Universidade de Aveiro
2021

Joint Doctoral Program in Telecommunications:
University of Porto, Aveiro and Minho, Portugal.

Universidade do Minho
Escola de Engenharia

ASAD UR REHMAN

Fiabilidade Em Sistemas Futuros de Comunicação Definidos Por Software

Reliability On Future Software-Defined Communication Systems

Joint Doctoral Program in Telecommunications:
University of Porto, Aveiro and Minho, Portugal

DOCTORAL PROGRAMME
IN TELECOMMUNICATIONS

Research Support and Funding

I would like to acknowledge the following entities for supporting and funding this work.

U.PORTO Faculty of Engineering, the University of Porto (FEUP).

universidade de aveiro Department of Telecommunications, Electronics and Informatics, the University of Aveiro (DETI).

Universidade do Minho School of Engineering, the University of Minho (SEUM).

MAP tele DOCTORAL PROGRAMME IN TELECOMMUNICATIONS MAP-Tele (a Joint Doctoral Program of the University of Porto, Aveiro, and Minho, Portugal.

it 25 YEARS instituto de telecomunicações Research Institute: Instituto de Telecomunicações Polo de Aveiro, Portugal for hosting this work.

telenor group Telecommunications Company: Telenor Research ASA, Fornebu, Norway for hosting this work.

PD F FCT PhD PROGRAMMES Fundação para a Ciência e a Tecnologia (FCT), Portugal for supporting The MAP-Tele Ph.D. program in telecommunications under Ph.D. Grant Ref: PD/BD/113822/2015.

Universidade de Aveiro
2021

ASAD UR REHMAN

Fiabilidade Em Sistemas Futuros de Comunicação Definidos Por Software

Reliability On Future Software-Defined Communication Systems

Universidade de Aveiro
2021

ASAD UR
REHMAN

Fiabilidade Em Sistemas Futuros de Comunicação Definidos Por Software

Reliability On Future Software-Defined Communication Systems

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia de Telecomunicações O MAP-Tele, O MAP-Tele Programa de Doutoramento em Telecomunicações é um empreendimento conjunto da Universidade do Minho, da Universidade de Aveiro e da Universidade do Porto (MAP), realizada sob a orientação científica do Doutor Rui. L. Aguiar, Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro da Universidade de Aveiro, e do Doutor João Paulo Barraca, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

This work is funded by Fundação para a Ciência e a Tecnologia/Ministério da Educação e Ciência (FCT/MEC) through national funds and when applicable co-funded by FEDER-PT2020 partnership agreement under the Project Ref: UID/EEA/50008/2019 (“Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio“).

This work is also funded by Foundation for Science and Technology/Ministry of Education and Science (FCT/MEC) under Ph.D. Grant Ref: PD/BD/113822/2015 (“Apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio“).

*This work is dedicated to the loving memory of my father, **Muhammad Yaqub Khan**, whose devotion to and passion for scientific knowledge stimulated my interest in higher education. His tireless efforts and support enabled me to excel in life effortlessly. May Allah bless his soul with eternal peace.*

o júri / the jury

presidente / president	**Doutor Carlos Fernandes da Silva** *Professor Catedrático da Universidade de Aveiro*
vogais / examiners committee	**Doutor Joaquim Arnaldo Carvalho Martins** *Professor Catedrático da Universidade de Aveiro*
	Doutor Rui Luís Andrade Aguiar *Professor Catedrático da Universidade de Aveiro (Orientador/Advisor)*
	Doutor Alexandre Júlio Teixeira dos Santos *Professor Associado com Agregação da Universidade do Minho*
	Doutor Paulo Alexandre Ferreira Simões *Professor Associado da Universidade de Coimbra*
	Doutor Ivan Vidal Fernandez *Professor Visitante, Universidade Carlos III de Madrid*

agradecimentos / acknowledgements

First and foremost, I would like to thank my Ph.D. thesis advisor, Prof. Dr. Rui. L. Aguiar, for providing me with insightful guidance, mentoring, technical support, and invaluable advice throughout my Ph.D. journey. I would like to also thank my Ph.D. thesis co-advisor, Prof. Dr. João Paulo Barraca, for reviewing my documents and assisting me in writing project proposals. Finally, I would like to thank Dr. Daniel Corujo, João Filipe Barraca, and Dr. Flávio Silva Meneses for their collaborative research work. It was a pleasure working with them.
I would like to express my appreciation for the members of my Ph.D. jury, Dr. Carlos Silva, Dr. Joaquim Martins, Dr. Rui Aguiar, Dr. Alaxandre Santos, Dr. Paulo Simões, and Dr. Ivan Fernandez, for their constructive comments on my work.
I would like to acknowledge the MAP-Tele doctoral program in Telecommunications (a joint doctoral program of the Universidade do Porto, the Universidade de Aveiro, and the Universidade do Minho, Portugal, all three universities with a strong tradition in the area of telecommunications engineering).
Let me take this opportunity to thank the scientific committee of the MAP-Tele program, especially Prof. Dr. Aníbal Ferreira and Prof. Dr. Henrique Salgado, Director MAP-Tele 9th Edition for their support, seamless organization, and hosting of the MAP-Tele program at the Faculty of Engineering at the University of Porto. Furthermore, I would like to thanks my Ph.D. classmates, for the great moments I shared with them.
I would like to express my sincerest gratitude to the Foundation of Science and Technology Portugal for supporting this work, as well as the research Institute of Telecommunications (Instituto de Telecomunicações) Polo de Aveiro for hosting this work.
I would like to thank all the past and current members of the Telecommunications and Networking Laboratory led by Prof. Dr. Rui. L. Aguiar at the Instituto de Telecomunicações Polo de Aveiro. I am honored to be part of such a fantastic research group, which provided me with excellent research support and a friendly working/networking environment.
I would like to express my greatest gratitude to the Foundation of Science and Technology Portugal for awarding me the prestigious scholarship to carry out my Ph.D. and also for providing funding support to carry out my research activities within the country and abroad.
I would like to thank Dr. Patrick Waldemar, Vice President of Telenor ASA, Headquarter, Fornebu, Norway, for providing me an opportunity to join their research team as a Visiting Scholar for further extension and development of my research work. I am also thankful to my thesis advisor Prof. Dr. Rui. L. Aguiar for planning and organizing this working collaboration. I would like to thank Mr. Håkon Lønsethagen and Dr. Min Xie for hosting and advising me at Telenor, reviewing my manuscripts, and providing thought-provoking, open-ended discussions. Working with them was a pleasure and I thoroughly enjoyed my stay at Telenor as a Visiting Scholar.
Finally, I am wholeheartedly thankful to my Mother, Wife, Brothers, Sisters, and Friends for their support and encouragement most of the time.
Asad Ur Rehman
Aveiro, Portugal.
Oct, 2020.

Palavras Chave

Redes definida por software, virtualização de funções de rede, computação em nuvem, redes definidas por software e virtualizadas, tolerância a falhas, redes 5G confiáveis, redes híbridas, garantia de serviço e escalonamento automático.

Resumo

As redes definidas por software e as redes virtualizadas estão a atrair uma atenção significativa, tanto da comunidade de investigação académica como da indústria, devido aos seus impactos construtivos previstos para o desenvolvimento de futuros sistemas de telecomunicações (redes de quinta geração (5G) e mais além). Os fornecedores de serviços de comunicações também terão de se envolver na transformação digital para se adaptarem à próxima geração de serviços de telecomunicações com base nas novas tecnologias de software associadas às redes 5G. Recentemente, vários estudos exploraram vários aspetos destas tecnologias, tais como as redes definidas por software (SDN), virtualização de funções de rede (NFV), computação em nuvem, e computação móvel/multi-acesso no acesso (MEC). Contudo, estão ainda em curso importantes desenvolvimentos de normalização, tanto no que diz respeito à arquitetura em si como ao apoio a ambientes de rede. Por conseguinte, o desenvolvimento de aplicações de rede para redes definidas e virtualizadas por software ainda não acelerou. O advento de tais redes apresenta novos desafios e também abre novos caminhos para o desenvolvimento de novas estratégias, arquiteturas e normas para permitir uma elevada fiabilidade, tolerância a falhas e garantias de serviço em redes 5G e mais além. Nesta linha, esta tese explora as tecnologias SDN, NFV e de computação em nuvem. Os objetivos são de compreender melhor as redes orientadas por software e virtualizadas, abordar os desafios da sua implementação arquitetural, explorar a tolerância a falhas e o apoio à garantia de serviços, delinear obstáculos à implementação comercial a nível conceptual e identificar futuras direções de investigação para o desenvolvimento de redes virtualizadas e definidas por software. A este respeito, este trabalho propõe um projeto de controlo de redes fora da banda, com recuperação automática, centralizado, híbrido, e definido por software, para conceptualizar de forma fiável a gestão e automatização de futuras redes dinâmicas. Além disso, a tese avalia o desempenho das tecnologias de virtualização em cenários 5G fiáveis e apresenta um mecanismo de recuperação que pode permitir a obtenção de um tempo de paragem de serviço quase nulo. Além disso, descreve igualmente a implementação de componentes de qualidade de serviço e subsequentes testes de escalabilidade automática utilizando gestão e orquestração de código aberto, para alargar ou reduzir dinamicamente os recursos informáticos atribuídos a aplicações de rede, num ambiente de rede definido e virtualizado por software, em tempo de execução, ou conforme necessário. Globalmente, a tese centra-se em propor, avaliar, testar e validar soluções em termos de fiabilidade, tolerância a falhas e garantias de serviço, utilizando implementações baseadas em SDN, NFV e computação em nuvem para determinar a viabilidade e fiabilidade de redes definidas e virtualizadas por software.

Keywords

software-defined networking, network functions virtualization, cloud computing, software-defined and virtualized networks, fault-tolerance, Reliable 5G networks, Hybrid Networks, Service Assurance, and Auto-scaling.

Abstract

Software-defined and virtualized networks are attracting significant attention from both the academic research community and the industry, due to their envisaged constructive impacts on the development of future telecommunication systems (fifth-generation (5G) networks and beyond). Communication service providers will also need to engage in digital transformation to adapt to the next generation of telecommunication services based on the novel software-driven technologies associated with 5G networks. Recently, several studies have explored various aspects of these technologies, such as software-defined networking (SDN), network function virtualization (NFV), cloud computing, and mobile/multi-access edge computing (MEC). However, major standardization developments, regarding both the architecture itself and support for networking environments, are still ongoing. Hence, the development of network applications for software-defined and virtualized networks has not yet accelerated. The advent of such networks presents new challenges and opens new paths for the development of novel strategies, architectures, and standards to enable high reliability, fault tolerance, and service assurance in 5G networks and beyond. Along this line, this thesis explores SDN, NFV, and cloud computing technologies. The objectives are to better understand software-driven and virtualized networks, address challenges in their architectural implementation, explore fault-tolerance and service assurance support, delineate obstacles to commercial deployment at a conceptual level and identify future research directions for software-defined and virtualized network developments. In this regard, this work proposes an out-of-band, self-healing, centralized, hybrid, software-defined, networking control design to reliably conceptualize the management and automation of future dynamic networks. Furthermore, the thesis evaluates the performance of virtualization technologies in reliable 5G scenarios and presents a recovery mechanism that can achieve near-zero service downtime. In addition, it describes the implementation of service assurance components and subsequent auto-scaling testing, using open-source management and orchestration, to dynamically extend or reduce the computing resources allocated to network applications in a software-defined and virtualized networking environment, at run time, or as required. Overall, the thesis focuses on proposing, evaluating, testing, and validating solutions in terms of reliability, fault-tolerance, and service assurance using SDN, NFV, and cloud computing-based implementations to determine the feasibility and reliability of software-defined and virtualized networks.

