that have a high-level of dependency on node i are included to the set $\hat{\mathcal{B}}_i$ (line 9). For the second-level dependency, all the nodes that are highly dependent on the critical nodes of node i will be included to $\hat{\mathcal{B}}_i$. The threshold, t_{DI}, should be assigned values that are between zero and one. If it is set to a very low value that is close to zero, the set $\mathcal{C}(i)$ will include a large number of network nodes. Therefore, it should be assigned a relatively large or medium values such as 0.5.

Algorithm 1 Heuristic for finding structurally correlated nodes

```
1:  G(N, L) → the network graph.
2:  B̂_i → set of nodes that are structurally correlated with
    node i.
3:  t_DI → threshold for high dependency
4:  for i ∈ N do
5:      Find C(i) using t_DI
6:      Insert C(i) to B̂_i
7:      for j ∈ N /i do
8:          if i ∈ C(j) then
9:              Insert j to B̂_i
10:         if j ∈ C(i) then
11:             for k ∈ N /i do
12:                 if j ∈ C(k) then
13:                     Insert k to B̂_i
14: return B̂_i
```

V. Redundancy Allocation Scheme

Some of the features considered in the design of the redundancy allocation scheme include:

- *Correlated failures*: To tolerate correlated failures caused by network structural dependency, backup NFs of a flow are not placed on nodes that are structurally correlated with the primary nodes of the flow.
- *State:* Stateful NFs can have states that are per-flow or shared across flows. Flows using the same primary stateful NF instance will be assigned to the same backup instance since they rely on a state shared among them.
- *Delay vs utilization:* To efficiently use network resources, minimal number of backup instances are created. However, this can increase the end-to-end backup chain delay of flows. To solve this problem, the scheme finds a balance between minimizing the backup chain delay, which is the delay flows will experience after failover, and the resource utilization.

A. Formulation: All-One

The first model considered is the All-One model in which *all* the services of a chain are assigned one backup that is a 1:1 active-standby mode. It is assumed that *one* backup node will be used to backup all the NFs of a flow, later on this assumption will be relaxed. The redundancy allocation is then formulated as an Integer Linear Program (ILP).

The redundancy allocation has three main objectives: (I) *to minimize the number of backup instances created*, (equation (4)), (II) *to minimize the number of backup nodes used*, (equation (5)), and (III) *to minimize the backup chain delay*, (equation (6)).

$$\text{minimize} \quad \sum_{\forall b \forall v} z_v^b \quad (4)$$

$$\text{minimize} \quad \sum_{\forall b} q^b \quad (5)$$

$$\text{minimize} \quad \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b) \quad (6)$$

The weighted sum method is used to combine the three objective functions into one by using equal unit weights. For positive weights, the optimal solution of the single-objective representation is also a Pareto optimal solution of the multi-objective problem [20]. The All-One optimization model is given us:

All-One:

$$\text{min.} \sum_{\forall b \forall v} z_v^b + \sum_{\forall b} q^b + \sum_{\forall b \forall f} (D(s_f, b) i_f^b + D(b, d_f) i_f^b). \quad (7)$$

s.t.

$$1 - (1 - \sum_{\forall b} i_f^b * A^b \prod_{g=1, v=S_f^g}^{g_f} A_v)(1 - A_{s_f}^p) \geq A_f \quad , \forall f : A_{s_f}^p < A_f \quad (8a)$$

$$\sum_{\forall f, \forall g / S_f^g = v} y_{f,g}^b \leq C_v^b \quad , \forall b, v \quad (8b)$$

$$\sum_{\forall v} z_v^b * k_v \leq K^b \quad , \forall b \quad (8c)$$

$$y_{f,g}^b = 0, \quad , \forall f, g \in \{1..g_f\}, \forall b \in \hat{\mathcal{B}}_p / p \in \mathcal{P}_f \quad (8d)$$

$$y_{f,g}^b = y_{f',g'}^b \quad , \forall f, f' \in \mathcal{F} / p_{f,g} = p_{f',g'}, S_f^g = S_{f'}^{g'}, T(S_f^g) = 1, \forall b, g, g' \quad (8e)$$

$$\sum_{\forall b} y_{f,g}^b = 1 \quad , \forall f : A_{s_f}^p < A_f, g \in \{1..g_f\} \quad (8f)$$

$$\sum_{\forall b} i_f^b = 1 \quad , \forall f : A_{s_f}^p < A_f \quad (8g)$$

$$y_{f,g}^b \leq z_v^b \quad , \forall b, f, g \in \{1..g_f\}, v = S_f^g \quad (8h)$$

$$q^b = \max_{v \in \mathcal{V}} z_v^b \quad , \forall b \quad (8i)$$

$$i_f^b = y_{f,g}^b \quad , \forall f, b, g \in \{1..g_f\} \quad (8j)$$

The constraints are classified into five group, which are *availability*, *capacity*, *correlated failure*, *state* and *assignment* constraints. Constraint (8a) belongs to the *availability* group and ensures that flows' availability requirement is fulfilled by the primary and backup NFs assigned. Constraints (8b) and (8c) are *capacity* constraints for the backup NF instances and backup nodes respectively. For each flow, constraint (8d) prohibits the usage of backup nodes that have high structural *correlation* with the primary nodes of the flow. Constraint (8e) is a *state* constraint, which makes sure that flows using the same primary instance of a stateful NF are assigned to the same backup instance.

The other constraints belong to the *assignment* group. *All* the services of a flow have a backup (constraint (8f)) and only *one* backup node hosts all the backup NFs of a flow (constraint (8g)). A flow is mapped to a backup instance on a given backup node only if the node is hosting the NF type (Constraint (8h)). Constraint (8i) identifies backup nodes that are hosting instances. A flow is assigned to a backup node only if it is using backup instance hosted on the node (Constraint (8j)).

TABLE I: Symbols used in formulation

Notation	Meaning
K^b	the number of cores available on backup node b.
$p_{f,g}$	primary node p is used by flow f's g^{th} service.
$D(p,b)$	the delay between nodes p and b.
A^b	the probability that backup node b is available.
$\hat{\mathcal{B}}_p$	set of backup nodes that have high structural correlation with node p.
k_v	the number of cores needed to instantiate NF type v.
$T(v)$	binary variable to show if NF type v is stateful ($T(v) = 1$) or not ($T(v) = 0$).
C_v^b	the maximum number of flows that an instance of NF type v hosted on node b can be a backup to.
A_v	the probability that the application software of a network function of type v is available.
A_f	the availability requirement of flow f.
s_f, d_f	source and destination nodes of flow f respectively.
$\overrightarrow{S}_f = (S_f^1, S_f^2 \dots S_f^{g_f})$	service chain of flow f.
A_{sf}^p	the availability of the primary service chain of flow f.
$\mathcal{P}_f$	the set of primary nodes used by flow f.
Decision variables	
$y_{f,g}^b$	a binary decision variable, to indicate if backup node b is used as a backup for flow f's g^{th} service.
z_v^b	an integer decision variable to indicate the number of backup instances of NF type v hosted on backup node b.
i_f^b	a binary variable that indicates if backup node b is used by flow f or not.
q^b	a binary variable to indicate if backup node b is hosting backup NF instances.

B. Formulation: All-Any

The "All-One" ILP model given above uses one backup node to backup all the NFs of a flow. This constraint is relaxed so that a flow can use one or more backup nodes. This model will be referred as "All-Any" since *all* of the services of a flow are backed up and a flow can use *any* number of backup nodes. The objective function for minimizing the backup chain delay needs to be modified as

All-Any:

$$\text{minimize} \sum_{\forall b \forall f} (D(s_f, b) y_{f,1}^b + D(b, d_f) y_{f,g_f}^b + \sum_{\forall b' \in \mathcal{B}, g=1}^{g_f - 1} D(b, b') y_{f,g}^b * y_{f,g+1}^{b'}). \quad (9)$$

Since a flow might use more than one backup node, the backup chain delay will include the delay between the backup nodes. All the constraints except three (8a, 8g and 8j) of the All-One model will also be included in the All-Any model. The three constraints will be replaced by constraints (10, 11 and 12) respectively.

$$1-(1-\prod_{\forall b}\max(1-i_f^b, i_f^b A^b \prod_{g=1}^{g_k}\max(1-y_{f,g}^b, y_{f,g}^b A_v))) \\ (1-A_{s_f}^p) \geq A_f \qquad \forall f : A_{s_f}^p < A_f \quad (10)$$

$$\sum_{\forall b} i_f^b \geq 1 \qquad \forall f : A_{s_f}^p < A_f \quad (11)$$

$$i_f^b = \max_g(y_{f,g}^b) \qquad \forall b, \forall f : A_{s_f}^p < A_f \quad (12)$$

Constraint (10) guarantees that the availability requirement of flows is satisfied by the primary and backup instances, which might be hosted on different backup nodes. One or more backup nodes are assigned to a flow (constraints (11)). A flow is assigned a backup node provided that it is using atleast one backup instance hosted on the node (constraint (12)). This model is an Integer Non-linear Program (INLP) because of the non linearity of equation (10). To decrease the complexity of the model, the non-linear constraint is approximated by a linear equation.

1) Linear approximation: The availability constraint is approximated by a linear lower bound function.

Theorem 1. *The probability that all E entities of a set $\mathcal{E}$ will be available is lower bounded by $1-\sum_{e=1}^{E} U_e$, where every entity $e \in \mathcal{E}$ fails independently with probability A_e and U_e is the unavailability of entity e.*

Proof: The probability that all the E entities will be available (A_t) is a product of the availability of each of them. That is

$$A_t = \prod_{e=1}^{E} A_e. \quad (13)$$

By definition, the availability of entity e, $A_e = 1-U_e$, where U_e is the unavailability of e. Thus,

$$A_t = \prod_{e=1}^{E}(1-U_e). \quad (14)$$

By expanding the product,

$$\prod_{e=1}^{E}(1-U_e) = 1-\sum_{e=1}^{E} U_e + \sum_{e=1}^{E-1} U_e * U_{e+1} + o(n), \quad (15)$$

where $o(n)$ represents the higher order terms. Usually, unavailability $U << 1$ so the product and the higher order terms can be ignored. We will then have

$$\prod_{e=1}^{E}(1-U_e) \geq 1-\sum_{i=1}^{E} U_e. \quad (16)$$

As a result,

$$A_t \geq 1-\sum_{e=1}^{E} U_e, \quad (17)$$

which concludes the proof. ■

For example, if there are two entities with availability $A_1 = 0.9$ and $A_2 = 0.99$, then $A_t = 0.891$. Using the linear approximation, $U_1 = 0.1, U_2 = 0.01$, so $A_t = 0.89$. Thus, the linear approximation is a lower bound to the actual availability value. Applying this linear approximation, equation (10) can be approximated by:

$$1-(1-(1-\sum_{\forall b} i_f^b(U^b + \sum_{g=1}^{g_k} y_{f,g}^b U_v)))(1-A_{s_f}^p) \\ \geq A_f \qquad ,\forall f : A_{s_f}^p < A_f, \quad (18)$$

where U^b and U_v are the unavailabilities of backup node b and backup NF v respectively. The lower bound approximation is conservative so flows availability requirement will not be violated. Equation (18) is not linear since it has a term that is a product of two variables, i_f^b and $y_{f,g}^b$. Let variable $r_{f,g}^b = i_f^b * y_{f,g}^b$,

$$1-(1-(1-\sum_{\forall b} i_f^b U^b - \sum_{\forall b}\sum_{g=1}^{g_k} r_{f,g}^b U_v))(1-A_{s_f}^p) \\ \geq A_f \qquad ,\forall f : A_{s_f}^p < A_f. \quad (19)$$

Equation (19) is a linear equation of the variables i_f^b and $r_{f,g}^b$. Thus, the non-linear inequality constraint in equation (10), can be substituted by equation (19) and the constraint $r_{f,g}^b = i_f^b * y_{f,g}^b$. However, since the variables i_f^b and $y_{f,g}^b$ are binary, their product can easily be linearized by substituting it with the following linear equations,

$$r_{f,g}^b <= i_f^b \\ r_{f,g}^b <= y_{f,g}^b \\ r_{f,g}^b >= i_f^b + y_{f,g}^b - 1. \quad (20)$$

Thus, the equivalent ILP model of the All-Any model will have constraints (19) and (20) instead of the non-linear availability constraint and all the other linear constraints of the All-Any INLP model.

C. Allocating more than one backup chain

The All-One and All-Any models assign one backup for each of the NFs of a flow's service chain. However, to guarantee the high availability of carrier-grade services, it might be necessary to allocate more than one backup chain. The following simple example is used to show case this. Consider a flow that has two services long chain. The primary chain of the flow is 90% available and the flow requires to be 99.999% available. The backup nodes and NF applications are 99% and 99.9% available respectively. Thus, after being allocated backup instances that are hosted on the same backup node, the flow will only be 99.88% (2'9s) available. Thus, a second backup chain is needed to reach the required 99.999% (5'9s) availability.