Contents

List of Figures

List of Tables

List of Code Snippets

List of Abbreviations, and Acronyms

5G	Fifth-Generation
5G-PPP	5G Public Private Partnership Projects
AaaS	Application as a Service
ACLs	Access Control Lists
AFRO	Automatic Failure Recovery
APIs	Application Programmable Interfaces
ATM	Asynchronous Transfer Mode
AWS	Amazon Web Services
B2B	Business to Business
B2C	Business to Consumer
B5G	Beyond Fifth-Generation
BBF	Broadband Forum
BFD	Bidirectional Forwarding Detection
BFT	Byzantine Fault-tolerant
BLR	Backward Local Rerouting
BRAS	Broadband Remote Access Server
BSS	Business Support System
CAPEX	Capital Expenditure
CCSP	Cloud Computing Service Provider
CDNs	Content Delivery Networks
CGN	Carrier-Grade Network
CGNAT	Carrier-Grade Network Address Translator
CLI	Command Line Interface
CO_2	Carbon dioxide
COTS	Commercial-Off-The-Shelf
CPR	Control Plan Recovery
CPU	Central Processing Unit
C-RAN	Cloud Radio Access Network
CSPs	Communication Service Providers
DevOps	Development and Operation
DFRS	Declarative Failure Recovery System
DMTF	Distributed Management Task Force
DNSs	Domain Name Systems
DoS	Denial of Service
DPI	Deep Packet Inspection
DRAM	Dynamic Random Access Memory
E2E	End-to-End
EIGRP	Enhanced Interior Gateway Routing Protocol
eMBB	Enhanced Mobile Broadband
EMS	Element Management System
EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
FIB	Forwarding Information Base
FLR	Forward Local Rerouting
FML	Flow-based Management Language
ForCES	Forwarding and Control Element Separation
GENI	Global Environment for Networking Innovations
GUI	Graphical User Interface
HSA	Header Space Analysis
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IBM	Institute of Business Machines

ICMP	Internet Control Message Protocol
ICN	Information-Centric Networking
ICTs	Information Communication Technologies
IDS	Intrusion Detection Systems
IEEE	Institute of Electrical and Electronics Engineering
IMS	IP Multimedia Subsystem
IoT	Internet of Things
IPS	Intrusion Prevention Systems
ISP	Internet Service Provider
JSON	JavaScript Object Notation
KPIs	Key Performance Indicators
KVM	Kernel Based Virtual Machine
LAN	Local Area Network
LCM	Lifecycle Management
LLDP	Link Layer Discovery Protocol
LOS	Loss of Signal
LTE	Long-term Evolution
LXC	Linux Containers
M2M	Machine-to-Machine Communications
MAC	Medium Access Control
MEC	Multi-access Edge Computing
mMTC	Massive Machine-type Communication
MON	Monitoring Module
MOS	Mean Opinion Score
NAT	Network Address Translator
NETCONF	Network Configuration Protocol
NetSoft	Network Softwarization
NFV	Network Functions Virtualization
NFVI	Network Functions Virtualization Infrastructure
NFVI-PoP	NFVI Point of Presence
NFV-ISG	NFV Industry Specification Group
NFV-MANO	NFV Management and Orchestration
NFVO	NFV Orchestrator
NGDCN	Next-Generation Data Center Networking
NGNs	Next Generation Networks
NP	Non-deterministic Polynomial Time
N-PoP	Network Point of Presence
NSD	NS Descriptor
NSP	Network Service Provider
ONAP	Open Network Automation Platform
ONF	Open Network Foundation
OPEX	Operational Expenses
OS	Operating System
OSM-MANO	Open source Management and Orchestration
OSPF	Open Shortest Path First
OSS	Operation Support System
OvS	Open vSwitch
P4	Programming Protocol-independent Packet Processors
PaaS	Platform as a Service
PACs	Programmable Automation Controllers
PNFs	Physical Network Functions
PoC	Proof of Concept
POF	Protocol-Oblivious Forwarding
POL	Policy-management Module
PSTN	Public Switched Telephone Network
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
REST	REepresentational State Transfer
RHEV	Red Hat Enterprise Virtualization
RTT	Round Trip Time
SDN	Software-defined Networking
SFC	Service Function Chaining
SLA	Service Level Agreement
SMR	State Machine Replication
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SOC	System-on-a-Chip
SSD	Solid State Device
STS	SDN Troubleshooting System

TCAM	Ternary Content Addressable Memory
TCO	Total Cost of Ownership
TCP	Transmission Control Protocol
TOSCA	Topology and Orchestration Specification for Cloud Applications
TSDB	Time Series Data Base
TSPs	Telecommunication Service Providers
UDP	User Datagram Protocol
URLLC	Ultra-reliable and Low-latency Communication
VAC	Value-added Connectivity
vCDN	Content Delivery Network
vCPE	Virtual Customer Premises Equipment
VDCE	Virtual Data Center Embedding
VDU	Virtual Deployment Unit
vEPC	Virtual Evolved Packet Core
vFW	Virtual Firewall
VIM	Virtual Infrastructure Manager
VINI	Virtual Network Infrastructure
VLANs	Virtual Local Area Networks
VM	Virtual Machine
VMM	Virtual Machine Manager
VNE	Virtual Network Embedding
VNFD	VNF Descriptor
VNFM	Virtual Network Function Manager
VNFs	Virtual Network Functions
VoIP	Voice Over Internet Protocol
VPNs	Virtual Private Networks
VXLAN	Virtual eXtensible Local Area Network
WAN	Wide Area Network
WIMs	WAN Infrastructure Managers
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
YAML	Yet Another Markup Language
YANG	Yet Another Next Generation Language

Chapter 1

Introduction

Chapter Outline: This chapter provides a brief introduction to the topics, underlying ideas, and motivation to pursue this work. Furthermore, it highlights the main contributions to the body of knowledge and resulting scientific publications during the development of this thesis. Finally, the entire document structure of this thesis is presented in this chapter.

1.1 Introduction and Motivation

The requirements for supporting Information Communication Technologies (ICTs) have recently evolved as never before. Due to the continuous growth of network devices, more and more devices will be connected simultaneously to ICTs infrastructure. To accommodate these devices requires re-designing the network architecture to support the future computing scenarios they enable. As a consequence, Next Generation Networks (NGNs) research is building its foundation based on multiple evolving technologies. These include technologies like Network Functions Virtualization (NFV) [1], Software-defined Networking (SDN) [2], Cloud Computing [3], Internet of Things (IoT) [4], Information-Centric Networking (ICN) [5], and the Fifth-Generation (5G) [6] of telecommunications networks, all reputed to transform ICTs infrastructure and fulfill the future needs of computing. It is important to note that all these technologies are directed towards supporting the growing trend of network programmability and network softwarization of telecommunications systems.

Architectures based on the traditional network are tightly integrated because of the vendor's specialized hardware and customized software's based systems. Thus, proprietary-based implementations can offer limited programmability support, which means that launching new network services for customers (i.e., service provisioning) is not only difficult but also time-consuming and requires specialized knowledge.

Network administrators manually configure multiple network appliance interfaces, and it varies not only across different vendors but even with the same vendor for different products and services. The network administration process remained stable in the last decade because to deploy a network the administrator would ultimately resort to the traditional per box configuration through Command Line Interface (CLI). Traditionally networks followed proprietary-based network appliance approaches (also known as middle-boxes) [7]. Typical examples are load balancers, firewalls, Network Address Translator (NAT), and Wide Area Network (WAN) accelerators. Changing of proprietary-based hardware protocol and instantiating new service deployments is time-consuming and hard because it is difficult to update a protocol running on proprietary-based network appliances [8]. This involved specialized manual human intervention to perform network management operations and slowed down development in network automation [9]. Network automation is a practice in which software automatically configure and test network devices to reduce human error, network operation cost and save time for troubleshooting a network.

Undoubtedly, the trend of programmability is changing the traditional way of customizing and managing the network to be more efficient with standardized Application Programmable Interfaces (APIs). This approach can support customized scripting through which configuration of various vendor equipment's will become easier. This offers a simpler environment for network administrators to automate network efficiency with minimum human intervention. This is the practice known as network programmability [10]. The near future networks will be programmable, more organized, and more controllable [11] and NFV, SDN, and Cloud Computing technologies are considered key enablers to support such scenarios in such future paradigms.

Recently, SDN, NFV, and Cloud Computing based technologies have evolved leading to the rise of software-driven and virtualized networks. These software-driven and virtualized networks are designed to benefit from self-management mechanisms that lead towards the autonomic management of communications networks [12], [13]. Unlike traditional networks, the software-driven architecture based on mentioned evolving technologies allows Open source development of software that can run on generic shared hardware [14].

SDN, NFV, and Cloud Computing technologies are complementary to each other but are independent and can be deployed alone or together. As depicted in Fig. 1.1, combining these technologies in a unified network architecture is desirable [15]. Moreover, integrating these technologies adds value (flexibility and agility) to telecommunications systems, and it offers the freedom to Communication Service Providers (CSPs) to create and manage network services without worrying about configuring vendor-specific networking devices. Thus, this integration can offer enhanced privileges to CSPs to set up network services almost effortlessly.

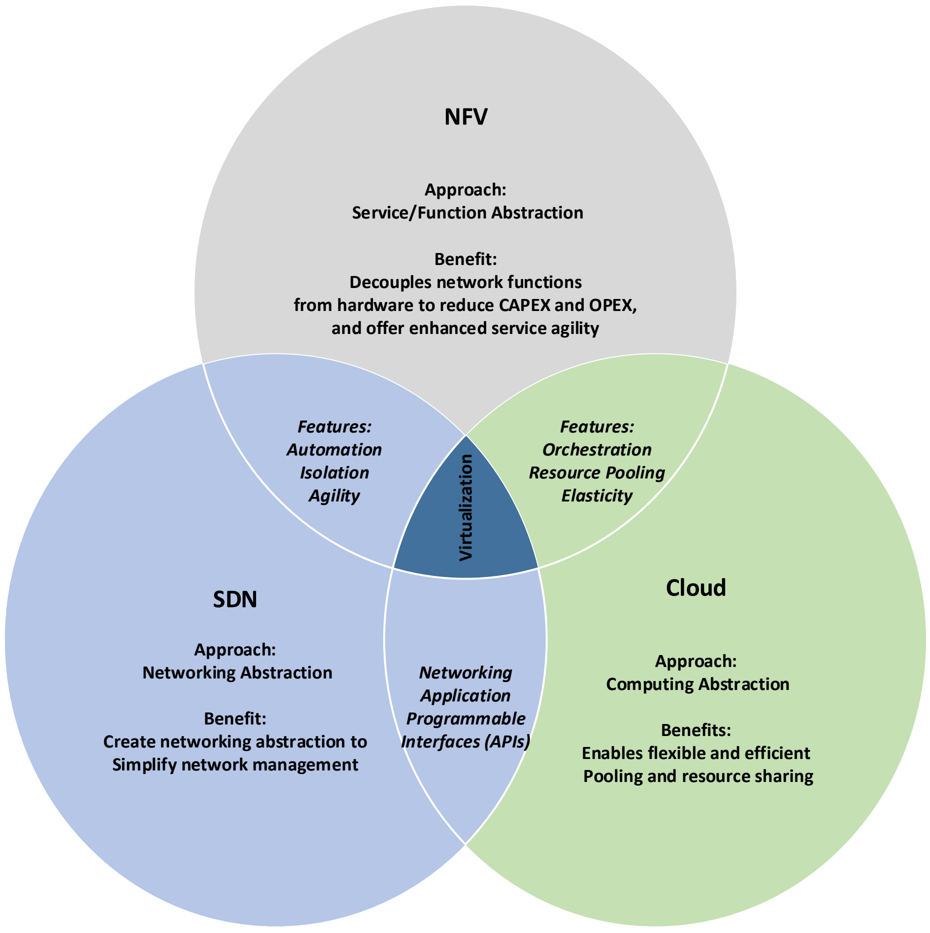


Figure 1.1: Relationship: NFV, SDN, and Cloud Computing [1]

As stated above those evolving technologies (NFV, SDN, and Cloud) based on software-driven and virtualized networks support the trend of network programmability and network softwarization of telecommunication systems. Next generation communication networks (network functions) are expected to be implemented on Open source virtualized infrastructures instead of current proprietary-based implementations. This is essential to support the 5G vision, calling for a new networking paradigm that directs flexible, dynamically configurable network elements in order to provide on-demand customized network services with enhanced service provisioning across the networks as well as supporting heterogeneity and diversity. CSP networks must undergo a transition from legacy physical components to a new virtualized infrastructure. During this transition, a complex Hybrid networking environment (involving both legacy and virtual technologies) will co-exist [16]. Hybrid Networks services will be based on SDN, Cloud, NFV, and legacy-based infrastructures. Therefore, Service Assurance would be devised to support evolving Hybrid Networks.

The management and orchestration of Virtual Network Functions (VNFs) are critical to Fifth-Generation (5G) and Beyond Fifth-Generation (B5G) software-driven and virtualized networks. Considering these aspects, the European Telecommunications Standards Institute (ETSI) proposed an Open source Management and Orchestration (OSM-MANO) framework. This framework offers End-to-End (E2E) management and orchestration solutions for 5G and B5G software-driven and virtualized networks. However, it is still in its infancy and yet to be commercially deployed.

The major problem faced today in the commercial deployment of software-driven and virtualized networks is the strict carrier-grade requirements [17]. Historically, the Public Switched Telephone Network (PSTN) architectures built on "Five 9s" ensuring 99.999% E2E reliability and availability. This is not only concerning meeting transmission "Five 9s" of reliability and availability but also with other requirements for Carrier-Grade Network (CGN) such as equipment management, security, billing, product development, and continuous operational support [18].

These evolving technologies undoubtedly offer flexibility, new network services, dynamic control, Open source solutions compared to traditional static and proprietary-based networks. However, the current solutions offered by these evolving technologies are still in their infancy. These evolving technologies solution aims to ultimately offer service reliability comparable to the "Five 9s" reliability of PSTN. However, validation and testing to achieve Quality of Service (QoS) equivalent to PSTN is still an intense area of investigation.

In a nutshell, on one hand, these evolving technologies based on software-driven and virtualized networks support the trend of network programmability and network softwarization of telecommunication systems. On the other hand, the reliability, fault-tolerance, and Service Assurance aspects of these evolving technologies have not yet matured and are lacking commercial deployments.

Next, the problem statement and studied research questions are presented.

1.2 Problem Statement

CSPs need a digital transformation in order to adapt properly to the next generation of telecommunication services that are based on novel software-driven technologies associated with 5G networks. In this research, we explored NFV and SDN technologies with a touch of Cloud Computing for evolving software-defined and virtualized networks. These concepts reflect different approaches: NFV provides network service or function abstractions, SDN provides network abstraction, while Cloud Computing provides computing abstraction. NFV promises to bring flexibility and cost reduction, SDN promises to bring programmable control and open interfaces, and Cloud Computing promises to bring flexible and efficient pooling and resource sharing of computing power. In this regard, this thesis is focused on proposing, evaluating, testing, and validating solutions in terms of reliability covering aspects of fault-tolerance and Service Assurance using SDN, NFV, and Cloud Computing based implementations to discover the feasibility and reliability of software-defined and virtualized networks.

1.2.1 Research Questions

NGNs research is building its foundation in new developments based on SDN, NFV, and Cloud Computing. Nevertheless, SDN, NFV, and Cloud Computing will themselves be placed under new and stringent requirements. For example, one of the main requirements for NGNs is inter-technology convergence, where network operations can impact access technologies of different kinds. In that respect, SDN, NFV, and Cloud Computing based implementations are overlooked, even though different architecture designs have started to manifest by the research community. It is imperative that the network remains effective in all conditions and scenarios, and its stability is not questionable. As such, there is a gap between the management of virtualization resources and the overall network communication infrastructure layer. As SDN and NFV are becoming coupled with Cloud Computing (e.g, OpenStack) and system virtualization, the problems are further compounded. In this line, the thesis aims to bring answers to the following questions:

Q1. What are the challenges and architectural impairments of software-defined and virtualized networks key enablers and key technologies namely, SDN and NFV?

Q2. How Service Assurance would be devised to support evolving 5G and B5G Hybrid Networks, a mix of both physical and virtualized networks?

Q3. Which virtualization technologies, namely containers and unikernels, envisioning the deployment to facilitate resilient communication networks in critical scenarios (5G environments)?

Q4. How to implement Service Assurance components and test auto-scaling using OSM-MANO to meet the network requirements (management and orchestration) of 5G?

1.2.2 Research Aims and Objectives

This research aims to address evolving software-defined and virtualized networks in terms of reliability, fault-tolerance, and Service Assurance using SDN, NFV, and Cloud

Computing based implementations. In light of this, the scientific merit of this research is pursued by accomplishing the following objectives:

i. Propose an out-of-band, self-healing, centralized, hybrid, software-defined, networking control design to reliably conceptualize the management and automation of future dynamic networks.

ii. Evaluates the performance of virtualization technologies in reliable 5G scenarios and proposes a recovery mechanism that can achieve near-zero downtime of service.

iii. Study the feasibility of Service Assurance components of software-defined and virtual network functions using OSM-MANO.

iv. Outline future research directions for software-defined and virtualized networks key enablers and key technologies development, namely, SDN, and NFV, as well as outline future research direction for SDN fault-tolerance, and Service Assurance for software-defined and virtualized network developments.

1.2.3 Advantages and Feasibility of this Research

Software-defined and virtualized networks offer several advantages to CSPs as compared to the existing hardware-based network functions [19]. The main advantages of transiting to software-defined and virtualized networks are as follows:

i. Enables the efficient use of ICTs infrastructure through softwarization of network functions. Hence, it brings more flexibility and agility and eases the management of telecommunications systems.

ii. Offers freedom to CSPs to create, deploy and manage network services without worrying about vendor-specific networking devices configuration, since, Virtual Network Functions (VNFs) are hosted on Commercial-Off-The-Shelf (COTS).

iii. Provide flexibility to adapt rapidly to technological innovation and provide a better return on investment for CSPs than the case of hardware-based appliances. As product lifecycles are becoming shorter, this can often become critical to support new network services.

iv. Software-defined and virtualized networks are important and useful for future networks for two reasons: firstly, they increase reliability and resilience without deploying dedicated physical architecture; secondly, they can potentially reduce Capital Expenditure (CAPEX) and Operational Expenses (OPEX) cost [1], [20], [21], [22].

v. Inherently, software-defined and virtualized networks can provide a reduction of inefficient energy consumption resulting in low Carbon dioxide (CO_2) thus supporting a green networking environment. Furthermore, as depicted in Fig. 1.2, this work supports four goals of Sustainable Development out of the seventeen defined goals by a nation united for global Sustainable Development [23].

11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
7 AFFORDABLE AND CLEAN ENERGY
• Sustainable Development Goals.
• Out of the 17 Goals, this work support 4 Goals of Global Sustainable Development.
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE

Figure 1.2: Sustainable Development Goals

1.3 Contributions, Scientific Publications, and Dissemination

This work evaluates the performance of virtualization technologies in reliable 5G scenarios and proposes a recovery mechanism that is able to achieve near-zero downtime of service. Following that, it proposes an out-of-band, self-healing, centralized, hybrid, software-defined, networking control design to reliably conceptualize the management and automation of future dynamic networks. Furthermore, to dynamically extend or reduce computing resources allocated to network applications in a software-defined and virtualized networking environment at run time, or when needed, it implements Service Assurance components and tested auto-scaling using OSM-MANO.

The list of contributions in terms of scientific publications in peer-reviewed high impact journals, international conferences, posters, public talks, and work progress disseminated is as follows:

1.3.1 Journals

i. A. U. Rehman, R. L. Aguiar, and João Paulo Barraca, "Network Functions Virtualization: The Long Road to Commercial Deployments", IEEE Access, vol. 7, pp. 60439-60464, 2019, DOI: 10.1109/ACCESS.2019.2915195.

ii. A. U. Rehman, R. L. Aguiar, and João Paulo Barraca, "Fault-tolerance in the Scope of Software-defined Networking (SDN)", IEEE Access, vol. 7, pp. 124474-124490, 2019, DOI: 10.1109/ACCESS.2019.2939115.

iii. A. U. Rehman, R. L. Aguiar, João Paulo Barraca, Håkon Lønsethagen, and Min Xie, "Service Assurance for 5G/B5G Hybrid Networks: Requirements, Key Considerations, and Challenges", (submitted for publication in Elsevier Journal).

iv. A. U. Rehman, R. L. Aguiar, and João Paulo Barraca, "Fault-tolerance in the Scope of Cloud Computing", (submitted for publication in IEEE Journal).

1.3.2 Conferences

v. A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Testing Virtual Network Functions Auto-Scaling using Open source Management and Orchestration", 12th conference on telecommunications, Telecoms Conference (ConfTELE), Leiria, Portugal, pp. 1-6, February 2021, DOI: 10.1109 ConfTELE50222.2021.9435471.

vi. J. B. Filipe, F. Meneses, A. U. Rehman, D. Corujo, and R. L. Aguiar, "A Performance Comparison of Containers and Unikernels for Reliable 5G Environments", 15th International Conference on the Design of Reliable Communication Networks (DRCN), Coimbra, Portugal, pp. 99-106, March 2019, DOI: 10.1109/DRCN.2019.8713677.

vii. A. U. Rehman, R. L. Aguiar, and João Paulo Barraca, "A Proposal for Fault-tolerant and Self-healing Hybrid SDN Control Network", Inforum2017- 9º Simpósio de Informática, Aveiro, Portugal, pp.16-23, October 2017.

1.3.3 Conference Posters

viii. A. U. Rehman, João Barraca Filipe, R. L. Aguiar, Daniel Corujo, and João Paulo Barraca, "Shaping Future Reliable and Critical Communications Using Virtualization Technologies", Encontro Ciência, Lisbon, Portugal, July 2018.

ix. A. U. Rehman, R. L. Aguiar, and João Paulo Barraca, "Reliability in Software-defined Networks", 10th Map-Tele workshop, Aveiro, Portugal, September 2017.

1.3.4 Public Talks

x. Topic: "Open source Management and Orchestration (OSM-MANO) and Service Assurance", Venue: Telenor Research ASA, HQ, Norway Fornebu, Dec 2019.

xi. Topic: "Reliability in Software-defined Networks", Venue: Universidade de Aveiro, Research Summit, July 2019.

xii. Topic: "Resilience in the scope of Software-defined Networks", Venue: IT - Instituto de Telecomunicações, Aveiro, Jan 2017.

xiii. Topic: "Fault-Tolerant and Self-Healing SDN Control Network", Venue: IT - Instituto de Telecomunicações, Aveiro, July 2017.