Algorithm for assigning more than one backup chain: Backup instances are to be allocated for a set ($\mathcal{F}$) of flows. The proposed models assign one backup chain. More than one backup chains are allocated to a flow sequentially one after the other. That is the first backup chain is allocated and if the availability requirement of the flow is not satisfied then

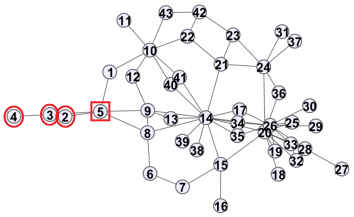


Fig. 1: GEANT network: Example of structurally correlated nodes

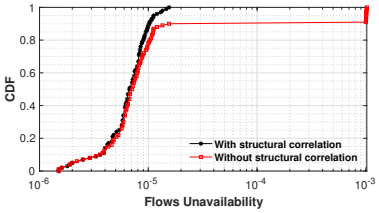


Fig. 2: Effect of not considering structural correlation: CDF of the unavailability

the second backup chain is assigned and so on. One problem with using the models directly for assigning backup chains sequentially is that if the availability requirement of a flow $f \in \mathcal{F}$ cannot be satisfied by one backup chain, the models will be infeasible. To solve this issue, for all of the flows in the set, it is checked whether their availability requirement can be satisfied while being assigned to the least available backup node. If not, the availability requirement of the flow will be downgraded to the next availability class, e.g., from 99.999% to 99.99% or from 99.99% to 99.9%. The original availability requirement of the flow is saved and the flow will be marked as a flow that might need more than one backup.

Then, the first backup chain will be allocated by using the models. The state of the network (including the placement of backup instances and their capacity) will then be updated. If the availability requirement of a flow is not satisfied by the first backup chain, then the same process will be used to allocate the second backup chain. Two variables are introduced to transfer the state of the network between the different rounds of backup chain allocations. These are oZ_v^b, the number of backup instances of type v already created on node b, and oC_v^b, the currently available capacity of an existing instance of type v hosted on node b. For this algorithm, in the models z_v^b will be replaced by $z_v^b + oZ_v^b$ and C_v^b will be replaced by $C_v^b + oC_v^b$. This is done to be able to use instances created previously in the current round of the backup allocation.

In case it is not possible to allocate backups due to shortage of resources, flows will be rejected. Resource shortage can occur at any round of the backup chains assignment. For example, when a flow is allocated a second backup chain. In this case, the flow has already been assigned one backup chain. However, the availability requirement of the flow is not yet satisfied. In cases like this, the flow will be rejected and the resources already assigned to it will be released to be used by other flows.

VI. Results

The performance of the proposed scheme is analyzed by conducting a number of experiments. The models are solved by using a commercial solver, CPLEX, together with Matlab for transferring updated network state information between different rounds. The GEANT network, Fig. 1, which is the pan-European research and education network, is used as a test network topology [21].

Eight nodes of the network are chosen to be the ingress/egress nodes. The ingress/egress nodes are a bottleneck for achieving high availability since the failure of one of them leads to service unavailability for customers using it. To avoid this, these nodes are paired to provide "dual homing", whereby one node is a backup for the other and vice versa. Twenty-two nodes of the network are assumed to be backup nodes (in shared or dedicated mode). Each backup node has 4 CPU cores to be used by the backup instances it hosts. The rest of the nodes are dedicated primary nodes. The availability of the nodes is assumed to be uniformly distributed between $0.99 - 0.999$, whereas NF instances have an availability between $0.999 - 0.9999$ and an NF instance can be a backup for up to 10 flows.

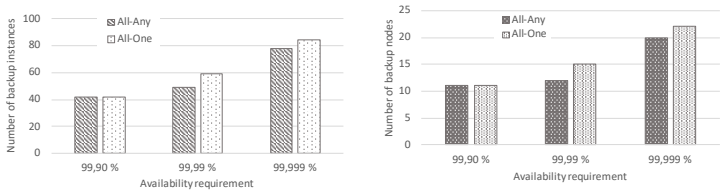
Flows are assumed to require a service chain that is composed of two NFs. NFs of a chain are randomly chosen out of the set of five services, which are Firewall, DPI, IDS, Proxy, NAT. Flows are assigned primary chains by using ClusPR algorithm [2]. The availability requirement of a flow is selected from the set $\{0.999, 0.9999, 0.99999\}$.

A. Structurally correlated nodes

Example of nodes that have high structural correlation, which are identified by the proposed algorithm are highlighted in Fig. 1. Nodes 2-4 have a high probability of experiencing a correlated failure with node 5 due to their dependencies. This is because, the failure of node 5 will lead to the unavailability of these nodes as well.

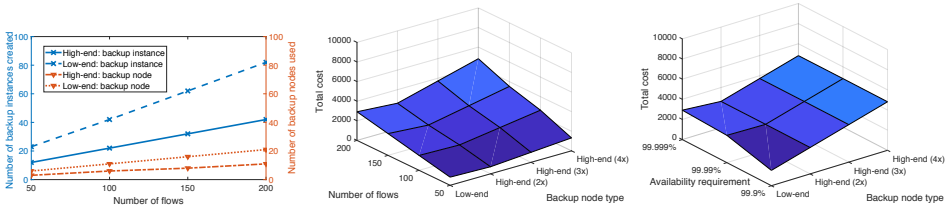
1) Effect of not considering structural correlation: In this section, a simple experiment is carried out to showcase the effect of not considering the structural correlation among nodes in the backup instance placement decision making. The baseline algorithm from [16] is used to decide the number of backup instances needed. It is assumed that the availability of a node is 0.999 (3'9s). According to the baseline algorithm, theoretically, one backup for each of the NFs is enough to meet the 99.999% availability requirement of a single service function chain containing two NFs. The primary and backup NF host nodes of a chain are randomly chosen from the network. When structural correlation is considered, the backup nodes of a chain will not have strong correlation with the primary nodes.

The availability of 100 flows is measured by conducting ten million simulation runs. In each simulation run, the state of each node, i.e., failed (0) or up (1), is randomly generated



(a) Backup instances (b) Backup nodes

Fig. 3: Number of backup NF instances and nodes utilized for acheiving different availability requirements



(a) Backup nodes and instances (b) Cost for achieving 99.999% availability (c) Cost for 200 flows

Fig. 4: "High-end" vs "Low-end": Number of backup NF instances created and backup nodes used (a), total cost for achieving 5'9s (b), total cost for 200 flows (c).

from Bernoulli distribution using the node's availability. The network is then updated considering the nodes state. Finally, the availability of the backup and primary chains is checked by verifying the availability of the host nodes of the chain's NFs and the path between them. The chain is said to available if either the primary or backup chain is available. A CDF of the unavailability of the 100 flows is shown in Fig. 2. When structural correlation is not considered around 10% of the flows have low availability, 3'9s and 2'9s.

B. Resource utilization

The number of backup instances created and nodes utilized for fulfilling the availability requirements of 200 flows for different availability requirements are shown in Fig. 3a and 3b respectively.

When the flows have 99.9% availability requirement, 42 backup instances are created and 11 backup nodes are used to host the instances by both All-Any and All-One models. The availability requirement of all of the flows is able to be fulfilled with 1:1 active-standby backup for each of the NFs of a chain. When the availability requirement of the flows increases to 99.99%, 49 backup instances are created by the All-Any model and 59 by the All-One model. For some of the flows, 1:1 active-standby was not enough to reach to the required availability therefore, a 1:2 active-standby, where one NF of a chain has two backup instances, is required. As a result, more backup instances are created. When comparing the two models, the All-One model created more instances than the All-Any model. This is because of the All-One model's constraint that forces a flow to use backup instances hosted only on the same node. For achieving 5'9s (99.999%) availability, most of the flows need 1:2 backup protection.

C. Effect of the type of backup nodes

In this section, the effect of using highly available COTS servers versus COTS with lower availability, referred to as "High-end" and "Low-end" respectively is analyzed. The "High-end" servers are assumed to be 99.9% available and the "Low-end' servers 99% available. The relative importance between the cost of installation of host server hardware and the cost of installation of the network function software license is chosen to be 100:10 for "Low-end" servers as in [22]. For the "High-end" servers, the cost is 200:10 if the "High-end" servers are twice (2×) more expensive, and 300:10 if they are 3× more expensive compared to the "Low-end".

Figure 4a and 4b show the number of backup instances created, nodes utilized and the total cost for fulfilling 99.999% availability requirement of different number of flows. For the "Low-end" servers, 1:2 backup have to be applied to reach the 5'9s requirement. Using the "High-end" servers, the availability requirement is fulfilled with 1:1 active-standby redundancy. Therefore, fewer backup instances are created when using "High-end" servers. However, the "High-end" servers are more expensive than the "Low-end" servers. The total cost spent for fulfilling the availability requirement of the flows depends on the relative cost of the servers. It is more economical to use "High-end" servers if their cost is not more than 2× the cost of the "Low-end" servers. If the cost of installation of the "High-end" servers is 3× or more compared to the "Low-end" servers, the total cost spent will be more than that spent using the "Low-end" servers.

Figure 4c shows the total cost for serving 200 flows when their availability requirement changes. The 1:1 active-standby protection is enough for meeting the 99.9% avail-

TABLE II: Effect of including delay in the objective function

Objective	Average delay (hops)	Worst-case delay (hops)	# Backup instances
Without delay	4.68	12	12
With delay	0.44	2	15

ability requirement. Thus, it is economical to use the "Low-end" servers. For both 99.99% and 99.999%, if the "High-end" servers are $2\times$ more expensive or less, then it is more economical to use the "High-end" servers.

D. Effect of minimizing the backup chain delay

The backup chain delay, in terms of number of hops, observed when the objective includes minimizing the end-to-end delay and the amount of resources used is compared with the case when the objective is only to minimize the resources used (i.e., instances created and nodes utilized).

Table II shows the results of the comparison for 50 flows that have 99.9% availability requirement. When the objective is to minimize the resource utilization, 12 instances are created. Compared to the primary chain, the backup chain delay is 4.68 hops longer on average. In the worst case, a flow's backup chain is 12 hops longer than its primary one. When the objective function is to minimize both the total backup chain delay and the resource utilization, the average backup chain delay is only 0.44 hop counts longer and the worst-case observed delay is 2 hops longer. In this case, 15 backup instances are created.

VII. Conclusion

In this paper, a redundancy resource allocation scheme that tolerates correlated failures caused by network structural dependency is proposed. The scheme identifies the sets of nodes that have strong structural correlation using a novel algorithm that is based on the node dependency index centrality measure. The experimental results demonstrate that not taking into account the structural correlation among nodes in backup instances placement decision making considerably affects the availability of flows. The results also give insights into the trade-off between cost and system performance.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2018.

[3] D. Collins, *Carrier grade voice over IP*. McGraw-Hill New York, 2003, vol. 2.

[4] 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.

[5] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[6] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz, "Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems," in *Reliable Distributed Systems, 2002. Proceedings. 21st IEEE Symposium on*. IEEE, 2002, pp. 362–367.

[7] S. Nath, H. Yu, P. B. Gibbons, and S. Seshan, "Subtleties in tolerating correlated failures in wide-area storage systems." in *NSDI*, vol. 6, 2006, pp. 225–238.

[8] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, "Heading off correlated failures through independence-as-a-service." in *OSDI*, 2014, pp. 317–334.

[9] S. Woo, J. Sherry, S. Han, S. Moon, S. Ratnasamy, and S. Shenker, "Elastic scaling of stateful network functions," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 299–312.

[10] J. Khalid, A. Alsudais, E. Keller, and F. Le, "Paving the way for NFV: Simplifying middlebox modifications using statealyzr." in *NSDI*, 2016, pp. 239–253.

[11] N. ISG, "Network function virtualisation (NFV)-resiliency requirements," *ETSI GS NFV-REL*, vol. 1, p. v3, 2016.

[12] P. X. Sherry, Justine Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. Martins, S. Ratnasamy, L. Rizzo *et al.*, "Rollback-recovery for middleboxes," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 227–240.

[13] Y. T. Woldeyohannes and Y. Jiang, "Measures for network structural dependency analysis," *IEEE Communications Letters*, 2018.

[14] J. Fan, C. Guan, Y. Zhao, and C. Qiao, "Availability-aware mapping of service function chains," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[15] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in nfv with the cost-efficient redundancy scheme," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[16] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*. IEEE, 2017, pp. 1–10.

[17] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *Globecom Workshops (GC Wkshps), 2015 IEEE*. IEEE, 2015, pp. 1–7.

[18] A. J. Gonzalez, B. E. Helvik, J. K. Hellan, and P. Kuusela, "Analysis of dependencies between failures in the UNINETT IP backbone network," in *Dependable Computing (PRDC), 2010 IEEE 16th Pacific Rim International Symposium on*. IEEE, 2010, pp. 149–156.

[19] B. Chun and A. Vahdat, "Workload and failure characterization on a large-scale federated testbed," *Intel Research Berkeley, Tech. Rep. IRB-TR-03-040*, 2003.

[20] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and multidisciplinary optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[21] *GEANT the pan-european research and education network,*, 2018 (accessed Septemper 10, 2018). [Online]. Available: http://www.geant.net.