1.3.5 Participation in Research Projects

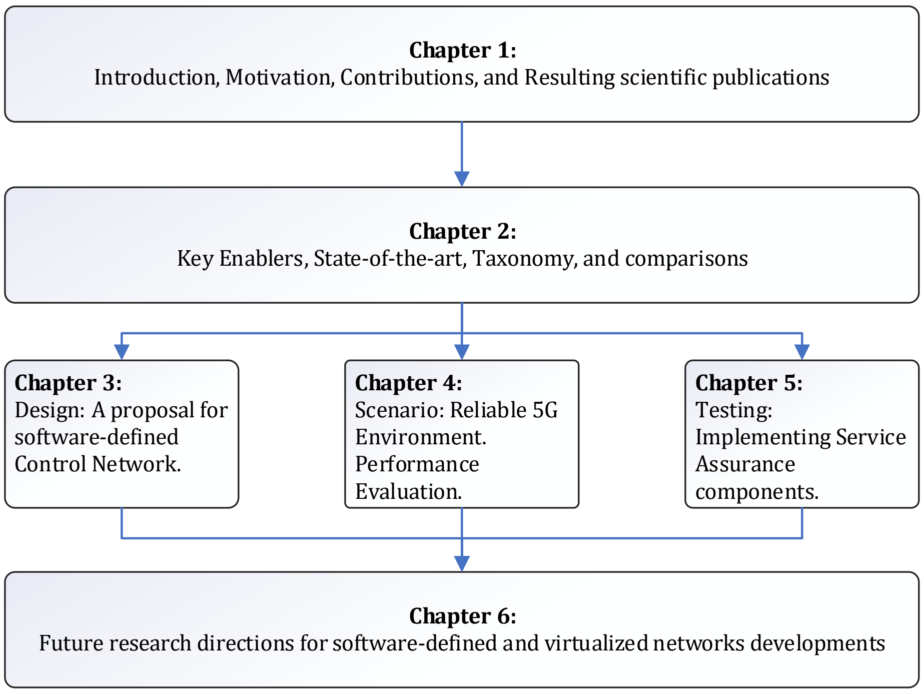
xiv. The SERENE Project: Proposal writing: A. U. Rehman, R. L. Aguiar, João Paulo Barraca, Title: "SDN Architecture-based reliability issues and solutions".

xv. The 5GO Project: Deliverable writing and contribution to project demonstrations: João Barraca Filipe, A. U. Rehman, and Daniel Corujo, Title: "Requirements Analysis and Mechanisms for Reliable Data Mobility in Machine-to-Machine Communications (M2M) Environments".

xvi. The SOCA Project: Deliverable writing and contribution to project demonstrations: A. U. Rehman, R. L. Aguiar, João Paulo Barraca, Title: "Integration of Cloud Computing and Virtual Networks".

1.4 Thesis Structure and Organization

The block structure of the thesis document is depicted in Fig 1.3. The thesis is composed of six chapters, starting with this introductory chapter which outlines the motivation for driving this work, research aims and objectives, contribution, dissemination, and resulting scientific publications. The remainder of the thesis is organized as follows:



Chapter 1:
Introduction, Motivation, Contributions, and Resulting scientific publications
Chapter 2:
Key Enablers, State-of-the-art, Taxonomy, and comparisons
Chapter 3:
Design: A proposal for software-defined Control Network.
Chapter 4:
Scenario: Reliable 5G Environment. Performance Evaluation.
Chapter 5:
Testing: Implementing Service Assurance components.
Chapter 6:
Future research directions for software-defined and virtualized networks developments

Figure 1.3: Block Structure of Thesis Document

- **Chapter 2: Key Enablers and The State-of-the-Art.**
  Presents an overview of the software-defined and virtualized networks key enablers and key technologies explored, namely NFV, and SDN throughout the proposed work. The requirements, concepts, design goals, main architectural impairments, and state-of-the-art research efforts are discussed in this chapter. In order to position our work, we review fault-tolerance in the scope of SDN and Service Assurance for Hybrid Networks.

- **Chapter 3: SDN Control Network.**
  This chapter proposes out-of-band control in SDN with additional in-band control (Hybrid SDN control network) to offer fault-tolerance support for SDN control plane reliability. The proposed resilient design ensures reliability between the SDN controller and underlying devices. The concept of a Hybrid SDN control network fulfills the fault-tolerance requirements for future heterogeneous networks made up of OpenFlow and traditional switches.
- **Chapter 4: Reliable 5G Networks.**
  This chapter presents a comparison of the use of two virtualization technologies, namely containers and unikernels, envisioning the deployment of VNFs close to the end nodes to contribute to the facilitation of resilient communication networks in critical scenarios (5G environments). Next, it addresses the concern that the probability of failure increases with the hardware limitation, imposing the development of failure detection and recovery mechanisms. In this line, the developed failure detection and recovery mechanism able to ensure VNF reliability by dynamically instantiating a backup VNF before failure, and using SDN to redirect the necessary data flow is presented.
- **Chapter 5: Auto-scaling Testing.**
  In this chapter, we explored comprehensively NFV deployment using OSM-MANO and OpenStack. Furthermore, we explored OSM-MANO Service Assurance and demonstrated auto-scaling using OpenStack and OSM-MANO. We then identified the research gaps, lessons learned for OSM-MANO Service Assurance development.
- **Chapter 6: Future Works.**
  This chapter concludes by enumerating future research directions for software-defined and virtualized network key enablers and key technologies development, namely, SDN, and NFV. Furthermore, it also outlines future research direction for SDN fault-tolerance, and Service Assurance for Hybrid Networks development.

Chapter 2

Key Enablers and The State-of-the-Art

Chapter Outline: This chapter first provides a detailed background of NFV to establish a comprehensive understanding of the subject, ranging from the basics to more advanced topics. It also highlights NFV requirements, design considerations, developmental architectural impairments, and barriers to commercial NFV deployments. Second, it addresses SDN fault-tolerance and highlight SDN-specific fault-tolerance issues and provide a comprehensive overview of the state-of-the-art SDN fault-tolerance research efforts, and structure SDN fault-tolerance research according to three distinct SDN planes (i.e., data, control, and application). Finally, this chapter addresses Service Assurance requirements, design goals, and key considerations in the context of Hybrid Networks.

2.1 Importance of Virtualization Technologies

Virtualization is a well-known concept, as virtualization can be claimed to have started in the 1960s when the Institute of Business Machines (IBM) introduced an Operating System (OS) named CP-40 (the first operating system that implements complete virtualization). The purpose of this was to implement time and memory sharing across users and applications in mainframe computers. This Mainframe virtualization concept laid down the foundation for virtualization that exists today [24]. Today, we define virtualization as a technology that provides an abstract view of underlying resources (hardware systems). This abstract view enables the creation of multiple simulated environments running multiple OS and applications on top of (potentially) different physical hardware systems. Virtualization can be applied in several ways to achieve different goals in computing, storage, and networking. For instance, the concept of virtualization can be applied to achieve virtualization of data, desktop, server, the OS, and network functions [25].

2.1.1 Virtualization, Network Virtualization and Network Functions Virtualization

Network Virtualization is the process of combining software, hardware resources, and network functionalities into a unified administrative domain known as a virtual network. One of the first initiatives of network virtualization was the Tempest project [26], which introduced the concept of switchlets in Asynchronous Transfer Mode (ATM). This approach was quickly followed by a diversity of overlay projects over the internet, such as MBone (for multicast) [27], the 6bone (test-bed for IPv6) [28], the X-Bone (automatic virtual networking) [29] and many others carried out by several projects including PlanetLab (geographically distributed an open platform for deploying, evaluating, and accessing planetary-scale network services) [30], Global Environment for Networking Innovations (GENI) [31] and Virtual Network Infrastructure (VINI) [32] (these last examples developed for experimentation, testing and validation of new concepts at scale). Furthermore, network virtualization plays a significant role throughout the evolution of the programmable network.

Network virtualization provides a logical abstracted view of the physical infrastructure. These logical networks run over shared infrastructures, leading to a reduction in CAPEX and OPEX. Overlay networks as above are one form of network virtualization. There are many forms of overlay networks [33]. For instance, a common example is a Virtual Private Networks (VPNs), created by the network administrator as a dedicated network to connect multiple sites through secure tunnels over public networks. Another examples are Virtual Local Area Networks (VLANs), acting as a private Local Area Network (LAN). VLANs can run over the same infrastructure of the normal network (i.e., switches and routers). VLANs can provide efficient traffic isolation for up to 4096 logical networks as specified in the Institute of Electrical and Electronics Engineering (IEEE) 802.1q VLANs tagging [34], [35]. Unfortunately, VLANs do not scale, as it is increasingly hard to configure and manage when it comes to dividing one physical resource into multiple isolated virtual environments. Due to this scalability issue with VLANs [36], Virtual eXtensible Local Area Network (VXLAN) have been developed to overcome practical network limitations of the VLANs using an overlay-based network virtualization approach [37]. Currently, IEEE 802.1aq supports more than 16 million possible virtual networks as compared to 4096 possible virtual networks available with IEEE 802.1q VLANs tagging.

The idea of an overlay network is quite old. Internet services started to run on top of the telephone network. Hence, an overlay acts as a computing paradigm of virtualization [38], [39]. Typically, in an Internet Service Provider (ISP) several overlay networks are running over the same network infrastructure to offer different services (Voice Over Internet Protocol (VoIP), video and broadband, for instance). These services can be logically separated within the same network infrastructure. Therefore, ISPs can save network infrastructure deployment, management, and maintenance costs by sharing some infrastructure to support multiple services.

Network Virtualization is usually confused with NFV. For clarity, the main difference is that Network Virtualization provides virtualized networking at layer 2 and on layer 3, while NFV aims to provide virtualized networking at layers (4-7). This approach enables the softwarization of protocol stacks beyond existing network virtualization solutions, thus allowing CSPs to operate, configure and deploy fully virtualized networking environments with flexibility and agility. NFV has been proposed to assist CSPs to get rid of proprietary-based networking appliances. Furthermore, NFV moves the cost of specialized hardware-based middle-boxes (Layers 4-7) network functions to more flexible and programmable customized pre-packaged software-based VNFs. The disruptiveness of NFV is illustrated in Fig. 2.1, which shows a comparison of traditional hardware-based appliances approach versus the NFV approach [19].

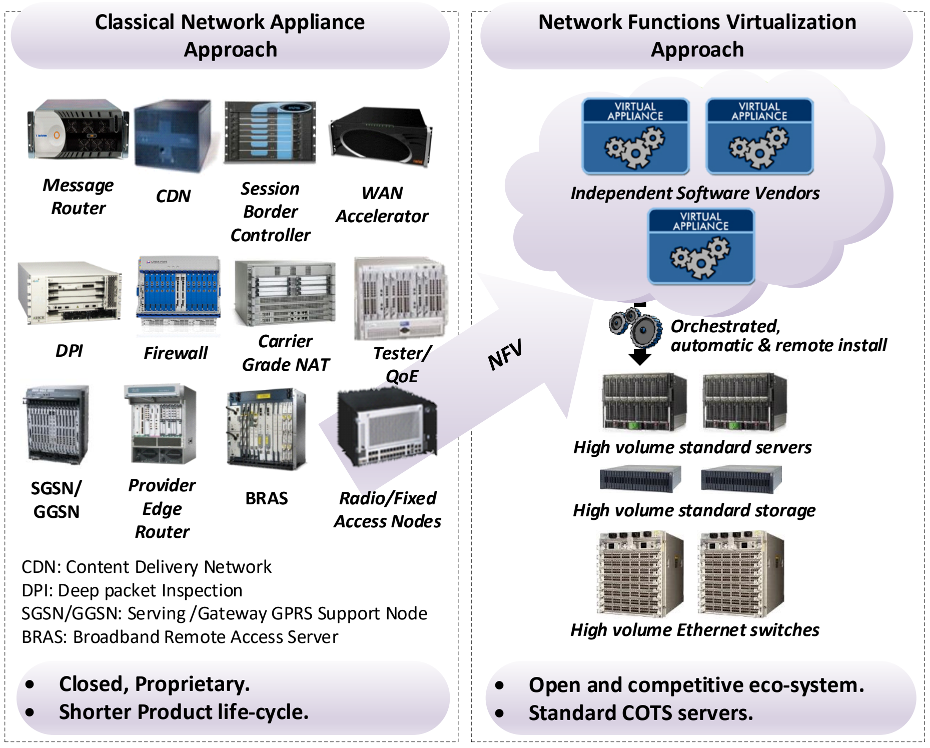

Classical Network Appliance Approach
Message Router
CDN
Session Border Controller
WAN Accelerator
DPI
Firewall
Carrier Grade NAT
Tester/ QoE
SGSN/ GGSN
Provider Edge Router
BRAS
Radio/Fixed Access Nodes
CDN: Content Delivery Network
DPI: Deep packet Inspection
SGSN/GGSN: Serving /Gateway GPRS Support Node
BRAS: Broadband Remote Access Server
• Closed, Proprietary.
• Shorter Product life-cycle.
NFV
Network Functions Virtualization Approach
VIRTUAL APPLIANCE
VIRTUAL APPLIANCE
Independent Software Vendors
VIRTUAL APPLIANCE
Orchestrated, automatic & remote install
High volume standard servers
High volume standard storage
High volume Ethernet switches
• Open and competitive eco-system.
• Standard COTS servers.

Figure 2.1: Traditional Hardware-based Network Appliances Approach Versus NFV Approach

2.1.2 Server Virtualization and Hypervisors:From Server Proliferation to Virtual Machine Development

In this section, we present concepts and terminologies related to virtualization technologies.

Traditionally, each application is required to run a single server and the server to run continuously, even when the server is not used to its fullest capacity (hardware resources) by the application. Therefore, this has led to the infrastructure problem known as server proliferation. This problem was due to two reasons: First, the number of servers was growing, and second, these servers were highly underutilized. Furthermore, operational challenges such as power and cooling systems for servers, as well as operational expenses to own or buy a place to support such infrastructure became increasingly costly. On top of that, additional servers for backup (probably in different locations) increase even more infrastructure costs for infrastructure owners. The development of the Virtual Machine (VM) concept has fixed the server proliferation problem by consolidating servers through virtualization as shown in Fig. 2.2 [40]. Due to this, it was possible to use servers more efficiently, offering cost saving for infrastructure owners, service providers, and businesses. In short, the VM development has fixed the server proliferation challenge.

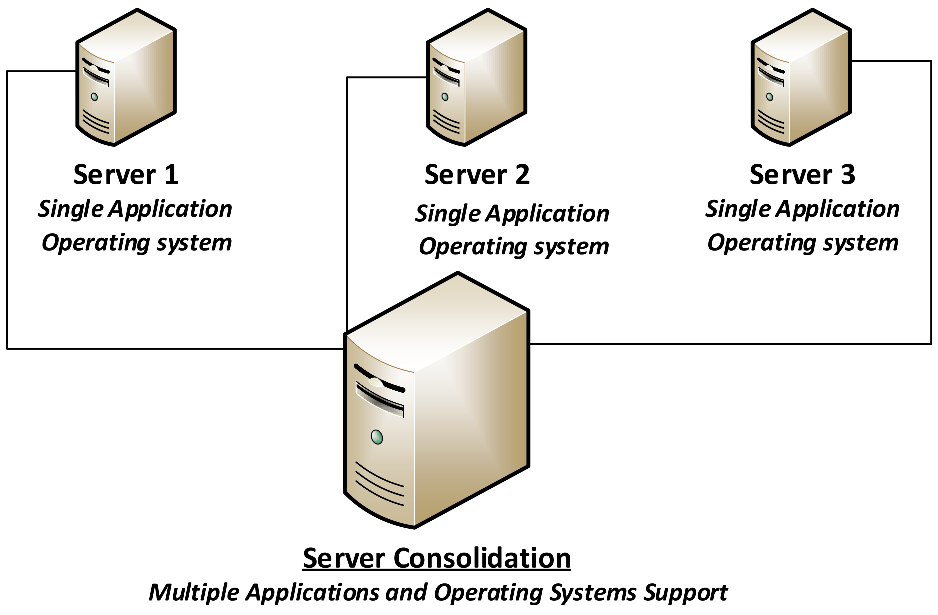


Figure 2.2: Server Consolidation

VMware defines VM as follows: "A VM is a tightly isolated software container that runs its own OS and applications as if it were a physical computer [41]." The three main components of a VM are: the host OS, the hypervisor or Virtual Machine Manager (VMM), and the guest OS [42].

A host OS is directly installed on the physical hardware.

The hypervisor is not a new concept; it was introduced in the 60s to run different OSs on a single mainframe computer. A hypervisor is a software program that is capable of hosting different VMs with different OSs installed and running over the same single hardware resources. Hence, it has the flexibility to support several VMs with multiple OS and applications running on a single hardware resource. Moreover, a hypervisor is responsible for resource allocation to the VM as well as responsible for monitoring and managing VMs through coordination with the underlying hardware primary OS.

Hypervisors are divided into two types: Type I and Type II, as shown in Fig. 2.3.

Type-I hypervisors, also known as bare metal or native or embedded hypervisors (hardware-based hypervisors), do not need any host OS because the communication to hardware resources is direct with full visibility of hardware resources [40]. Currently, there are several Type-I hypervisors in the market, with different flavors led by different vendors (for instance, Microsoft Hyper-V, Open source Kernel Based Virtual Machine (KVM), Xen/Citrix Xen Server, Red Hat Enterprise Virtualization (RHEV), and VMware vsphere/ESXI).

The type II hypervisors, also known as hosted or embedded hypervisor (software-based hypervisor), requires a host OS because the type II hypervisors run on top of the supported OS (an additional layer that interacts with the underlying hardware resources in order to manage VM/Server). Currently, there are several Type-II hypervisors, such as (Oracle virtual box, VMware Workstation, and Microsoft Virtual PC) [43].

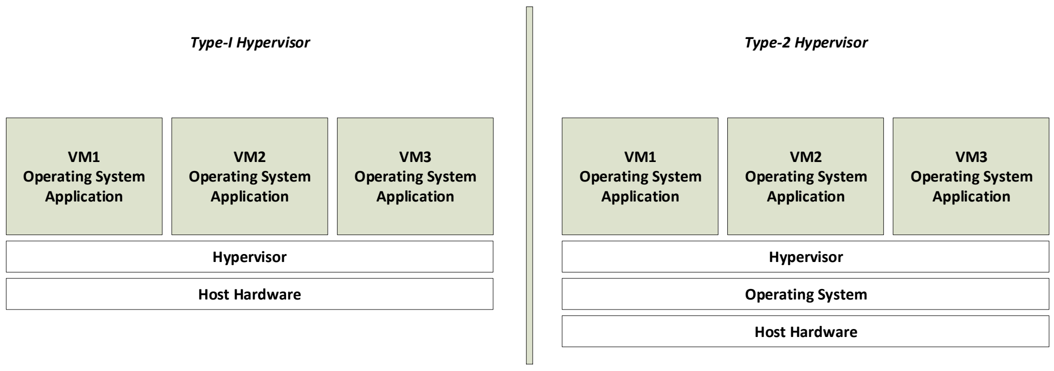

Type-I Hypervisor
VM1 Operating System Application
VM2 Operating System Application
VM3 Operating System Application
Hypervisor
Host Hardware
Type-2 Hypervisor
VM1 Operating System Application
VM2 Operating System Application
VM3 Operating System Application
Hypervisor
Operating System
Host Hardware

Figure 2.3: Comparison Type-I Versus Type-II Hypervisors [44]

Type-I hypervisors are more secure than Type-II, are faster and more efficient. Type-I hypervisors sit on hardware and communicate directly without any additional virtualization layer. However, they are hard to set up. The Type-II hypervisors are less secure as compared to Type-I. These types of hypervisors are slightly slower and less efficient because an additional layer is needed to manage VM indirect communication to hardware. However, they are easy to set up. Indeed, the different types of hypervisors utilize different virtualization techniques and would be classified based on their virtualization techniques. Thus, Hypervisors are an integral part of any research on networks virtualization [45]. We summarize the features of Type I and Type II hypervisors in Table 2.1.

Table 2.1: Type I and Type II Hypervisor Comparison

Type I Hypervisor	**Type II Hypervisor**
Directly runs on server hardware	Runs on top of the supported OS
Minimizing overhead due to direct hypervisor interaction with hardware resources.	Incur overhead as hypervisor runs on top of the supported OS.
Provide better hardware resource utilization	Provide less hardware resource utilization.
More secure due to hardware-based hypervisor	Less secure due to a software-based hypervisor.
Hard to Set up	Easy to Set up.
Examples: vSphere, XenServer, Hyper-V, KVM.	Examples: VMware Workstation, VMware Player, Microsoft Virtual PC, Oracle Virtual Box, and Free BSD, etc.

As explained above, VMs created on the top of a hypervisor layer acts as a virtual server running different OSs and applications. Thus, it requires an operating system to boot up, manage the devices and applications within the virtual server environment. This is known as a guest OS. Unlike the host OS, the guest OS does not need any modification to run on VMs, therefore, does not have precise visibility of the underlying hardware. However, hypervisor manages application requests from users that are supported by the guest OS through an additional layer and map these requests to "physical" hardware or host OS, and allocates resources, in such a way that it seems that guest OS is directly interacting to physical hardware or host OS.

2.1.3 Containerization (Lightweight Virtualization)

The VM concept and implementations discussed earlier fixed the problem of server proliferation. However, this method still imposes a performance and resource cost due to the overhead associated while imitating the hardware into a virtual environment with high-level isolation and non-shared host Kernel/OS to create a VM. To cope with this overhead and hypervisor performance degradation, a lighter packaged/Kernel-based virtualization with low-level isolation and shared Kernel OS can be used instead. This is called container-based virtualization or containerization [46]. Containers are sometimes referred to as Linux Containers (LXC) because of their origin on the Linux Kernel. Nevertheless, not all containers are Linux-based containers. Containerization differs from

VMs because it provides policy-based segregation of system resources usage. Containers offer superior performance as compared to the VMs [46], [47] because hypervisors are not being used but instead lightweight APIs within the Kernel. This bypasses the overhead created during hyper visors interaction in VMs environments thus, offering enhanced performance. Due to the shared Kernel, container-based virtualization is much less secure than VM virtualization.

In addition to that, containers can also be deployed in VMs to provide multi-tenant isolation. The concept of 5G network slicing is an example of such isolation where multi-tenancy can be supported by slicing underlying physical infrastructure [48], [49].

2.1.4 Unikernels

"Unikernels are specialized, single-address-space machine images constructed by using library OSs" [50], [51]. These specialized images can then be run on standard hypervisors. The footprint of a unikernel is considerably smaller than a VM or containers, and thus can provide better performance [52]. Unikernels are designed to be able to run a single process. They are also not meant to be multi-user or multi-process. Thanks to this single-minded design, a unikernel is small, lightweight, and quick [53]. Open source work on unikernels includes projects such as ClickOS, IncludeOS, and MirageOS [54].

As discussed above, virtualization approaches (VM, container, and unikernel) are critical when it comes to applying them to NFV. To explain this, we illustrated the internal architectures of these virtualization technologies and their possible level of implementation in Fig. 2.4. Also, In Table 2.2 we have provided a brief comparison of

these virtualization technologies, which can be useful to determine (from the network designing perspective) which of the virtualization technology can be better suitable (concerning performance and deployment cost) for different opted scenarios within virtual networking environments [55].