[22] P. Vizarreta, M. Condoluci, C. M. Machuca, T. Mahmoodi, and W. Kellerer, "QoS-driven function placement reducing expenditures in nfv deployments," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

PAPER F

CoShare: An Efficient Approach for Redundancy Allocation in NFV

Yordanos T. Woldeyohannes, Besmir Tola, Yuming Jiang, and K.K. Ramakrishnan

Submitted to IEEE/ACM Transactions on Networking
A preliminary version is available on arXiv: https://arxiv.org/pdf/2008.13453.pdf

CoShare: An Efficient Approach for Redundancy Allocation in NFV

Yordanos Tibebu Woldeyohannes*, Besmir Tola*, Yuming Jiang*, and K. K. Ramakrishnan†
*Norwegian University of Science and Technology, NTNU, Trondheim, Norway
†University of California Riverside, California, USA

***Abstract*—An appealing feature of Network Function Virtualization (NFV) is that in an NFV-based network, a network function (NF) instance may be placed at any node. This, on the one hand, offers great flexibility in redundancy allocation; on the other hand, it makes the allocation a unique and difficult challenge. One particular highlight is that there is inherent correlation among nodes due to the structure of the network, implying that special care is needed. To this aim, a novel approach, called *CoShare*, is proposed. Originally, its design takes into consideration the effect of network structural dependency. In addition, to efficiently make use of resources, CoShare proposes the idea of *shared reservation*, where multiple flows may be allowed to share the same reserved backup capacity at an NF instance. Furthermore, CoShare factors in the heterogeneity in nodes, NF instances and availability requirements of flows in the design. The results from a number of experiments conducted using realistic network topologies show that CoShare is able to meet diverse availability requirements in a resource-efficient manner, requiring less resource overbuild than using the idea of *dedicated reservation* commonly adopted for redundancy allocation in NFV.**

I. Introduction

Network softwarization is transforming how networks are designed and operated to deliver specialized / innovative services and applications. Network Function Virtualization (NFV) has emerged as the key driver of network softwarization, promising, among other aspects, full network automation, flexible service provisioning, and cost reduction [1]. NFV is also considered as a key enabler for the new generation of communication networks such as 5G cellular networks [2] where carrier-grade services are demanded. However, the successful adoption of NFV in production networks is associated with new challenges. One of them is to ensure high availability of services provided by an NFV-based network [3]–[5].

The "de-facto" technique for an NFV-enabled network to achieve high availability for its services is through allocation of redundant / backup network function (NF) resources to compensate for the failures of primary NFs [3]. It has been shown that merely provisioning primary NF service chains is insufficient to meet the high-availability, carrier-grade, requirements for services [6], [7]. In different virtualization technologies considered so far, redundancy is provided in the form of hot-standby replicas of NF instances. Typical solutions such as VMware Fault Tolerance [8] and the more recent NFV system-level framework [9] envision the instantiation of a *dedicated backup instance*, which runs *on a separate node*. However, such solutions can be resource inefficient, as each NF requires at least two instances (one primary and one backup). In addition, the protection is at the instance or node level, while the services provided by NFV-based networks are typically at the flow level in the form of network function (NF) chains.

Recent work on meeting the service availability requirements at the NF chain level in NFV include [10]–[13]. However, as discussed in Sec. VII in more detail, the existing results are typically obtained under restrictive setups or assumptions, e.g., hosting backup instances of a service chain on the same node or having dedicated capacity reserved for the backup instances. In addition, a unique characteristic of NFV is that *an NF instance may be hosted at any node in the network*. This appealing characteristic offers additional flexibility in redundancy allocation in NFV, i.e., the ability to decide *where to place the backup instances*, in addition to *how to assign backup instances* to form backup service chains for flows [14], [15]. However, previous work has focused mainly on how to assign backup instances, without considering where to place the backup instance. More importantly, the problem of jointly considering both has not been considered.

Furthermore, a fundamental piece of information of the network, which is its topology or structure, is not leveraged, ignoring its potentially significant impact on redundancy allocation in NFV. In particular, the failure of a critical node can cause difficulty for the other nodes to reach each other due to the structure of the network [16]. As a consequence, such inherent dependencies, called *network structural dependencies* [17], [18], among nodes imply that the impact of one node's failure on the services provided by the network may significantly differ from that of another node's failure.

In this paper, a novel redundancy allocation approach for NFV-based networks, called CoShare, is proposed. The contributions of CoShare are several-fold. First, CoShare explores the flexibility offered by the unique characteristic of NFV, assigning backup NF instances to nodes meticulously in order to avoid the potential concurrent unavailability of the primary and backup service chains due to correlated failures caused by network structural dependencies. For this purpose, the information centrality measure called *node dependency index* [17], [19] is exploited to identify correlation among nodes (Cf. Sec. III). The correlation information identified is made use of in both backup instance placement (§ IV-B) and assignment (§ V-A). To the best of our knowledge, CoShare is the first redundancy allocation approach for NFV-based networks, which explicitly considers the network structure-

caused correlation among nodes in the design.

In addition, CoShare proposes to improve resource utilization efficiency by exploiting *shared reservation*, where multiple service chains that are not susceptible to simultaneous service interruption are allowed to share the same reserved backup capacity of an NF instance. In the literature, the general idea of sharing backup resources has long been exploited for backup allocation to improve resource utilization in various types of networks, e.g. MPLS, IP, and optical mesh networks [20]–[24]. However, there is a fundamental difference. In traditional network settings, the locations of the backup resources are typically fixed in the network, while in an NFV-based network, owing to the flexibility offered by NFV, they cannot or need not be assumed a priori. In other words, the placement of backup NF instances can have a significant impact on the decision of how they can be shared, making the problem new and challenging. Moreover, CoShare takes into account the heterogeneity in the availability requirement across flows in allocating back up NF resources, together with the node heterogeneity in supporting NFs, where the heterogeneity in node and instance availability levels is also considered.

In brief, CoShare places backup NF instances and assigns them to form backup service chains to meet diverse availability requirements of flows by *jointly considering the network structural dependency-caused impact* (Sec. III, Sec. IV-B, Sec. V-A), *the heterogeneity in nodes, instances and availability requirements* (Sec. IV-B, Sec. V-A), and *the efficiency in resource utilization* (Sec. V-C). The main contributions of this paper are summarized as follows:

- A novel redundancy allocation approach for NFV, called CoShare, is proposed, where an information centrality measure is exploited to identify the inherent correlation among nodes due to the structure of the network (§ III), based on which, backup NF instances are placed (§ IV-B) and assigned (§ V-A) to form the backup NF chains.
- The proposed heuristics of CoShare adopts a new idea, referred to as *NF shared reservation*, to achieve higher efficiency in resource utilization (§ V-C). Specifically, under this idea, independent flows with disjoint primary service chains are allowed to share common reserved capacity at an NF instance for fault-tolerance purpose.
- The placement and assignment of NF instances take into account the heterogeneity of nodes, instances, and required service availability levels in deciding where to place backup NF instances (§ IV-B) and how to assign NF instances to form the backup chains (§ V-A).

The remainder is organized as follows. In Sec. II, the system model and the redundancy allocation problem are described. Sec. III introduces the node dependency index and describes how to identify the network structure-caused correlation among nodes. In Sec. IV, the idea for placement of backup NF instances is presented. In Sec. V, the proposed NF instance assignment scheme is explained. Sec. VI presents the results. The related work is discussed in Sec. VII. Finally, concluding remarks are made in Sec. VIII.

II. The System Model and Problem Definition

A. The Network Model

We consider the services provided by an NFV-based network, with the network represented as an undirected connected graph $G(\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{L}$ the set of links. Each flow f has a source (s_f) and a destination (d_f), and has a traffic rate denoted as λ_f in packets per second (pps). The flow f is identified by the source-destination pair (s_f, d_f), where s_f and d_f are in $\mathcal{N}$. We call such source and destination nodes "end nodes" in this paper and assume that the end nodes are not involved in hosting NF instances. Each of the other nodes may have multiple CPU cores to host NF instances. *An NF instance can be hosted on any node in $\mathcal{N}$, which is not a source or destination of a flow, and has enough available resources*. Keeping with typical deployment approaches [25], [26], we assume that a CPU core, if allocated, is dedicated to a single NF instance. An NF instance may be allocated one or more CPU cores.

The network service provided to flow f is represented by an NF chain $\overrightarrow{S}_f$, i.e., a set of network functions $(S_f^1, S_f^2 \dots S_f^{g_f})$ that are performed in the specified order, where g_f denotes the service chain length. An NF instance v is assumed to require k_v number of cores and has μ_v processing capacity (i.e., the amount of traffic the instance can process per unit time). An NF instance may process multiple flows whose service chains include that corresponding NF. It is required that $\sum_{f \in \mathcal{F}^v} \lambda_f \leq \mu_v$, where $\mathcal{F}^v$ denotes the set of flows processed by v.

It is assumed that for each flow, its service chain has already been allocated using an NFV resource allocation algorithm, e.g., ClusPR [26], where, however, the availability aspect of the service has not been taken into account. We term this allocated service chain as the *primary chain* of the flow.

B. The Availability Model

The network service provided to each flow f has an availability requirement A_f^r. For practical reasons, the failure impact of the flow's source and destination nodes is excluded. In addition, to simplify the representation and analysis, we focus on the impact of (hardware) node and (software) NF instance failures on the availability, and assume that nodes and instances fail independently. Without loss of generality, we classify flows according to their availability requirements, and within each class c, the availability requirement is the same, denoted as A_c. Reflecting the typical classification of availability requirements, as seen in the specification by the European Telecommunication Standard Institute (ETSI) [27], we consider three levels of service availability with regard to NFV resiliency (unless specified otherwise).

We are interested in *meeting the availability requirements of flows through redundancy*, by allocating additional NF instances. More specifically, if the primary chain cannot provide the required A_f^r, *backup chain(s)* are created to improve the availability level of the service and meet the requirement. The service to a flow is considered available if either the primary or one of the backup service chains is available.

Let A_n denote the availability of node n, and A_v the availability of NF instance v. For the primary chain p_f of flow f, we use A_f^p to denote its availability. For a backup chain b, its availability, denoted as A_f^b, is calculated as:

$$A_f^b = \prod_{g=1}^{g_f} A_{v(g)} \prod_{n \in \mathcal{N}^b(f)} A_n \tag{1}$$

where $\mathcal{N}^b(f)(\subset \mathcal{N})$ denotes the set of nodes that host NF instances of the backup chain b, and $v(g)$ the instance on b for the g-th NF of the service chain. The availability of the primary chain is computed in a similar manner. If the backup chains and the primary chain are independent, the overall service availability A_f is given by a parallel combination:

$$A_f = 1 - (1 - A_f^p) \prod_{i=1}^{h_f} (1 - A_f^{b_i}) \tag{2}$$

where h_f denotes the number of backup chains used.

C. The Redundancy Allocation Problem

For an NFV-based network, the redundancy allocation problem has three aspects to consider, which are,

- *(C1) Deciding the number of backup instance for each NF:* The needed number for each NF depends on the availability requirements of flows, the availability and capacity characteristics of both the nodes and the NF instances, as well as the topology of the network. When the availability requirements are stringent while nodes and/or NF instances have low availability values, more than one backup chain may have to be allocated [7], [28].
- *(C2) Placing the backup instances:* Similarly, the placement of backup instances is also influenced by the same factors. For example, the placement of a backup instance should not only comply to anti-affinity constraints [3] to avoid common failure modes, but it should also take into account the node's resource capabilities: placing a backup instance in a given node may happen only if the node has sufficient resources, e.g., CPU cores.
- *(C3) Assigning instances to form backup chains for flows:* This concerns the assignment of instances to each flow to form backup service chains to meet the flow's availability requirement. Similar considerations apply. For instance, a flow can be assigned backup instances only if those instances have sufficient resource capacity to accommodate the flow.

The three considerations discussed above are entangled, and together with the special NFV characteristic that any node may host instances of any NF, makes redundancy allocation in NFV both unique and challenging. To have a better overview of the problem and to achieve efficient resource utilization while providing redundancy allocation, we formulate it as an optimization problem as follows:

$$\begin{aligned} \text{Given:} \quad & G(\mathcal{N}, \mathcal{L}) \\ \text{Minimize}_{\forall(\beta,\alpha)} \quad & N_v(\beta, \alpha), \qquad \forall v \in \mathcal{V} \\ \text{Such that} \quad & A_f(\beta, \alpha) \geq A_f^r, \quad \forall f \end{aligned} \tag{3}$$

where β and α respectively denote the adopted NF backup instance placement and assignment strategies, $\mathcal{V}$ is the set of NFs involved in providing the services, $N_v(\beta, \alpha)$ denotes the number of instances of NF v and $A_f(\beta, \alpha)$ is the achieved availability for flow f, under the strategies β and α.

For simplicity, only the topology and the availability conditions / constraints are included in (3). In our earlier work [29], a more complete version of the problem including the constraints can be found. In addition, an Integer Linear Program (ILP) model has been developed in [29] to solve the problem. However, the complexity of the problem is formidable, since it is an NP-hard problem as implied in the formulation (see [12], [30] for additional instances). As a consequence, when the network is large, solving the problem optimally in limited time is difficult. For this reason, a heuristic approach, called CoShare, is proposed in this paper to address (C1) – (C3). The heuristic is introduced in detail in Sec. IV and Sec. V.