Table 2.2: Virtual Machines Versus Containers Versus Unikernels

Virtual Machines	**Containers**	**Unikernels**
Heavyweight	Lightweight	Tiny
Useful when power and storage are not an issue	Useful when power and storage become critical	Useful when power and storage are in short supply
Run its own OS	Shared Kernel OS	Specialized, single addressed-space machine images using library OS
Limited performance	Superior performance	Superior performance
Fully isolated and hence more secure	Shared Kernel-based isolation hence less secure	Fully isolated and hence more secure
Booting time in minutes	Booting time in milliseconds	Booting time in milliseconds
High overhead due to hypervisor interaction	Low overhead due to lightweight APIs instead of using hypervisor	Minimal overhead due to specialized library OS
Supports multiple applications at a time	Supports multiple application at a time	Supports single application at a time

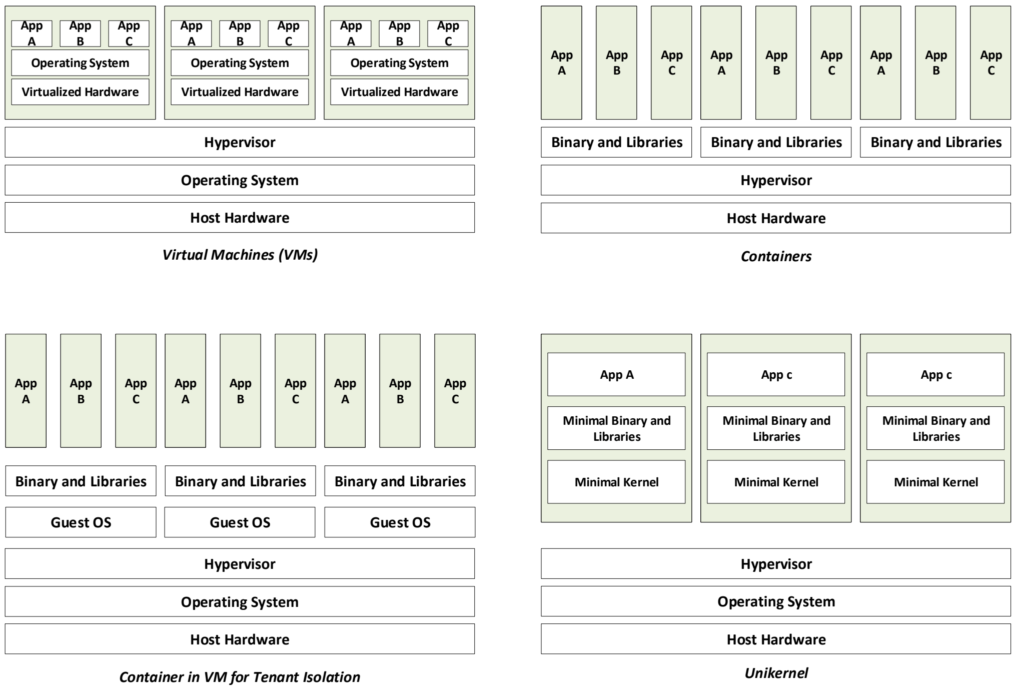


Figure 2.4: Virtualization Technologies Comparison

2.2 NFV Comprehensive Overview

The collaborative work on NFV officially started in October 2012 when leading Telecommunication Service Providers (TSPs) produced a white paper [56] jointly, highlighting the NFV concept and its benefits, and calling for industrial research actions. Moreover, in November 2012, AT&T, British Telecom, Verizon and other leading Telecom network operators developed an Industry Specification Group for NFV inside European Telecommunications Standards Institute (ETSI). Since then, the ETSI NFV Industry Specification Group (NFV-ISG) became the home of the Industry Specification Group for NFV [57]. Further, ETSI is working and primarily has accomplished work of "level-one" with the publication of the first five ETSI Group Specifications documents in October 2013 [58]. The first four were focused on NFV across the industry and the fifth was about promoting and coordinating public demonstrations, i.e., Proof of Concept (PoC). In 2014, eleven other documents were published, and the first phase was completed as pre-standardization work.

The first phase ("Release 1") includes an overview of infrastructure update, architectural framework, hypervisor, and domain of network infrastructure. Furthermore, these specifications also covered aspects of NFV Management and Orchestration (NFV-MANO), security, reliability, resilience, and QoS metrics [59].

The follow-up "Release 2" was then focused on the inter-networking of equipment and services, addressing functional blocks requirements, including ETSI NFV architecture framework interfacing and reference points. The "Release 2" documentation was completed in 2016.

The work on "Release 3" started in 2017, in parallel to the protocols and data models; network service descriptor file structure specifications, and the implementable protocol and data model solutions of interfaces and other artifacts. NFV "Release 3" was focused on enriching the architectural framework in order to make NFV ready for global deployment and operation. Some of the "Release 3" features were completed in July 2019. Some features had been closed, and some others were carried over to "Release 4".

The work on "Release 4" was officially launched in mid of 2019. ETSI NFV "Release 4" is underway, and all the previous evolutions show that NFV is evolving very rapidly." Release 4" focuses on new features such as network connectivity integration, NFV-MANO automation, and autonomous networks, NFV enhancements for 5G, Multi-tenancy enhancements for NFV-MANO, service-based architecture for NFV-MANO, VNF generic management functions continuous VNF integration, and Policy models. NFV already moved from the conceptual framework to a PoC stage. The "Release 4" documentation is expected to be completed in mid of July 2021.

NFV provides Telecom network operators with the advantage to combine and expand their current networks with smooth evolution. NFV-based solutions and research activities span around two main areas: TSPs and Next-Generation Data Center Networking (NGDCN) [60]. The scope for NFV to transform operator network architecture includes scalability, high-performance backbone, Overlay VPN and internet services, amongst others.

Currently, ten working groups are exploring different aspects of NFV architectural framework. We list the details of these working groups in Table 2.3. ETSI has also formed

Table 2.3: ETSI NFV Research Areas and Working Groups

Working Groups	**Research Area**
NFV-INF	Deals with architecture for the virtualization Infrastructure
NFV-MANO	Deals with the management and orchestration support.
NFV-SWA	Focuses on research purely on software architecture.
NFV-REL	This group focuses on the aspect of reliability, availability, resilience and fault tolerance.
NFV-TST	Focuses on pre-deployment testing and validation of Open source NFV.
NFV-TSC	Technical steering committee group to keep an eye on NFV work and facilitate its acceleration.
NFV-IFA	Focuses on interfaces and architectural aspects of NFV reference framework.
NFV-NOC	Focuses on network operators councils to develop NFV solution based on their feedback.
NFV-SEC	Focuses on security aspects of NFV.
NFV-EVE	Focuses on evolutionary aspects and ecosystem strategies for NFV.

a NFV-ISG PoC forum. We list in Table 2.4 selected PoC modules demonstrated by different Telecom network operators in liaison with ETSI [61]. The open demonstration of the PoC at ETSI is intended to show that NFV is an operable technology. The demos listed in Table 2.4 suggest different business motivations from operators and vendors in validating different PoCs. Most of these demonstrations used implementations based on cloud technologies, (e.g., OpenStack). Subsequently, ETSI NFV-ISG works closely with TSPs and equipment vendors, to specify their requirements for NFV adaptation based on their working environment. This assessment is important to prevent interoperability problems in NFV standardization. Therefore, in reality, NFV is progressing rapidly to reshape the CGN services for ISPs shortly.

Table 2.4: NFV Industry Specification Group PoC Demonstrations

Organization	PoC Demos
British Telecom	Virtual Broadband Remote Access Server (BRAS)
Deutsche Telekom	Virtual IP Multimedia Subsystem (IMS)
Orange Silicon Valley	Virtual Evolved Packet Core (vEPC)
Telefonica	Carrier-Grade Network Address Translator (CGNAT) and Deep Packet Inspection (DPI)

2.2.1 Description of NFV Architecture Framework

Architectures based on the traditional network are tightly integrated because of the vendor's specialized hardware and customized software's based systems. Unlike the traditional network, the NFV-based architecture allows Open source development of software that can run on generic shared hardware [14]. In this section, we provide a detailed description of ETSI NFV reference architecture.

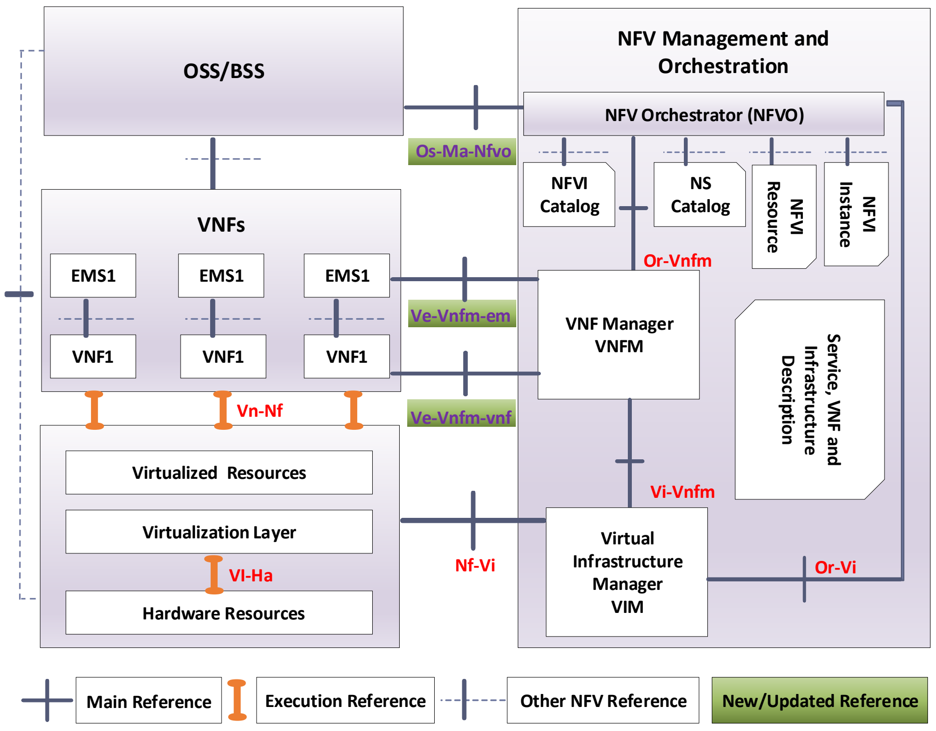


Figure 2.5: ETSI Architecture and Reference Framework for NFV

ETSI proposed the NFV architectural framework and identified functional blocks and the main reference points between the functional blocks as shown in Fig. 2.5 [20]. ETSI describes the NFV architectural framework at the functional level and does not propose any specific implementation. However, NFV architectural framework is proposed already considering the changes that possibly occur in an operator's network due to the network virtualization process (transition) [20]. Due to these expected changes that can occur in an operator's network, ETSI also defines NFV reference points to ensure consistent information exchange between functional blocks is guaranteed across vendor's implementations for functional blocks. The details of these functional blocks are as follows.

2.2.1.1 NFVI

The Network Functions Virtualization Infrastructure (NFVI) functional block is the combination of physical hardware (compute, storage, and network) and virtualized resources (abstracted view of computing, storage, and network). Generally, a hypervisor provides an abstraction to create a virtual environment over underlying infrastructure, in which VNFs can be deployed, managed and executed [62].

NFVIs can be geographically distributed and generally, VNFs deployment location may not be visible (i.e., it can be implemented using available physical resources across different geographical locations).

2.2.1.2 VNFs

The VNFs functional block is composed of multiple VNF and multiple Element Management System (EMS). VNF is the virtualization of legacy (hardware-based) NFs and EMS is responsible for the management aspects of these VNFs. A VNF can be deployed stand-alone in a single VM, or it can be deployed across multiple VMs. However, when VNFs are deployed collectively in a group to implement a specific network service, then it must be processed in a certain order due to the possibility that some of the functions have dependencies on others.

2.2.1.3 NFV-MANO

The NFV-MANO functional block is the management and orchestration framework required for the provisioning of the VNFs. It steers the deployment and operation of VNFs onto the NFVI [63]. Moreover, it has a database that stores information that can help determine the life-cycle properties of services and resources and resources.

2.2.1.4 OSS/BSS

This block is also responsible for coordinating with the traditional network system such as Operation Support System (OSS) and Business Support System (BSS) to ensure the NFV-MANO, NFVI and functions running on legacy equipment with pre-defined communications interfaces.

We have noticed some recent changes that have been made to the ETSI NFV architecture and reference framework. The changes that are being made to a revised ETSI architecture and reference framework include [64] the re-positioning of the "Service VNF and Infrastructure Description" which was moved inside the NFV-MANO. Previously this was outside the NFV-MANO block with reference point "se-Ma". Since "Service VNF and Infrastructure Description" is re-positioned the "se-Ma" reference point becomes obsolete. Also, new interfaces were defined for the MANO, including reference points explicitly re-positioned and renamed "Os-Ma" to "Os-Ma-Nfvo", "ve-vnfm" to "ve-vnfm-vnf" and new additional reference point "ve-vnfm-em" as depicted in Fig. 2.5. All these recent changes show that NFV is still undergoing major standardization developments.