It is worth highlighting that *redundancy allocation for a backup chain should try to avoid sharing risk of failures with the primary service chain, i.e., the failure of any primary instance or its hosting node should have minimal impact on the backup service chain.* This is crucial and is also the basis for applying (1) and (2). However, even though the nodes and instances may be independent as individual systems, they are inherently correlated due to network structural dependence: the failure of one node may cause the unreachability of other nodes if these nodes have strong network structural dependency with the failed node (Cf. Sec. III). A novel idea of CoShare is to explicitly take into account the inherent correlation due to the network topology in the redundancy allocation problem (3).

III. Identifying Correlation among Nodes Due to Network Structural Dependence

A. Network Structural Dependency Measure

The inherent structural dependencies among nodes imply that the impact of one node's failure on the services provided by the network may significantly differ from the failure of another node. To reflect this difference, several measures have been proposed in the literature [17], [19]. For the NFV redundancy allocation problem, a key is to choose the proper nodes to place the backup instances. To this aim, the *node dependency index* [17], [19] is adopted.

The *node dependency index* $DI(i|n)$ measures the average level of dependency that node i has on node n in connecting to the other nodes of the network [17]. Specifically, $DI(i|n)$ is calculated from the path dependency index $DI(i \to j|n)$, which measures the dependency that the path between nodes i and j has on node n. $DI(i \to j|n)$ is defined as

$$DI(i \to j|n) \equiv \begin{cases} I_{ij} - I_{ij}^{-n} & \text{if } A_{ij}^{-n} = 1 \\ 1 & \text{if } A_{ij}^{-n} = 0, \end{cases} \tag{4}$$

where I_{ij} and I_{ij}^{-n} are the information measures between nodes i and j before and after the deactivation of node n [31]. Specifically, they are defined as:

$$I_{ij} = 1/d_{ij}; \qquad I_{ij}^{-n} = 1/d_{ij}^{-n}$$

with d_{ij} and d_{ij}^{-n} denoting, respectively, the length, in terms of hop count, of the shortest path between nodes i and j before and after the disabling of node n [31]. The binary variable A_{ij}^{-n} measures the reachability of node j from node i given that node n has failed: $A_{ij}^{-n} = 1$ if node i can reach node j after the deactivation of node n and 0 otherwise.

The node dependency index is defined as [17] [1]:

$$DI(i|n) = \frac{1}{N-2} \sum_{j \in \mathcal{N}^{-n}/i \neq j} DI(i \to j|n). \tag{5}$$

where $\mathcal{N}^{-n} = \mathcal{N} - \{n\}$, i.e. the set of nodes excluding n.

It can be verified: $0 \leq DI(i|n) \leq 1$. For the two extreme cases, $DI(i|n) = 0$ tells that i does not experience connectivity problem with removal of n, while $DI(i|n) = 1$ implies that i is unable to connect with any of the other nodes after n's failure. Fundamentally, a higher $DI(i|n)$ value indicates higher dependence of i on n due to the network structure.

Remark: It is intuitive that a longer path with more network elements on the path is subject more to unavailability than a shorter path with fewer elements. This intuition motivates us to use (4) as the basis to measure how the reachability between two nodes depends on another node. Essentially, $DI(i \to j|n)$ quantifies the extent that this reachability from i to j is affected by the failure of node n, and hence the extent that $i \to j$ depends on n. Note that, except for (4), the other ideas of CoShare do not rely on the specific definition of $DI(i \to j|n)$, and hence could be readily applicable when other definitions for $DI(i \to j|n)$ are preferred.

B. Network-Structurally Correlated Nodes

As discussed above, even though individual nodes may fail independently, such a failure can affect the communication of other nodes in the network due to the inherent network structural dependence. From the definition of the node dependency index, i.e., (5), if node i has a higher-level dependency on node n, the failure of node n will result in greater difficulty for node i to reach the other nodes in the network.

We introduce $\mathcal{C}(i)$ as *the set of critical nodes of node* i which node i highly depends on. Node i is said to highly depend on a node $n \in \mathcal{C}(i)$, or in other words, n is critical to i, if $DI(i|n)$ is above a given threshold t_{DI}:

$$\mathcal{C}(i) = \{n \mid DI(i|n) > t_{DI}, n \in \mathcal{N}^{-i}\} \tag{6}$$

If $\mathcal{C}(i)$ is empty, it means that i is not highly dependent on the other nodes. For example, in a full mesh network, all nodes are structurally independent of each other as the failure of one node does not affect the connectivity among the others.

[1]In [17], $N-1$ is used in the denominator, but since there are only $N-2$ choices for $j \in \mathcal{N}^{-n}/i \neq j$, the more intuitive $N-2$ is adopted in (5). Note that, this change does not affect the resulting ranking of nodes.

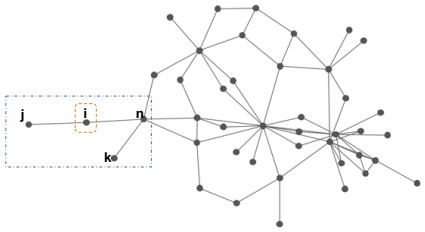


Fig. 1: Network-structurally correlated nodes with node i

Algorithm 1 Finding network-structurally correlated nodes

Input: $G(\mathcal{N}, \mathcal{L})$, t_{DI}
Output: $\hat{\mathcal{B}}_i$

1: Find $\mathcal{C}(i)$ using (6)
2: Insert $\mathcal{C}(i)$ to $\hat{\mathcal{B}}_i$
3: **for** $j \in \mathcal{N}^{-i}$ **do**
4: **if** $i \in \mathcal{C}(j)$ **then**
5: Insert j to $\hat{\mathcal{B}}_i$
6: **if** $j \in \mathcal{C}(i)$ **then**
7: **for** $k \in \mathcal{N}^{-j}$ **do**
8: **if** $j \in \mathcal{C}(k)$ **then**
9: Insert k to $\hat{\mathcal{B}}_i$
10: return $\hat{\mathcal{B}}_i$

It is worth highlighting that the network structural dependence relation between two nodes, i and j, has two directions, i.e., $DI(i|j)$ – dependence of i on j and $DI(j|i)$ – that of j on i. As a consequence, to minimize the influence of network structure-caused correlation, such that node i can be used as a backup for node j and vice versa, we should avoid $j \in \mathcal{C}(i)$ or $i \in \mathcal{C}(j)$. We call this the *first-level dependency* among nodes. To elaborate, Fig. 1 shows an example, where nodes n and j have first-level dependency with node i. In particular, while $n \in \mathcal{C}(i)$ is critical to i, j is network-structurally highly dependent on i, i.e., $i \in \mathcal{C}(j)$.

In addition, as easily seen in Fig. 1, a critical node n of i may also be critical to another node k i.e., $n \in \mathcal{C}(i)$ and $n \in \mathcal{C}(k)$. In this case, both nodes i and k depend on the same node n. As a result, the failure of such a critical node, e.g. n, may result in the unavailability of those nodes that depend on it, e.g., i and k, hence presenting a structural correlation that we refer to as the *second-level dependency* among nodes. An implication of this is that: we should avoid using k as a backup for node i, even though k is not in $\mathcal{C}(i)$.

Based on the above analysis, Algorithm 1 presents the pseudo code of the algorithm for finding the set of nodes, denoted as $\hat{\mathcal{B}}_i$, which are network-structurally correlated with node i. $\hat{\mathcal{B}}_i$ is initially empty, the algorithm starts by finding the set of nodes based on the first-level dependency in both directions (Lines 1-2 and Lines 3-5 respectively). Then, nodes that have the second-level of dependency described above are added (Lines 6-9).

IV. Placement of Backup NF Instances

This section focuses on estimating the needed numbers of backup NF instances and deciding where to place them. Originally, we take network structural dependence into account.

A. Estimating Numbers of Backup NF Instances

The number of backup instances for each NF, needed to satisfy the service availability requirements of flows, is influenced by several factors, such as the availability and length of the primary chains and the availability and capacity of the NF instances, in addition to their availability requirements. In addition, flows with higher availability requirements might need more than one backup chain while lower service availability requirements may be fulfilled with only one backup chain [7], [29], [32]. Further due to the heterogeneity in node availability, instance availability, and service availability requirements of flows, finding the needed numbers of backup NF instances is not trivial. In CoShare, we use the following approach to estimate such numbers.

Specifically, we first estimate the number of backup chains needed to fulfill the availability requirement of flows in each class, based on which, the number of needed backup instances is then calculated. For the former, the estimation assumes that each NF instance is hosted at a different node and the backup chains and the primary chain are node disjoined. Then, the number of needed backup chains for a flow in class c, denoted as h_c, is estimated as:

$$h_c = \min_{x \in \mathcal{Z}^+} \{x | 1 - (1 - \min_{f \in \mathcal{F}_c} A_f^p)(1 - \tilde{A}_c^b)^x \geq A_c^r\} \quad (7)$$

with

$$\tilde{A}_c^b = (\min_{n \in \mathcal{N}^b} A_n \min_{v \in \mathcal{V}} A_v)^g$$

where $\mathcal{F}_c$ represents the set of flows in class c, A_f^p the availability of flow f provided by the primary chain, and A_c^r the availability requirement for flows in class c. $\tilde{A}_c^b$ may be interpreted as the availability of a backup chain for a class c flow, where $\mathcal{N}^b \subset \mathcal{N}$ denotes the set of nodes that have the capacity to host backup NF instances, $\mathcal{V}$ the set of NFs, A_n the availability of a node n, A_v the availability of an instance of NF v, g the maximum NF chain length of flows in the availability requirement class c. Since to satisfy the required availability more than one backup chain may be required, h_c is estimated from (7), taking into account the parallel effect of these backup chains.

Next, the number of backup instances of each NF v, denoted as z_v, which are needed to fulfill the service availability requirements of all related flows, is calculated as

$$z_v = \sum_c z_v(c) \quad (8)$$

where $z_v(c)$ denotes the number of backup instances of NF v which are needed for related flows in class c and is simply estimated from, assuming that all flows in the class require h_c number of backup chains:

$$z_v(c) = h_c \lceil \frac{\sum_{f \in \mathcal{F}_c / v \in \overrightarrow{S}_f} \lambda_f}{\mu_v} \rceil \quad (9)$$

Note that, the estimation (7) has adopted conservative assumptions, e.g. NFs of a chain are hosted at different nodes, to get (7). In comparison with the literature approaches in [10]–[13], they may instead assume instances for the same NF chain to be hosted at the same backup node. In addition, with the estimation (8), it can be expected that the estimated numbers of backup instances are higher than the optimal numbers needed to fulfill the availability requirements of flows. However, we highlight that, the numbers from (8) are only "rough" estimates as the starting point. By setting the objective in assigning them to form backup chains to be maximizing the utilization of the backup instances (cf, Sec. V), the actually used and hence needed numbers of backup instances can be significantly reduced (cf, Sec. VI-C1).

B. Placement of the Backup Instances

After the numbers of backup instances are estimated, CoShare places them on nodes. The heuristic is presented in Algorithm 2. The placement is made by performing *bin-packing* [33] of the NF instances on the nodes, where the *network structural correlation* among nodes, the *heterogeneity* in the availability level of nodes and NF instances, and in the availability requirements of flows, and the number of backup instances for each NF, are particularly taken into consideration.

On the node side for bin-packing, nodes are categorized and prioritized. Specifically, to avoid simultaneous unavailability of the primary and backup chains, it is intuitive that the NFs of a backup chain for a flow should avoid being hosted on those nodes that are "critically" correlated with the nodes hosting the primary NF chain of the flow. To this aim, based on the structural correlation information input, $\hat{\mathcal{B}}_n$, from Algorithm 1, nodes are categorized into two sets, $Q'_n(c)$ and $Q''_n(c)$, where the former represents the set of nodes that are not structure-critically correlated with the primary nodes of flows in class c, and the latter the rest. In the placement or bin-packing, as the intuition indicates, nodes in $Q''_n(c)$ are considered only after nodes in $Q'_n(c)$ have been exhausted (Lines 5 - 8).

In addition, nodes may have different availability levels, e.g., high-end nodes having 99.9% availability or higher while low-end nodes having 99% availability or lower. In [29], it is shown that for the low availability requirement class, it is more cost-efficient to use the low-end nodes. While for the medium and high availability requirement classes, using high-end nodes is preferable as this will lead to the use of fewer backup instances and nodes. Considering these, CoShare prioritizes nodes based on their availability levels. This prioritization is translated into the sorting of nodes (Lines 3 and 4), before the bin-packing is performed.

On the instance side for bin-packing, prioritization is also performed. Specifically, NF type that has greater number of backup instances to be placed is given higher priority to be placed. The underlying intuition is, the estimate (9) implies that more flows require this NF and its backup instances to achieve their availability requirements. Hence, giving it priority will more likely accommodate a higher number of

Algorithm 2 CoShare's Placement Heuristic

Definitions:
$\mathcal{N}^c \leftarrow$ set of nodes hosting primary instances of class c flows
$Q'_n(c) \leftarrow$ priority queue of structurally uncorrelated candidate backup nodes
$Q''_n(c) \leftarrow$ priority queue of the other candidate backup nodes, i.e., $\mathcal{N}^{-Q'_n(c)}$
$Q_v(c) \leftarrow$ priority queue of all NF types v to be placed for each class c
$z_v(c) \leftarrow$ number of instances of NF type v to be placed for class c
Input: $G(\mathcal{N}, \mathcal{L})$, $\hat{\mathcal{B}}^s_n = \mathcal{N}^{-\hat{\mathcal{B}}_n}$ complement of set $\hat{\mathcal{B}}_n$ (from Algorithm 1)
Output: Placement of backup instances on nodes
Initialize:
$ActiveNode \leftarrow$ `null`

1: $Q_v(c) =$ sort $z_v(c)$ in descending order
2: **for** each class c **do**
3: $Q'_n(c) = \bigcap_{n \in \mathcal{N}^c} \hat{\mathcal{B}}^s_n$ sorted based on node availability
4: $Q''_n(c) = \mathcal{N}^{-Q'_n(c)}$ sorted based on node availability
5: **if** $Q'_n(c)$ is not empty **then**
6: $ActiveNode \leftarrow$ top of $Q'_n(c)$
7: **else**
8: $ActiveNode \leftarrow$ top of $Q''_n(c)$
9: **while** $Q_v(c)$ not empty **do**
10: $v \leftarrow$ NF type from top of $Q_v(c)$ with $z_v(c) > 0$
11: **while** $Q'_n(c)$ and $Q''_n(c)$ not empty **do**
12: **if** $ActiveNode$ has capacity **then**
13: Place v on the $ActiveNode$
14: $z_v(c) = z_v(c) - 1$
15: $v \leftarrow$ next NF type from top of $Q_v(c)$
16: **else**
17: Remove $ActiveNode$ from $Q'_n(c)$ or $Q''_n(c)$
18: **if** $Q'_n(c)$ is not empty **then**
19: $ActiveNode \leftarrow$ top of $Q'_n(c)$
20: **else**
21: $ActiveNode \leftarrow$ top of $Q''_n(c)$

flows [26]. This prioritization is reflected in sorting the NF instances based on their numbers (Line 1).

Finally, the placement is completed by bin-packing the backup instances onto the nodes, based on the prioritization introduced above. Specifically the top-prioritized node is checked for its capacity. If it has enough capacity, the NF instance with the highest priority is placed on it. If not, the next prioritized node is checked for possible placement of this instance (Lines 9-21). As a highlight, a node may have capacity to host multiple NF instances. In such a case, CoShare diversifies the types of instances placed on the node. The underlying intuition is that, the backup chain delay is minimized if all its NFs are hosted on the same node, and placing instances of different types on one node increases this chance. This is reflected by Lines 13-15 in Algorithm 2, where after placing a given NF type on a node, the next NF type from queue $Q_v(c)$ is chosen to be placed on the same node instead of another instance of the same type. This procedure is repeated until all the instances are placed or all the backup resources are utilized.

V. Assignment of NF Instances to Flows

The goal of redundancy allocation in NFV is to achieve the desired availability levels for flows. To this aim, having estimated the needed backup instances for each NF and decided where to place them in Sec. IV, CoShare assigns NF instances to flows to form backup chains for them so as to meet their availability requirements, which is the focus of this section.

A. Feasible NF Instance Set for Assignment

Note that every backup service chain of a flow must include all the NFs ordered in the same way as the primary chain. However, for every NF, it may have multiple instances. This subsection is devoted to identifying a set of such instances, called the "feasible set", which are considered for the assignment in CoShare. By *feasible set*, it is meant that with the backup chain formed by any combination of related instances in the set, the flow's availability requirement can be met.

It is obvious that any instance without enough capacity to accommodate the flow should not be included in the feasible set. In addition, as discussed in Sec. III, an NF instance whose hosting node has critical structural correlation with a node of the primary chain (C.f. Algorithm 1) should not be included in the feasible set either. Let $\tilde{\mathcal{I}}_{S^g_f}$, $g = 1, \ldots, g^f$ denote the resultant instance set of each NF S^g_f of the flow.

Since each instance and the hosting node have implicit availability levels, this information can be made use of to find the feasible set. In particular, for one backup chain, the best availability range, which can be achieved, is between (MIN, MAX) which are calculated as

$$MIN = \min_{v(g) \in \tilde{\mathcal{I}}_{S^g_f}, \forall g} 1 - (1 - A^p_f)(1 - A^b_f) \quad (10)$$

$$MAX = \max_{v(g) \in \tilde{\mathcal{I}}_{S^g_f}, \forall g} 1 - (1 - A^p_f)(1 - A^b_f) \quad (11)$$

where A^b_f is found from (1). Note that (1) has two parts: $\prod_{g=1}^{g_f} A_{v(g)}$ that is the availability resulted from the involved instances $v(g), \forall g = 1, \ldots, g^f$, and $\prod_{b=1}^{|\mathcal{N}^b(f)|} A_b$ that is the availability resulted from the involved hosting nodes $\mathcal{N}^b(f)$. When all instances $v(g), \forall g = 1, \ldots, g^f$ are hosted at different nodes, Eq. (1) can be re-written as

$$A^b_f = \prod_{g=1}^{g_f} A_{v(g)} A_{n(v(g))}$$

where $n(v(g)$ denotes the node hosting the instance $v(g)$.

Intuitively, when the required availability A^r_f is smaller than MIN, this implies that all possible combinations out of $\tilde{\mathcal{I}}_{S^g_f}$, $g = 1, \ldots, g^f$, to form a backup NF chain for the flow are able to help meet the requirement and hence the set is already a feasible set. When $MIN < A^r_f < MAX$, it means the required availability can be achieved by some (but not all) combinations of instances in $\tilde{\mathcal{I}}_{S^g_f}$, $g = 1, \ldots, g^f$. In other words, this set is not a feasible set yet, and additional effort is needed as explained below.

CoShare uses Algorithm 3 to find the feasible set. To illustrate the idea, Fig. 2 shows a simple example. In the example, the considered flow needs a service composed of two NFs, firewall and load-balancer, and its availability requirement is 0.9999, but the primary service chain only has availability of 0.99. In the network, after excluding those instances without enough capacity left or whose nodes are network-structurally correlated with the primary chain nodes, there are still five candidate instances of each NF hosted at different nodes. After

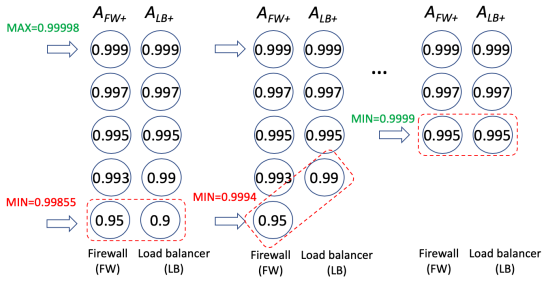


Fig. 2: Example of finding feasible instance set (Availability requirement: 0.9999; Availability of the primary chain: 0.99)

Algorithm 3 Finding feasible set of NF instances

Definitions:
$\mathcal{V} \leftarrow$ set of NF instances of type v
$A_{v+} \leftarrow A_v * A_b$, where b is the backup host node of NF $v \in \mathcal{V}$
Output: set of feasible NF instances $\{\mathcal{I}_{S_f^1}, \cdots, \mathcal{I}_{S_f^{g_f}}\}$

Sort the instances in $\mathcal{V}$ in descending order of A_{v+}
Find MIN from the instances that have $\min A_{v+}$ using Eq. (10)
Find MAX from the instances that have $\max A_{v+}$ using Eq. (11)
if $MIN \geq A_f^f$ & $|\mathcal{N}^b(f)| = g_f$ **then**
    Insert the instances to the feasible set
else
    while $MIN < A_f^f$ || $|\mathcal{N}^b(f)| < g_f$ **do**
        if $MIN \geq A_f^p$ **then**
            Insert the instances making $\min A_{v+}$ to the feasible set
        Replace the instance with the smallest availability
        Recalculate MIN
    Insert the instances to the feasible set

sorting, their availability levels, $A_{FW+} = A_{FW} \cdot A_n$ and $A_{LB+} = A_{LB} \cdot A_{n'}$, taking into account both node availability and instance availability, are shown in the figure.

Clearly, the required availability level 0.9999 is within this (MIN, MAX). An implication is that this requirement can be met with one backup chain. Another is that, some of the instances should not be included in the feasible set. To this aim, we drop the instance with the lowest availability, which is the load balancer instance with availability of 0.9, and then re-calculate MIN as shown in the middle part of Fig. 2 and compare it with the required availability. This process is repeated until MIN is equal to or higher than the required availability. All the remaining instances form the feasible set.

In the above discussion and example, the required availability level is less than MAX. If however this is not true, it means that one backup chain is not enough to fulfill the service availability requirement for the flow. In such cases, we update A_f^p with MAX in Algorithm 3 when applying (10) and (11), add the corresponding instances to the assignment and remove them from the candidate lists, and consider using an additional backup chain. This process is repeated until the required availability can be achieved, or there are not enough candidate instances left to form additional backup chains, implying the availability requirement is infeasible to achieve.

B. Feasible Backup Chains

Let $\{\mathcal{I}_{S_f^1}, \cdots, \mathcal{I}_{S_f^{g_f}}\}$ denote the feasible NF instance set, where $\mathcal{I}_{S_f^v}$, $v = 1, \ldots, g_f$, represents the set of instances of network function S_f^v in the feasible set. Then, the possible backup chains for the flow are easily obtained as:

$$(I_{S_f^1}, \cdots, I_{S_f^{g_f}}) \equiv \mathcal{R}^d(f)$$

$\forall\ I_{S_f^v} \in \mathcal{I}_{S_f^v}$, $v = 1, \ldots, g_f$, where $I_{S_f^v}$ denotes an instance of NF S_f^v. It is easily verified that, the total number of such possible backup chains is:

$$|\mathcal{I}_{S_f^1}| \cdots |\mathcal{I}_{S_f^{g_f}}|.$$

C. Assignment of NF Instances to Flows

In this subsection, we introduce the assignment strategy of CoShare. In brief, for each flow f, out of the feasible backup chains, CoShare assigns to the flow the chain that maximizes the utilization of resources so as to minimize the needed numbers of NF instances.

1) Backup capacity reservation: Since each flow f has an arrival rate λ_f, every backup NF instance assigned to the flow needs to also reserve λ_f amount of capacity to the flow. As a consequence, it is intuitive to reserve the same amount of resource for every backup chain as for the primary chain, where dedicated capacity is reserved at every instance [7], [10], [34]. We call this approach *dedicated reservation*. Here, it is worth highlighting that, even in this approach, there is sharing at the instance level, i.e., the capacity of the instance can be shared by backup chains of multiple flows as long as the capacity constraint allows, i.e. $\sum_{f \in \mathcal{F}^v} \lambda_f \leq \mu_v$, where $\mathcal{F}^v$ denotes the set of flows using this NF instance v.

However, in practice, the probability that multiple independent failures occur at the same time is low and planning the redundancy considering this rare occasion is costly in terms of resource utilization [35]. Taking this into consideration, the idea of sharing reserved backup capacity among multiple flows has long been exploited in backup allocation to improve resource utilization in different types of networks [21]–[24]. In CoShare, we propose to adopt the same idea to reduce the needed numbers of backup NF instances in redundancy allocation. We call this approach *shared reservation*.

Specifically, in CoShare, flows with disjoint primary service chains, referred to as *independent flows*, are allowed to share reserved capacity at a backup instance. Let $\mathcal{F}_v$ denote the set of independent flows whose service chains use backup instance v. Then, in CoShare with shared reservation, the backup NF instance v will only need to reserve a capacity of $\max_{f \in \mathcal{F}_v} \lambda_f$ for all these flows. In comparison, if dedicated reservation is used, for the same set of flows $\mathcal{F}_v$, the backup instance will need to reserve a capacity of $\sum_{f \in \mathcal{F}_v} \lambda_f$ for them. In Fig. 3, an example illustrating the difference in capacity allocation between shared reservation and dedicated reservation is presented.

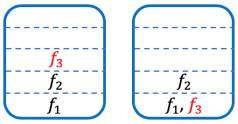


Fig. 3: Illustration of dedicated reservation (left) v.s. shared reservation (right): The instance has capacity of 5. Each flow requires capacity of 1. Flows f_1 and f_3 are independent.

2) The assignment heuristic: Note that for each flow, a set of feasible backup chains, denoted as $\mathcal{R}^d(f)$, which together with the primary service chain can meet the availability requirement of the flow has been identified in the previous subsection. Now, the challenge is how to apply the idea of shared reservation in this special setup, i.e. to choose / assign backup chain(s) to the flow to meet Objective (3). To this aim, CoShare utilizes a weight-based approach:

For each chain $r \in \mathcal{R}^d(f)$, it is given a weight $W^p(r,f)$ calculated as

$$W(r,f) = \sum_{j=1}^{g_f} w(v^r_{f,j}) \tag{12}$$

which is the summation of the weight of each instance constituting r, where $w(v^r_{f,j})$ denotes the weight of the instance of the j-th NF in r for flow f, $v^r_{f,j}$ this instance, and g_f the service chain length for flow f.