2.2.2 Realization of NFV Architectural Implementation Challenges

In this section, we discuss ETSI NFV architectural implementation challenges. Providing a solution to these challenges can significantly improve NFV developments. The ETSI NFV-ISG group architectural framework has to define NFV architecture building blocks and reference points but has not yet indicated specified NFV implementation and PoCs. We discuss the challenges of the NFV architectural framework briefly as defined by ETSI.

2.2.2.1 NFVI: Network Function Virtualization Infrastructure

NFVI covers three-layer: Hardware resources, Hypervisor domain, and Virtualized resources. Moreover, this block supports a virtual environment where VNFs are executed and deployed over the underlying hardware resources. The virtualized environment is composed of servers, virtual machines switches, and virtual switches, etc.

We consider several main challenges for the functional blocks of NFVI as follows [65]:

- How hardware resources can be designed and utilized to translate the virtual environment efficiently?
- How to maintain and update a software-based environment (virtualized environment)?
- How to keep track of continuous development and integration of software that is interacting with underlying hardware resources?
- How can NFVI maintain connectivity between locations such as data centers and private/public or hybrid cloud environments?

The questions as above-mentioned arise because the standard procedure and implementation of NFVI are not yet fully regularized. The NFVI functional block is critical to NFV-based VNFs implementation such as vEPC, Content Delivery Network (vCDN), and Virtual Customer Premises Equipment (vCPE) [66]. Therefore, establishing Key Performance Indicators (KPIs) to ensure VNFs consistency and performance becomes challenging.

Hypervisor and hardware resources (compute, storage and network) are going to be provided by different vendors. Integrating and incorporating trust and security in a multi-vendor virtual environment is also challenging.

Managing virtualized resources is critical for NFVI. NFV mostly depends on the software, and developing a practice to maintain the continuous quality of software in NFVI is also challenging. However, it is well-known that VNFs are independently deployed through a cross-layer platform such as OpenStack, but additional tools for monitoring and managing such VNFs (deployed over the COTS hardware) need to be developed to successfully implement VNFs functionality across the network [67]. The above-mentioned challenges for ETSI NFVI blocks must be addressed before NFV pre-deployment testing and validation phases.

2.2.2.2 VNF's: Virtual Network Functions

The VNFs block consists of multiple VNFs and multiple EMS. Each VNF is assigned to a specific EMS to implement services in a virtualized environment. EMS system keeps track of associated VNFs configuration, and its monitoring while VNFs are running on single or multiple VMs. VNFs are created uniquely and in isolated virtual environments to meet the scalability, security and performance requirements. However, guaranteeing these aspects is challenging [65]. Since VNFs deployment utilized NFVI, three main challenges arise as follows: Portability, Resource allocation, and Performance.

First of all, the performance of VNFs depends on both the hardware and the software that builds together a virtual environment, where the concept of the NFV is practically realized [68], [69]. The VNFs must provide performance values on commodity servers similar to NFs running on hardware equipment. VNF software must be of high quality and must avoid performance bottlenecks, and maintain accountability at each layer of the virtual environment. Indeed, network service functions such as firewall, Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) are now virtualized to support multi-user multi-service environments (multi-tenancy). Thus, a physical switch can connect the service node to the network, but users are logically separated through virtual switching inside the multi-tenant environment.

This virtual switching is implemented in software to mirror the functionality of the physical switch and requires several operations such as encapsulation, de-encapsulation, NAT and policy that are implemented to shape traffic inside a multitenancy environment. This consumes significant computing resources that lead to performance degradation of the virtualized system (especially in high data rates environment). To avoid such consumption of resources there is a need to and increase the efficiency of virtual switching by developing high-performance virtualized systems and environments [70].

Another challenge is portability. The VNFs must be portable between servers and across the network. The live migration of VNFs must be possible without any performance degradation. Several cross-platforms such as OpenStack, Eucalyptus, oVirt, OpenNebula, and Nimbula are also working towards supporting the portability of VNFs in an NFV environment [67].

2.2.2.3 OSS/BSS: Operational and Business Support System

NFV is transforming the way the telecommunications infrastructure is deployed. Therefore, the way service is delivered by CSPs is going to change significantly. Thus, NFV inevitably is imposing new demands for OSS.

Traditionally, the OSS/BSS system is oriented to a reasonably static networking environment. However, in today's highly dynamic networking environment, the OSS/BSS system already needs some re-design to adapt it to the more dynamic nature of businesses. This is now compounded by the need to provide the operational and business support of deploying services using software-defined infrastructure [71]. CSPs need to advance their OSS system to simplify and align with evolving software-defined infrastructure. Moreover, to control the dynamicity of NFV, new mechanisms to keep track of performance need an enhanced service assurance system to handle the dynamic nature of the processes.

This service assurance system must be integrated with an OSS for two main reasons. First, to guarantee the SLAs are met, and second, to act as a tool to identify and manage network failures. In today's changing networking environments, service assurance must be adaptive in order to meet the requirements of future heterogeneous networks. At present, there is a need to modernized OSS/BSS systems according to the NFV, and new evolving technologies introduce to achieve enhanced automation, scalability, capacity optimization, and service elasticity in software-defined infrastructure [72].

2.2.2.4 NFV Management and Orchestration

Resource allocation in NFV is also a problem. When talking about resource allocation concerning the NFV architectural framework, the NFVI and NFV-MANO blocks are mainly responsible for provisioning resource allocation for VNFs, because VNFs are deployed onto the NFVI and resources are allocated through orchestration. According to [65] resource allocation in NFV is accomplished in three stages as follows:

1) VNFs Chain Composition or Service Function Chaining (SFC): SFC is the mechanism to connect VNFs in such that they form a chain of service functions [73]. This enables flexibility for CSPs to make the best use of virtualized software to define infrastructure [74]. Moreover, it enables the composition of a chain of VNFs dynamically.

TSPs can get the benefit from the composing dynamic chain of VNFs and develop elastic network services according to their business needs [75]. However, the main challenge arising while composing such a chain is how efficiently NFVI can be utilized to concatenate VNFs and control dynamicity.

2) VNFs Forwarding Graph Embedding: VNFs Forwarding Graph Embedding in NFV is a concept closely related to the Virtual Network Embedding (VNE) [76] and Virtual Data Center Embedding (VDCE) as described above [77].

A chain composed of VNFs is a connection of graphs to form an E2E service. This E2E service is known as VNFs forwarding graph embedding.

3) VNFs Scheduling: VNFs are deployed using NFVI that comprised of several high-volume servers. VNFs scheduling is the process of embedding VNFs in such a way to compose a chain of VNFs that minimize the total run time service execution. VNFs scheduling is carried out carefully without performance degradation and affecting high-volume servers in operating NFVI.

It is important to note that VNFs deployment architectures vary based on the implementation of NFVI functional blocks. Examples include VM, container-based and unikernels based deployments [78], [79]. A study carried out in [80] showed how NFV deployment practices could be optimized to achieve higher performance. Moreover, it also discusses how a carrier can fine-tune NFV deployment on standard high-volume servers by applying embedded instrumentation techniques.

All the above stages of acquiring resource allocation in NFV require efficient algorithms to determine the locations of required VNFs in high- volume servers located in the data center. This then enables the migration of servers from one location to another for efficient utilization of the NFVI. Further, this flexible placement of VNFs can offer load balancing, optimization of traffic flow, recovery from failures and possible reduction in CAPEX and OPEX [81]. Placement of VNFs is naturally challenging and particularly the different problem of how to optimize VNFs placement arises.

In order to improve the aspects of VNFs scalability dynamically for initial VNFs placement, three mechanisms are discussed as follows: i) horizontal scaling: Virtualized resources are either added or removed, ii) vertical scaling: Virtualized resources capacity or size is reconfigured, and iii) Migration: Virtualized resources are migrated to appropriate location [82], [83].

There is a need to enhance the computation efficiency of resources in NFV so that better resource allocation can be achieved in NFV environments. The management and orchestration challenge is a big concern for NFV success [84], [85], [86]. In an NFV environment, new management aspects of virtual VNFs have been introduced for creating and maintaining Lifecycle Management (LCM) of the virtualized resources for the VNFs. This includes instantiation, scaling, updating and terminating VNFs [85]. Furthermore, function placement and dynamic resource allocation must be automated and self-configurable in NFV. Currently, this is an area of intense investigation and development for NFV. Several studies have been carried out for the optimization of VNFs function placement [87], [88], [89], [90], [91]. These studies carried out multiple approaches as the optimization problem is a Non-deterministic Polynomial Time (NP), NP-Hard problem. Work done towards solving this NP-Hard problem followed heuristic and meta-heuristic algorithms to minimize the complexity in solving mixed-integer linear programming models [92], [93], [94]. To improve the computation efficiency close to the optimal is challenging.

From the NFVI management and orchestration aspect, the Network Point of Presence (N-PoP) (a location where network function is implemented as VNF or Physical Network Functions (PNFs) and NFVI Point of Presence (NFVI-PoP) (N-PoP where network function can be deployed as VNF) [95] is essential, resources such as memory and storage are accessed from N-PoP and must be handled in NFVI-PoP. This helps to chain VNFs with other VNFs or PNFs (physical appliances) to realize a network service [84].

2.2.3 Impairments to Commercial NFV Deployment

This section addresses some aspects that are obstructing the swift deployment of NFV in commercial environments.

2.2.3.1 Lack of NFV Equipment's/Products to Meet Carrier-Grade Requirements

The major problem faced today in NFV commercial deployment is the strict carrier-grade requirements [17]. The main reason for this for CSPs is to ensure a higher level of trustworthiness and service protection. Moreover, this enables CSPs to meet customer expectations for specific services and develop new business models for revenue generation. Indeed, the CGN services are incredibly reliable, tested, and compatible. This is not only concerning meeting "Five 9s" of reliability and availability but also with other requirements for CGN such as equipment management, security, billing, product development, and continuous operational support [18].

The basic research challenge is moving to all IP-based, IT-based technology, how can future NGNs fulfill the reliability, resilience and security requirements offered by traditional telecommunications networks (i.e., "Five 9s" reliability)?

The OPNFV project proposed reliability design goals for NFV, as listed in Table V. Meeting the CGN criteria of "Five 9s" reliability is essential for NFV, but it is challenging considering a virtual environment (Software-based). NFV offers benefits of agility and automation of network services, but CSPs have not yet assured that NFV appliances/products are matured for CGN services. Therefore, NFV-based products/equipment must satisfy the "Five 9s" reliability criteria for CGN. The CSPs are concerned about it because testing and validation of NFV products/equipments is still in its early stage.

2.2.3.2 Incomplete Standardization and Openness

There are several efforts (from academia and industry) for standardization and prominent NFV projects initiatives efforts to develop NFV for standardization. However, despite these efforts of academia and industry which showed continuous improvement of NFV development over the years, the NFV standardization is still underway. Indeed, a unified fabric for NFV and IT is not yet developed.

Although several examples of commercial NFV deployment exist, there is still a need to extend NFV deployments specifically to implement VNFs in multi-vendor virtual environments. The technological innovation of NFV continues to grow but standardization is considered a key factor for rolling out NFV in a large-scale production environment, and at the moment there is a need to standardize all aspects of NFV to accelerate its deployment. Another barrier to NFV deployment is openness. Openness is one of the benefits that NFV aims to offer. However, openness promoted by several vendors or industry-specific solutions or Open source NFV projects does not necessarily follow approaches to offer openness that attracts uniform acceptance from CSPs and NFV promoted communities. This issue of openness has also affected the financial picture of NFV.

2.2.3.3 Interoperability and Portability Issues

Interoperability and portability are also crucial issues holding back commercial NFV deployment. Portability refers to the support that an NFV framework aims to provide to move VNFs across a different server in multi-vendor, multi-service and multi-network virtual environment [96]. Interoperability in a virtual environment refers to the software exchange of information among different NFV vendor appliances in a multi-vendor, multi-service, and multi-network virtual environment.