In (12), the instance weight $w(v^r_{f,j})$ is calculated using:

$$w(v^r_{f,j}) = \begin{cases} g_f, & f \perp\!\!\!\perp f_a \quad \& \quad f \perp\!\!\!\perp \mathcal{SR}_{v^r_{f,j}}(f_a), \quad f_a \in \mathcal{D}_{v^r_{f,j}} \\ \frac{\sum_{f_a \in \mathcal{D}_{v^r_{f,j}}} \lambda_{f_a}}{\mu_{v^r_{f,j}}}, & \text{otherwise.} \end{cases} \tag{13}$$

where $\mathcal{D}_v$ denotes the set of flows that are reserving backup capacity at instance v, $\mathcal{SR}_v(f)$ the set of flows that are sharing the capacity of instance v with flow f, λ_f the rate of flow f, and μ_v the capacity of instance v. In addition, in (13), $\perp\!\!\!\perp$ is used to represent independence between flows.

The key idea of (13) is as follows. If flow f is independent with flow f_a, i.e. $f \perp\!\!\!\perp f_a$, as well as all flows that are sharing capacity of the instance $v^r_{f,j}$ with flow f_a, i.e., $f \perp\!\!\!\perp \mathcal{SR}_{v^r_{f,j}}(f_a)$, then it implies that flow f can *share reserved capacity* with flow f_a at the instance $v^r_{f,j}$. In this case, the instance will be given a weight equal to the service chain length of the flow, i.e. g_f. Otherwise, the instance is given a weight that is equal to the current utilization level of the instance.

In CoShare, among all backup chain compositions $\mathcal{R}^d(f)$ of a flow f, it is assigned with the backup chain that has the maximum chain weight i.e., $\max_{r \in \mathcal{R}^d(f)} W(r,f)$. Implied by (13), this assignment strategy prioritizes assigning flows to more utilized instances – an intuitive approach to minimize the needed number of instances of each NF in Objective (3).

D. Efficiency and Scalability of CoShare

The assignment heuristic of CoShare ensures giving higher priority to instances where shared reservation is possible as proved in Theorem 1. Since shared reservation is more resource efficient than dedicated reservation, it helps increase resource efficiency in assigning backup instances to flows to meet their availability requirements, and consequently reduce the total needed number of backup instances.

Theorem 1. *Consider two backup chain choices r and r' ($\in \mathcal{R}^d(f)$). The other conditions are the same, but r has at least one instance $v^r_{f,j}$ on which the flow can share reserved capacity with other flows, while r' does not have. Then, r has a larger chain weight than r', i.e.,*

$$W(r,f) > W(r^{'},f).$$

Proof: Refer to Appendix A.

The complexity of CoShare is determined by the three involved parts, namely network structural analysis (Sec. III), placement (Sec. IV), and assignment (Sec. V). In network structural analysis, the computational complexity of finding the critical node set $\mathcal{C}(n)$ for every node $n \in \mathcal{N}$ is $O(N^2)$ from (6) and (5), so its complexity is $O(N^3)$. CoShare's placement heuristic performs a bin-packing of backup NF instances on nodes. The complexity of the algorithm is a function of the number of instances to be placed and the number of candidate backup host nodes. Specifically, sorting is performed on both the node side and the instance side, whose complexities are $O(N^2)$ are $O(z_v^2)$ respectively, where z_v is the maximum number of backup instances (8) that may be involved in the sorting. Approximating the number of NF instances by the number of nodes, the placement heuristic has a complexity $O(N^2)$. The assignment of NF instances to form backup chains for flows has a complexity of $O(FN^G)$, where G denotes the longest service chain length. This is because, for every flow, the worst case is to search through all the possible chain compositions and the total number is $|\mathcal{I}_{S_f^1}| \cdots |\mathcal{I}_{S_f^{g_f}}|$ which is upper-bounded by $O(N^G)$. Thus, the complexity of CoShare is $O(N^3 + N^2 + FN^G)$, which is approximately $O(FN^G)$, under practical assumptions $F \geq N$ and $G \geq 2$.

In brief, the complexity of CoShare can be written as a function of the number of flows that are to be assigned backup chains, the number of nodes / instances in the network, and the longest length of service chains, which is summarized as:

Theorem 2. *CoShare has a complexity of $O(FN^G)$, where F is the number of flows, G the longest length of service chains and N the number of nodes in the network.*

VI. Results and Discussion

This section presents results showing the performance of the proposed redundancy allocation approach, CoShare. Recall that, a novel idea of CoShare is to exploit network structural correlation information in the design. Sec. VI-A is hence devoted to showing the impact of such correlation. The remaining Sec. VI-B – Sec. VI-D focus on introducing the performance of CoShare where comparisons are also included.

Specifically, a number of experiments are conducted on two realistic ISP network topologies. In the study, if not otherwise

specified, it is assumed that each node hosts 8 CPU cores and has 16 GB memory, of which half of the capacity i.e., 4 CPU cores and 8 GB of memory, is used by the primary NFs and the rest by backup NFs. The primary NFs' placement as well as the assignment of primary service chains to flows is conducted by using ClusPR [26]. Every NF instance requires one CPU core and 2GB of memory, having a total NF processing capacity (μ_v) of 10Mpps. Five types of NFs are considered (e.g., Firewall, DPI, NAT, IDS, and Proxy). The availability requirements of flows are set according to three levels, namely 99.9% (3'9s), 99.99% (4'9s) and 99.999% (5'9s). The NF processing capacity required by each flow f, i.e., λ_f, is set to 0.5 Mpps. The length of the service chain for each flow is assumed to vary in the range of 2 to 4. NFs and the service types in the chain of each flow are selected randomly out of the five NFs considered. The availability of the hosting nodes is assumed to be uniformly distributed between $0.99 - 0.999$, NF instances have an availability between $0.999 - 0.9999$, and the threshold algorithmic parameter t_{DI} is set to 0.5 if not otherwise specified. If a flow's availability requirement cannot be satisfied by the adopted redundancy allocation approach, it is rejected. All algorithms and simulations are run on a Dell workstation with a single Intel® Xeon® octa-core processor with 2.4 GHz base frequency and 64 GB of RAM.

A. Impact of Network Structural Correlation

In this subsection, a simple experiment is carried out to showcase the effect of not considering the network structural correlation among nodes in the backup instance placement decision making. The GEANT network topology [36] is used for this experimental study. The GEANT network is a pan-European network connecting research and education institutions consisting of 44 nodes and 136 links. (For visibility, a version with 44 nodes and 68 links is illustrated in Fig. 1.) Only node availability impact is considered. It is assumed that the availability of each node is 0.999 (3'9s). For each flow, the service function chain contains two NFs, and the required availability of the flow is 99.999%. The baseline algorithm from [7] is used to decide the number of backup instances needed. According to the baseline algorithm [7], theoretically, one backup for each of the NFs is enough to meet the 99.999% (5'9s) availability requirement.

Two backup instance placement strategies are considered. One does not consider network structural correlation, where the primary and backup NF host nodes of a chain are randomly chosen in the network. Another takes the structural correlation into consideration, where the backup host nodes are chosen from those without critical network structural correlation with the primary host nodes.

The availabilities of 100 flows are measured by conducting ten million simulation runs. In each simulation run, the state of each node, i.e., failed (0) or up (1), is randomly generated from Bernoulli distribution using the node's availability. The unavailability CDF of the 100 flows is shown in Fig. 4. As can be read from the figure, the strategy taking network structural correlation into consideration performs significantly

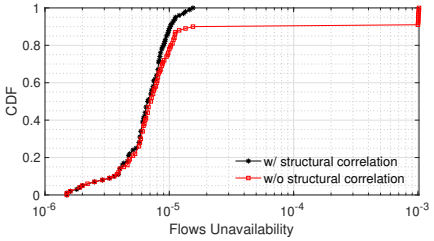


Fig. 4: Impact of Network Structural Correlation

better. Specifically, when network structural correlation is not considered, about 10% of the flows only have availability of 3'9s or even lower, in contrast to taking it into consideration where all flows have at least 4'9s availability. In addition, while only 80% of flows can reach the required 5'9s in the former, this percentage increases to 90% in the latter.

There are two implications of this experimental study. One is that the number calculated by the baseline algorithm [7] may not be enough to meet the availability requirements of all flows when applied to a real network. Another is that, the inherent network structural correlation among nodes can have significant impact on the availabilities of the services, and hence is a crucial factor that should be taken into consideration for redundancy allocation.

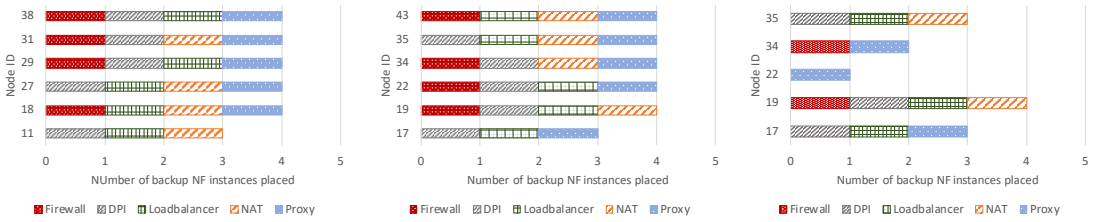
B. Comparison with Optimal Model

In order to assess the optimality level of CoShare, its performance is compared with an integer linear program (ILP) approach called *AllAny*, which has been proposed to solve the optimization problem (3) in [29]. The ILP model assumes dedicated reserved capacity, i.e. dedicated reservation, at the backup instances. The GEANT network is also used for this analysis. Backup chains are allocated for 200 flows whose primary chains are served by using 23 NF instances, where every flow has a chain length of 2, and its availability requirement is randomly set to be 3'9s, 4'9s, or 5'9s.

For the comparison, we adopt the concept of *resource overbuild*, which is a key figure-of-merit in assessing redundancy capacity efficiency [37], i.e. the extra capacity needed to meet the service availability objective as a percentage of the capacity for the service under no redundancy. In this paper, it is defined to be the ratio of the total number of backup NF instances actually used to meet the availability requirement to the total number of primary NF instances.

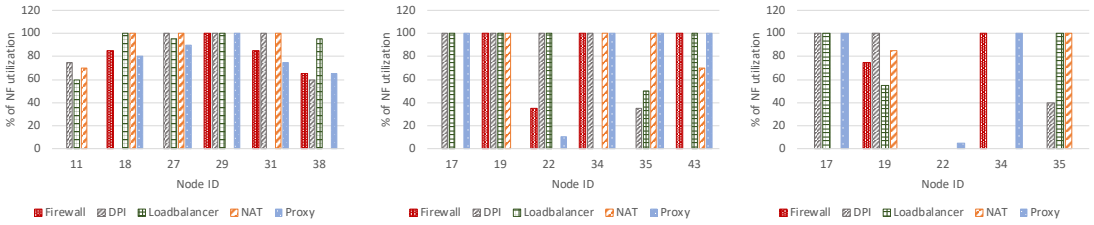
Fig. 5 shows the placement of the backup instances, i.e., the number and type of backup NF instances placed together with their host nodes, obtained using the *AllAny* model and CoShare. For CoShare, both shared reservation and dedicated reservation are considered.

The number of backup instances for each NF which are created by CoShare with dedicated reservation is the same as that created by the *AllAny* model. In total, 23 backup instances are created by each of the two approaches and six backup nodes are used to host the instances, i.e., 100%



(a) *AllAny* model (b) CoShare: dedicated reservation (c) CoShare: shared reservation

Fig. 5: Placement of backup NF instances by the *AllAny* model [29], and CoShare using dedicated and shared reservations



(a) *AllAny* model (b) CoShare: dedicated reservation (c) CoShare: shared reservation

Fig. 6: Per NF utilization of the placed backup NF instances

resource overbuild. The per NF utilization level, which is the percentage of the NF capacity that is reserved by the flows, is shown in Fig. 6. Since CoShare intends to maximize the utilization of NF instances, most of the backup instances are 100% utilized in contrast to the NFs of the *AllAny* model. Moreover, with CoShare using shared reservation, only 13 backup NF instances are created and five backup host nodes are used, which results in only about 56% resource overbuild.

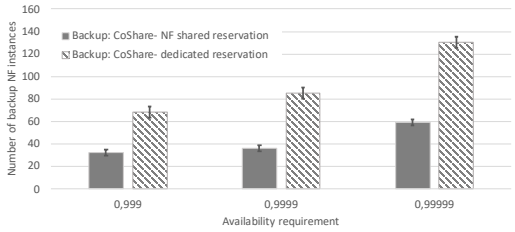
It is worth highlighting that in this example, the *AllAny* is solved by using the LP solver, CPLEX, and hence the results are optimal. However, even for this simple example, it took more than a dozen minutes to find the optimal solution by solving the model, due to the NP complexity nature of the optimization project (3). In contrast, CoShare obtained the results in less than one second. These showcase that CoShare is able to get near optimal results in much less time, and when applying shared reservation, CoShare can achieve better resource efficiency.

C. Effect of Shared Reservation

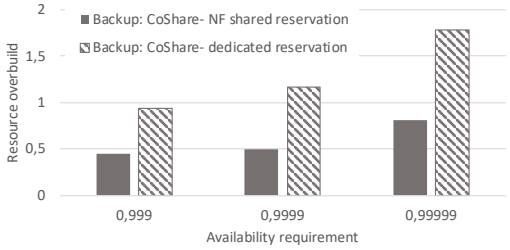
We now consider a larger network: The Rocketfuel topology AS 1221 with 100 nodes and 294 links [38] is adopted. With the increased numbers of nodes and links, the computation of the optimal results has increased too much (due to NP-complexity) to be handled by the adopted workstation. For this reason, CoShare with dedicated reservation, which has similar performance as the optimal model shown in the above example, will be used in the comparison.

1) Resource overbuild: Fig. 7a shows the number of backup NF instances that are needed to satisfy different levels of availability by using CoShare with shared and dedicated reservation, where all flows require the same availability level. The results shown are average values with 95% confidence intervals over ten simulation runs for each availability level. In each of the availability levels, the NF instances are created for 700 flows having a service chain containing two NFs. In addition, the resource overbuild for both backup allocation strategies is illustrated in Fig. 7b. As can be observed, the higher the availability requirement level the more the number of backup NF instances required.

Recall that, in the placement phase, CoShare first "roughly" estimates the number of backup chains for each availability requirement class and accordingly the number of backup NF instances, i.e. h_c and $z(c)$ in Sec. IV-A, which are possibly needed. For the three availability levels [0.999, 0.9999, and 0.99999], the corresponding rough estimates are [1, 2, and 3] for h_c and [70, 140, and 210] for $z(c)$ respectively. As discussed in Sec. IV-A, the actually used and hence needed numbers of backup instances under CoShare can be expected to be much lower. This is confirmed by Fig. 7a that shows the real total numbers of used backup instances under CoShare with dedicated reservation and shared reservation. Specifically under CoShare with dedicated reservation, they are [68, 85, 130] for the three availability levels, which are reductions of 2.8%, 39%, and 38% from the rough estimates respectively. Under CoShare with shared reservation, there are further reductions of 44%, 64%, and 69%.

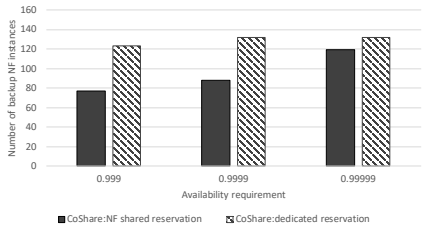


(a) Number of backup NF instances

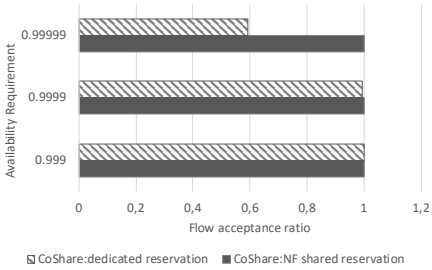


(b) Resource overbuild

Fig. 7: Comparison: Resource overbuild



(a) Number of backup NF instances



(b) Flow acceptance ratio

Fig. 8: Comparison: Flow acceptance ratio

Fig. 7b further compares dedicated reservation and shared reservation using resource overbuild. As shown by the figure, to fulfill 0.99999 availability, the former requires 178% resource overbuild, in contrast to the much reduced 93% by the latter. Similar reduction is found also for the other two availability levels. Overall, shared reservation enables more efficient utilization of resources with significant decrease in the required number of backup NF instances.

2) Flow acceptance ratio: In the above experiments, no flow is rejected, i.e. all flows' availability requirements can be met with CoShare, with or without shared reservation. In the following experiment, we consider 650 flows each with a service chain consisting of four NFs. Similar to Fig. 7a, Fig. 8a compares the number of backup NF instances under dedicated and shared reservation. It also shows that less instances are needed with shared reservation.

Additionally, Fig. 8b compares the flow acceptance ratio. As shown by the figure, all flows can be admitted with both shared and dedicated reservation when the availability requirements are under the two lower levels. However, when 5'9s availability level is required, only about 60% of the flows can be admitted with dedicated reservation, in contrast to 100% with shared reservation. This is because *CoShare with dedicated reservation* requires a higher number of backup NF instances than what can be provided by the network. If that number would have been possible, Fig. 8a would have shown an even higher reduction by using shared reservation. This again implies that NF shared reservation enables more resource efficiency which in turn maximizes the number of flows that can be admitted to the network.

D. Effect of the Threshold

CoShare has one algorithmic parameter, the threshold $t_{DI} \in (0, 1)$, which is used in (6) to help identify the set of critical nodes due to network structural correlation. As can be expected from (6), a higher t_{DI} leads to a smaller set. To have a better overview about the effect of the threshold, experiments have also been conducted. Figure 9 shows the number of backup NF instances instantiated for fulfilling the availability requirement of 0.99999 under different threshold values for the case of Rocketfuel topology with 700 flows. For the threshold value between 0.2 and 0.9, the same number is found. In fact, with a closer look, it has been found that the same set of structurally correlated nodes are resulted from (6) with a threshold value in this range. However, when $t_{DI} \in (0, 1)$ is too small, e.g. 0.1, it is observed that all the network nodes are included in the set. These indicate that the performance of CoShare is generally robust to the threshold except when a too small value is given to it.

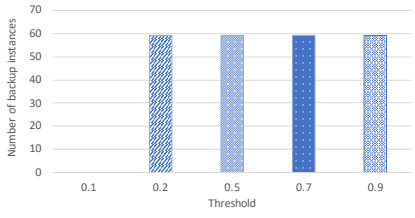
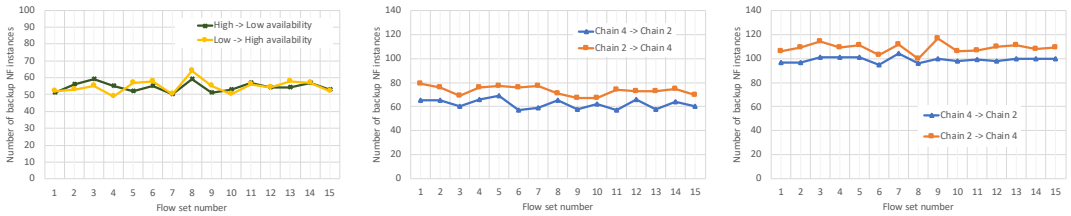


Fig. 9: Effect of t_{DI}



(a) Chain length=2 (b) Low Availability class: 0.999 (c) High Availability class: 0.99999

Fig. 10: Effect of prioritization based on availability requirements (a) and service chain length (b) and (c).

E. Effect of Assignment Order

Note that when CoShare is applied, flows are checked one-by-one in the step of assigning instances to form back chains for them (C.f. Sec. V). When there are multiple flows, these flows may be applied with CoShare at arbitrary order. In the rest, the effects of two specific yet intuitive orders are investigated.

In this investigation, the Rocketfuel topology is adopted. 15 sets of randomly generated 700 flows are considered. Each set is used for one simulation run. The availability requirement of each flow is randomly chosen among 3'9s, 4'9s and 5'9s. CoShare with shared reservation is focused.

1) Based on availability requirement: First, we consider the effect of ordering the assignment based on the required availability levels. In this study, flows have the same service chain length of two. The flows are assigned with needed backup instance in the order of their availability requirements, from high (5'9s) to low (3'9s) or from low to high.

Fig. 10a compares the total needed number of backup NF instances in fulfilling the availability requirements of flows in the 5'9s class under each simulation run using a different set of flows. As can be seen from the figure, there is no clear evidence about which order is better. Similar observation has also been found for the other two classes. This implies that prioritizing assignment based on availability requirements has minimal effect on the performance of CoShare.

2) Based on service chain length: Next, the impact of ordering based on the flow's service chain length is assessed: long to short or short to long. The experiment setup is similar to above, but the chain length of a flow is randomly chosen between 2 and 4.

Figs. 10b and 10c compare the number of needed backup NF instances for the 3'9s and 5'9s classes respectively. As can be observed from these figures, prioritizing flows that have longer service chain length results in fewer backup NF instances needed. An underlying reason is that, when a longer chain flow is prioritized, due to the involvement of more instances, the chance of finding flows that can share capacity on some of these backup instances is higher. Nevertheless, the difference is minimal, about 10% or less.

F. Discussion

The evaluation highlights the significant improvement that CoShare brings in terms of resource utilization efficiency and flow acceptance ratio. However, generalizing the results requires performance evaluation in more diversified NFV setups. We briefly present some aspects that could be further explored as future work.

The placement decisions are subject to the underlying network topology structure and its intrinsic graph characteristics (i.e., node degree, betweeness etc.). Aggregated results on a variety of network topologies, e.g., those present in SNDlib [39], may provide more detailed insights about the effects of structural correlation on the placement and performance. Moreover, it is common that NFV-enabled network operators provide services through well-defined service chains, typically included in their NFV service catalogs, rather than through a random composition of NF instances. An alternative evaluation would be to consider a set of specific service chains and characterize their availability demands and probability of occurrence, i.e., probability of being requested. In addition, we made some simplifying assumptions compared to a more truthful representation of network nodes' capacities. In realistic networks, some of the nodes could be central offices with rather limited capabilities, whereas some other nodes could be data centers with significantly more computing resources.

VII. Related Work

Guaranteeing service availability in NFV-enabled networks represents an important challenge that needs to be addressed to fully exploit the benefits of NFV [3], [4]. To this aim, there has been a continuous effort in the recent literature to investigate and propose resource efficient and scalable algorithms for resource / redundancy allocation in NFV.

Fan *et al.* [10] presented an algorithm to minimize the employed physical resources by protecting the most unreliable NFs. On similar lines, they extended the work by proposing methods for allocating backup resources in order to maximize the number of accommodated service requests while meeting heterogeneous availability demands [7]. In [32], the authors studied the suitability of various data center architectures for resilient NFV service deployments. Ding *et al.* [40] improved the design in [10] by proposing a method to select the

most appropriate NFs to protect by exploiting a cost-aware critical importance measure rather than the least available NFs. However, these contributions are based on assumptions which may significantly impact their applicability in more general setups [28], [32]. Such assumptions include homogeneous backup nodes, considering only the failure of NFs while ignoring physical nodes' failure and vice versa [10], [40], or assuming NF instances fail independently irrespective of their placement [7].

In [14], three ILP models are proposed for VNF placement and service chaining. However, their aim is to protect the service chains against different types of failure without taking into account the specific availability requirement of each service chain. In addition, the evaluation shows that providing protection against the considered failure scenarios comes with at least twice the amount of resources in terms of the number of nodes being deployed into the network [14].

As redundancy can be costly, it is desirable to share redundancy at maximum possible in NFV based networks to enable more efficient resource utilization as having been done in traditional networks, e.g., [41], [42]. In [13], a multi-tenancy based approach, which allows a backup NF instance to be utilized by multiple flows, is proposed, and it is also demonstrated that the multi-tenancy based approach outperforms single-tenancy based approaches. However, in the approach proposed in [13], a backup chain is constrained to only using NFs hosted on one node, similar to [34]. In [10], aiming at minimizing the physical resource consumption, a joint protection scheme is proposed where the sum of resources between two adjacent NFs are allocated for protection. In [11], shared path protection is used to allocate backup paths that protect against single link failures. In [12], adjacent NFs share the resources of a host machine. In all these approaches, when a backup NF instance is assigned to a flow, dedicated backup capacity for the flow is reserved, same as in ChoShare with *dedicated reservation*.

CoShare is different from these literature works in several aspects. First, CoShare takes into account the heterogeneity present in terms of resources at network nodes and NF instances and their availability, in contrast to [28], [32]. Second, both node failure and NF failure as well as the impact of a node's failure on its hosted NFs are considered, different from [10], [40] and [7]. Third, CoShare aims to meet flow-level specific availability requirement, different from [14].

Forth, in terms of shared reservation, to enable resource efficiency, CoShare allows a backup service chain being constructed by NF instances placed at different nodes, as opposed to the approaches studied in [13], [34]. In addition, CoShare also differs from [10], [12]. Specifically, the shared reservation mechanism employed in CoShare provides protection to multiple service chains that request the same NFs, rather than protecting adjacent NFs of the same service chain in [10], [12]. It is worth highlighting that in all the literature approaches [10], [12], [13], [34], even though a backup NF instance may be shared among multiple flows or tenants, dedicated capacity is reserved for each flow / tenant, the same as in *dedicated reservation* discussed in Sec. V-C. In other words, as in CoShare's dedicated reservation, resource sharing in all these approaches is at the instance level. However, CoShare's shared reservation *additionally* allows the sharing to be made at the flow or service chain level, leading to improved efficiency in making use of resources. Moreover, none of the previous works takes into consideration topological dependencies among network nodes. Since such dependencies are inherent in the network structure / topology, disregarding them could lead to the failure of both the primary and the backup chains at the same time, and consequently affects the actually delivered availabilities of flows lower than expected as shown by the example in Sec. VI-A. To this end, CoShare not only makes another novel contribution but also sheds a new insight for redundancy allocation in NFV.

VIII. Conclusion

In this paper, a novel scheme, called CoShare, is proposed for redundancy allocation in NFV. An original and crucial idea of CoShare is to explicitly take into account the inherent network structure-caused correlation / dependence among nodes in both redundancy placement and assignment. As a result, CoShare is able to minimize the impact of correlated failures due to network structural dependence on service availability. In addition, CoShare allows *shared reservation* among flows, i.e. let them share the same reserved backup capacity at an instance, to improve resource efficiency without compromising their availability. This forms another contribution of CoShare. Moreover, CoShare stands out with supporting diverse flow availability requirements under heterogeneous nodes and instances in terms of both resources and availability. The experimental results demonstrate that a redundancy approach without considering network structural dependence in its design can unfortunately fail to meet its promised availability. In addition, when backup capacity is dedicatedly reserved for each flow at a shared NF instance, the performance of CoShare (with dedicated reservation) is close to that of the optimal solution, but CoShare is scalable. Furthermore, with shared reservation, CoShare can reduce the resource overbuild significantly, e.g. about half or more in the Rocketfuel topology experiment. These results indicate that CoShare is appealing for redundancy allocation in NFV. They also imply the criticality and potential of taking into account network structural dependence in addressing the NFV redundancy allocation problem.

Appendix A

Proof of Theorem 1: Suppose there is one NF instance on which flow f can share reserved capacity in the backup chain r. Then, the chain weight will be

$$W^p(r,f) = g_f + \sum_{j=1}^{g_f-1} \frac{\sum_{f_a \in \mathcal{A}_{v^r_{fj}}} \lambda_{f_a}}{\mu_{v^r_{fj}}}. \tag{14}$$

For the chain r', it has no NF instance on which the flow can share reserved capacity, i.e., flow f is not independent with

all the flows in $\mathcal{A}_{v^{r'}_{fj}}$, $\forall j \in \{1 \dots g_f\}$. Then, the chain r' will have a weight

$$W^p(r^{'}, f) = \sum_{j=1}^{g_f} \frac{\sum_{f_a \in \mathcal{A}_{v^{r'}_{fj}}} \lambda_{f_a}}{\mu_{v^{r'}_{fj}}}. \tag{15}$$

Note that, the NF instances on all the backup chains in $\mathcal{R}^d(f)$ satisfy the NF capacity constraint. Thus,

$$0 \leq \frac{\sum_{f_a \in \mathcal{A}_{v^{r'}_{fj}}} \lambda_{f_a}}{\mu_{v^{r'}_{fj}}} < 1 \text{ and, } 0 \leq \frac{\sum_{f_a \in \mathcal{A}_{v^{r}_{fj}}} \lambda_{f_a}}{\mu_{v^{r}_{fj}}} < 1. \tag{16}$$

Since the service chain length $g_f \geq 1$ and the other conditions are the same, we have $W^p(r, f) > W^p(r^{'}, f)$. When there are more instances on r where the flow can share reserved backup capacity, following the same derivation as above, it can be easily verified that the weight of r will then be even higher. This concludes the proof. ■

REFERENCES

[1] SDN and OpenFlow World Congress. Network Function Virtualization, update white paper. Accessed 2019 Apr. [Online]. Available: https://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[2] B. Blanco *et al.*, "Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN," *Computer Standards & Interfaces*, vol. 54, pp. 216–228, 2017.

[3] N. ISG, "Network functions virtualisation (NFV); reliability; report on models and features for end-to-end reliability," *ETSI GS NFV-REL*, vol. 1, p. v1, 2016.

[4] R. Mijumbi *et al.*, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[5] B. Han *et al.*, "On the resiliency of virtual network functions," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 152–157, 2017.

[6] N.-T. Dinh and Y. Kim, "An efficient availability guaranteed deployment scheme for iot service chains over fog-core cloud networks," *Sensors*, vol. 18, no. 11, p. 3970, 2018.

[7] J. Fan, M. Jiang, and C. Qiao, "Carrier-grade availability-aware mapping of service function chains with on-site backups," in *25th International Symposium on Quality of Service (IWQoS)*, 2017.

[8] VMware vSphere Availability. Accessed 2020 Jun. [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/6.7/vsphere-esxi-vcenter-server-671-availability-guide.pdf

[9] S. G. Kulkarni *et al.*, "Reinforce: Achieving efficient failure resiliency for network function virtualization based services," in *Proceedings of ACM CoNext*, 2018, pp. 41–53.

[10] J. Fan *et al.*, "GREP: Guaranteeing reliability with enhanced protection in NFV," in *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, 2015, pp. 13–18.

[11] A. Tomassilli *et al.*, "Resource requirements for reliable service function chaining," in *IEEE ICC*, 2018.

[12] L. Qu, M. Khabbaz, and C. Assi, "Reliability-aware service chaining in carrier-grade softwarized networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 558–573, 2018.

[13] D. Li *et al.*, "Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1651–1664, 2019.

[14] A. Hmaity *et al.*, "Virtual network function placement for resilient service chain provisioning," in *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016, pp. 245–252.

[15] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *IEEE NFV-SDN*, 2016.

[16] M. Lalou, M. A. Tahraoui, and H. Kheddouci, "The critical node detection problem in networks: A survey," *Computer Science Review*, vol. 28, pp. 92–117, 2018.

[17] Y. T. Woldeyohannes and Y. Jiang, "Measures for network structural dependency analysis," *IEEE Communications Letters*, vol. 22, no. 10, pp. 2052 – 2055, 2018.

[18] M. D. Schirmer *et al.*, "Network structural dependency in the human connectome across the life-span," *Network Neuroscience*, 2019.

[19] D. Y. Kenett *et al.*, "Dependency network and node influence: Application to the study of financial markets," *International Journal of Bifurcation and Chaos*, vol. 22, no. 07, p. 1250181, 2012.

[20] J. Doucette, M. Clouqueur, and W. D. Grover, "On the availability and capacity requirements of shared backup path-protected mesh networks," *Optical Networks Magazine*, vol. 4, no. 6, pp. 29–44, 2003.

[21] G. Li *et al.*, "Efficient distributed path selection for shared restoration connections," in *IEEE INFOCOM*, 2002.

[22] C. Ou *et al.*, "New and improved approaches for shared-path protection in wdm mesh networks," *Journal of Lightwave Technology*, vol. 22, no. 5, pp. 1223–1232, 2004.

[23] M. Tornatore *et al.*, "PHOTO: an efficient shared-path-protection strategy based on connection-holding-time awareness," *Journal of Lightwave Technology*, vol. 23, no. 10, p. 3138, 2005.

[24] D. Xu *et al.*, "On the complexity of and algorithms for finding the shortest path with a disjoint counterpart," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 147–158, 2006.

[25] W. Zhang *et al.*, "OpenNetVM: A platform for high performance network service chains," in *Proc. the 2016 workshop on Hot topics in Middleboxes and Network Function Virtualization*. ACM, 2016.

[26] Y. T. Woldeyohannes, A. Mohammadkhan, K. Ramakrishnan, and Y. Jiang, "ClusPR: Balancing multiple objectives at scale for NFV resource allocation," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1307–1321, 2018.

[27] I. N. ETSI, "ETSI GS NFV-REL 001 V1. 1.1: Network Functions Virtualisation (NFV); Resiliency Requirements," 2015.

[28] J. Zhang *et al.*, "Raba: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE INFOCOM*, 2019, pp. 1918–1926.

[29] Y. T. Woldeyohannes, B. Tola, and Y. Jiang, "Towards carrier-grade service provisioning in NFV," in *15th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2019.

[30] L. Qu *et al.*, "Reliability-aware service provisioning in nfv-enabled enterprise datacenter networks," in *International Conference on Network and Service Management (CNSM)*, 2016, pp. 153–159.

[31] K. Stephenson and M. Zelen, "Rethinking centrality: Methods and examples," *Social networks*, vol. 11, no. 1, pp. 1–37, 1989.

[32] S. Herker *et al.*, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *IEEE Globecom Workshops*, 2015.

[33] S. Martello and P. Toth, "Bin-packing problem," in *Knapsack problems: Algorithms and computer implementations*. Wiley, 1990, pp. 221–245.

[34] J. Fan *et al.*, "A framework for provisioning availability of NFV in data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2246–2259, 2018.

[35] Y. Bejerano *et al.*, "Algorithms for computing qos paths with restoration," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 648–661, 2005.

[36] *GEANT – the pan-european research and education network,*, 2019 (accessed December 1, 2019). [Online]. Available: http://www.geant.net.

[37] C. Ou *et al.*, "New and improved approaches for shared-path protection in wdm mesh networks," *Journal of Lightwave Technology*, vol. 22, no. 5, pp. 1223–1232, 2004.

[38] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[39] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0—survivable network design library," *Networks: An International Journal*, vol. 55, no. 3, pp. 276–286, 2010.

[40] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme," in *IEEE ICC*, 2017.

[41] S. Ramamurthy and B. Mukherjee, "Survivable wdm mesh networks. part i-protection," in *IEEE INFOCOM*, 1999.

[42] G. Li *et al.*, "Efficient distributed restoration path selection for shared mesh restoration," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 761–771, 2003.

Part III

Secondary Papers

Failure Process Characteristics of Cloud-enabled Services

Besmir Tola, Yuming Jiang, and Bjarne E. Helvik
NTNU-Norwegian University of Science and Technology, Norway
Email:{besmir.tola, yuming.jiang, bjarne.e.helvik}@ntnu.no

Abstract – The design of cloud computing technologies need to guarantee high levels of availability and for this reason there is a large interest in new fault tolerant techniques that are able to keep the resilience of the systems at the desired level. The modeling of these techniques require input information about the operational state of the systems that have a stochastic nature. The aim of this paper is to provide insights into the stochastic behavior of cloud services. By exploiting the willingness of service providers to publicly expose failure incident information on the web, we collected and analyzed dependability features of a large number of incident reports counting more than 10,600 incidents related to 106 services. Through the analysis of failure data information we provide some useful insights about the Poisson nature of cloud service's failure processes by fitting well known models and assessing their suitability.

Published in – *Proceeding of the 9th Workshop on Resilient Networks Design and Modeling, (RNDM), Alghero, Italy, 2017, pp. 1-7.*

On Monolithic and Microservice Deployment of Network Functions

Sachin Sharma[1], Navdeep Uniyal[2], Besmir Tola[3], and Yuming Jiang[3]
National College of Ireland[1], University of Bristol, U.K.[2], and NTNU, Norway[3]
Email: sachin.sharma@ncirl.ie[1], navdeep.uniyal@bristol.ac.uk[2],
{besmir.tola, yuming.jiang}@ntnu.no[3]

Abstract – Network Function Virtualization (NFV) has recently attracted telecom operators to migrate network functionalities from expensive bespoke hardware systems to virtualized IT infrastructures where they are deployed as software components. Scalability, up-gradation, fault tolerance and simplified testing are important challenges in the field of NFV. In order to over- come these challenges, there is significant interest from research communities to scale or decompose network functions using the monolithic and microservice approach. In this paper, we compare the performance of both approaches using an analytic model and implementing test-bed experiments. In addition, we calculate the number of instances of monoliths or microservices in which a network function could be scaled or decomposed in order to get the maximum or required performance. Single and multiple CPU core scenarios are considered. Experimentation is performed by using an open source network function, SNORT and running monoliths and microservices of SNORT as Docker containers on bare metal machines. The experimental results compare the performance of monolith and microservice approaches and are used to estimate the validity of the analytic model. The results also show the effectiveness of our approach in finding the number of instances (monoliths or microservices) required to maximize performance.

Published in – *Proceedings of the 2019 IEEE Conference on Network Softwarization (NetSoft), Paris, France, pp. 387-395.*

Keeping Connected When the Mobile Social Network Goes Offline

Øystein Sigholt*, Besmir Tola†, Yuming Jiang†
NTNU-Norwegian University of Science and Technology, Norway
Email: oysteils@stud.ntnu.no*, {besmir.tola, yuming.jiang}@ntnu.no†

Abstract – WiFi Direct is an embedded technology in a vast majority of smartphone devices running the Android operating system. As a result, it represents a promising technology that can be exploited in re-establishing connectivity among user devices in case of cellular network outages. A technique that smart devices can use to restore connectivity in situations where they are unable to connect to a cellular tower or access point, but close enough to support device-to-device communication is presented. The proposed technique envisions a combination of security layers that ensure authentication, confidentiality, and integrity of communications among end users. Each device is issued a certificate by a central authentication entity at sign up and when it is unable to connect to the server component, it will attempt to form a group with nearby devices in the same situation over WiFi Direct. Once a WiFi Direct group has been formed, the group owner will temporarily assume the role of the server, and each group member and the group owner will verify each others identity and connect using mutual Transport Layer Security (mTLS), facilitating secure communication. The approach is validated through the implementation of a mobile social application involving several mobile devices, and overheads due to the additional security features are investigated.

Published in *– International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 2019, pp. 59-64.*