To scale products and services software portability is essential for porting VNFs and VMs with ease and without vendor lock-in. Also, portability enables integration in a multi-domain networking environment. However, designing a standard interface for portability is a challenge. Once the standard for portability is designed, a standard unified approach for integration must be enabled to offer inter-operable orchestration in multi-vendor and multi-domain virtual environments [97].

2.2.3.4 Limited Business Cases for Service Providers

There has been an ongoing discussion on the business end of NFV because there is a lack of killer applications for NFV and business cases are challenging to define [98]. Moreover, most of the service providers have not yet transformed their operational support systems to support NFV [99].

The virtualization benefits of NFV were not the only benefits that CSP's are interested. There are other benefits such as how NFV can optimize networks, offer value-added service and market agility. ETSI is in a liaison relationship with Broadband Forum (BBF) is a non-profit industry consortium dedicated to developing broadband network specifications to explore the business end of NFV. BBF is optimistic about NFV application for broadband users and believes that NFV can create new revenue streams to sustain business growth in the broadband market [100].

2.2.3.5 Lack of Certified Ecosystem for NFV

There is a need to develop NFV ecosystems and offer full solutions to CSPs to adopt NFV with confidence. However, building an ecosystem is hard. Currently, ETSI NFV PoC testing and validation is lacking behind [61]. CSP's need assurance from NFV ecosystems to offer simple installation and development of customer service and also automated network management built-in capabilities with seamless integration with other legacy systems and multi-domain networking with Service Level Agreement (SLA) guaranteed.

2.2.4 In Summary

NFV has attracted significant interest from academia and the telecommunications industry as a technology that will potentially revolutionize network-based services with low deployment costs for network operators.

Virtualization approaches are paving the way for NFV, from VMs, containers to more advanced techniques such as unikernels. We discussed Type I and Type II hypervisors: the selection of a correct hypervisor is essential for the efficient utilization of underlying hardware as well as virtual environments.

NFV standardization efforts have evolved in the last few years, but the standardization is still underway. NFV applicability and PoCs are progressing and NFV is becoming a key enabler of the 5G network slicing concept.

We discussed NFV requirements, design goals and key considerations that are essential to accelerate the deployment of NFV. We provide a detailed comparison of currently selected NFV projects to lead by industry and CSPs to explore the different aspects of the ETSI NFV framework. We also discussed the NFV architectural challenges and discussed ongoing research efforts to overcome those challenges, and identified the impairments that cause a delay in NFV commercial deployment.

NFV offers several potential significant advantages over present solutions available for CSPs, but at the same time, several challenges need to be overcome soon through collaborative work on NFV to gain industry and CSPs confidence. This will play a pivotal role in propelling NFV deployments.

The NFV-based solution aims to offer service reliability comparable to the "Five 9s" reliability of PSTN. However, validation and testing to achieve QoS equivalent to PSTN is still an intense area of investigation.

NFV usage is overgrowing, and innovative ideas and practices are in progress although, its deployment is still at an early stage. The most important aspect to consider in NFV is the testing and validation of hypothetical models [101]. There are also unexplored research areas in the NFV-based proposed solution. For instance, fault management, interoperability, E2E reliability, security performance must be addressed in-depth. Furthermore, network automation is essential for improving NFV resilience.

NFV is a still evolving paradigm for programmable networks [102]. Softwarization of the telecommunications systems can provide a long-term solution to the gradual network ossification problem faced by the existing internet and NFV is an essential element directed towards a solution to this problem. Furthermore, in Section 6.3.1, we concluded by enumerating future research directions for NFV development.

2.3 Fault-Tolerance in the Scope of SDN

Due to the lack of software programmability in today's networks, it is quite challenging to modify (program) networks. Traditionally, there was no underlying programming abstraction provided to deal with the inherent complexity of distributed system failures. One of the primary features that SDN provides is data and control plane separation, laying the ground for simple network programmability. Although there is an extensive set of SDN research, most of the research performed so far focuses on exploring SDN as a programmatical technology, without considering fault-tolerance aspects [103], [104], [105], [106].

Fault-tolerance is a broad area of knowledge, and covering all aspects of fault-tolerance concepts in a single chapter is difficult. Hence, in this section, we briefly discuss key fault-tolerance concepts and focus more on fault-tolerance in the scope of SDN. It is important to note that fault-tolerance and fault-management concepts are different. On the one hand, fault-tolerance is a characteristic of a system, which is designed in such a way that it can minimize service failures in the presence of system components faults. On the other hand "Fault management" is a term used in network management, describing the overall processes and infrastructure associated with detecting, diagnosing, and fixing faults, and returning to normal operations in telecommunication systems [107].

Generally, fault-tolerance is an essential part of the design of any communication system/network. Computer networks are built on physical infrastructure or virtualized versions of the physical infrastructure. These infrastructures are critical because business applications rely on the proper operation of such infrastructures. However, such infrastructures are prone to a wide range of challenges and attacks such as natural disasters or Denial of Service (DoS) attacks and major issues such as faults, failures, and errors all of which cause failure and disruption in network service. Therefore, to overcome these network service issues, resilience procedures and fault-tolerance mechanisms are essential to identify and heal the system/network in the presence of such failures [108].

SDN provides network flexibility through a clear separation of control and data planes, inherently simplifying network management [109], although SDN fault-tolerance is still in its infancy. SDN is exposed to new sets of failures and issues at each layer of its architecture, as discussed in Section 2.3.3. It is necessary to address these issues and safeguard each layer of the SDN architecture to provide enhanced fault-tolerance. We overview fault-tolerance techniques and typical phases of fault-tolerance. We then highlight fault-tolerance issues according to the SDN's three main layers (data, control, and application) and classify SDN fault-tolerance research according to these three layers.

2.3.1 Background and Related Concepts

In this section, we discuss fault-tolerance and its techniques and provide a brief overview of fault-tolerance in traditional networks and its relationship with dependability.

2.3.1.1 Fault-Tolerance Overview

Any system is prone to some sort of threats that affect the operation of the system. Moreover, in computer networking, both distributed and centralized network systems are also prone to three major issues: Failures, Errors, and Faults.

A failure happens if a system is unable to implement the specified function appropriately. This means the service deviates from its specifications. An error is caused because one or more of the sequences of system states deviate from the specified sequence, and can cause service disruption. Faults can cause errors and lead to single or multiple failures [110]. A fault is the hypothesized cause of an error, for instance, a software bug, human-made error or hardware power failure [111]. Relationship between fault, error, and failure is depicted in Fig. 2.6 [112].

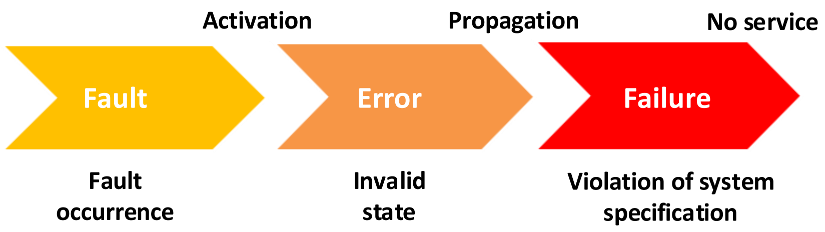


Figure 2.6: Relationship:Fault, Error, and Failure

Fault-tolerance is the outcome of a design process of building a reliable system from unreliable components [113]. Faults can be classified into two main categories [114], [115]: Crash faults and Byzantine faults. Crash faults can cause system fatal errors (for instance process and machine power-related failures), while Byzantine faults can cause the system to deviate from normal operation [114]. Fault-tolerance systems are equipped with several mechanisms that not only respond to these issues but also continuously offer and maintain correct system operation. However, it is hard to design in practice a fault-tolerance system that can guarantee flawless communication but even in worst-case scenarios, fault-tolerance systems usually still offer graceful degradation of services. Nevertheless, we can always design efficient mechanisms for faults and errors that are most likely to happen and affect any system. Such approach can improve and enhance fault-tolerance communication systems.

2.3.1.2 Fault-Tolerance Phases

The typical four main stages of a fault-tolerance are as follows [116]:

1. **Error Detection:** In this stage, faults are first detected and then reported to determine the root cause of failure (observing failures).
2. **Damage Confinement and Assessment:** In this stage, the damaged or corrupted state of the system is assessed to determine the extent of the damage caused by faulty components.

3. **Error Recovery:** In this stage, recovery strategies are imposed to restore the system to a consistent and fault-free state. There are two different kinds of recovery techniques used:
   - Backward Recovery: In this technique, system states are recorded and stored so that a corrupted state can be discarded and the system can be restored to the previous fault free (correct) state.
   - Forward Recovery: In this technique, the system is being brought to a new correct fault-free state from the current corrupted state.
4. **Fault Treatment and Service Continuation:** In this stage, the location of faults are identified first and then faults are either repaired or the system is reconfigured to avoid faults. Service continuation is essential to ensure that the system will perform its operation normally and without immediate manifestation of faults.

   The typical phases of implementing fault-tolerance is shown in Fig. 2.7.

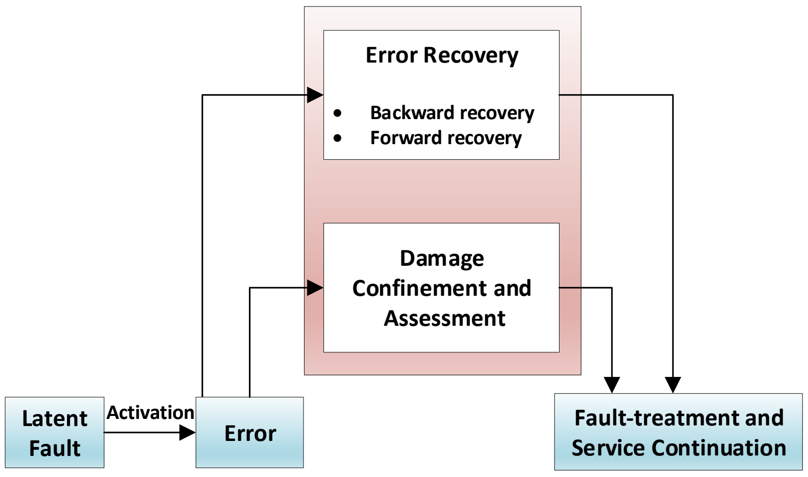


Figure 2.7: Typical Phases in Fault-tolerance

2.3.1.3 Fault-Tolerance Techniques

Several fault-tolerance techniques are being used to avoid service failure in the presence of faults [117]. Fault-tolerance is carried out through error detection and system recovery, or simply detection and recovery mechanisms. Error detection identifies the presence of an error, while "recovery transforms a system state that contains one or more errors and (possibly) faults into a state without detected errors and faults that can be activated again " [118]. Recovery techniques can be further classified into two main categories: i) recovery with error handling; which eliminates errors from the system state; and ii) recovery with fault-handling; which prevents faults from being activated again. The choice of error detection and recovery techniques are being adopted based upon the underlying fault assumption. In the context of SDN, these fault-tolerance techniques must be explored in order to enhance fault-tolerance in future SDN environments.

Table 2.5: Details of Fault-tolerance Techniques

Error Detection	**Recovery (Error Handling)**	**Recovery (Fault Handling)**
Concurrent Detection: Occur during normal service delivery	Rollback: Restores system state to a saved last good known configurations before error occurrence	Diagnosis: Identify both error localization and its types
Preemptive Detection: Occur while normal delivery service is suspended; to check latent errors and dormant faults	Rollforward: Initiate a new system state without detected errors	Isolation: prevent the participation of faulty components that can leads to service failure
-	Compensation: Recover system erroneous state by enabling error to be masked through redundancy	Reconfiguration: Reassigns tasks among non-failed components
-	-	Reinitialization: Check and update system based on new configurations

The taxonomy of fault-tolerance techniques can be seen in Fig. 2.8, Table 2.5 [111] summarizes the details of such fault-tolerance techniques.

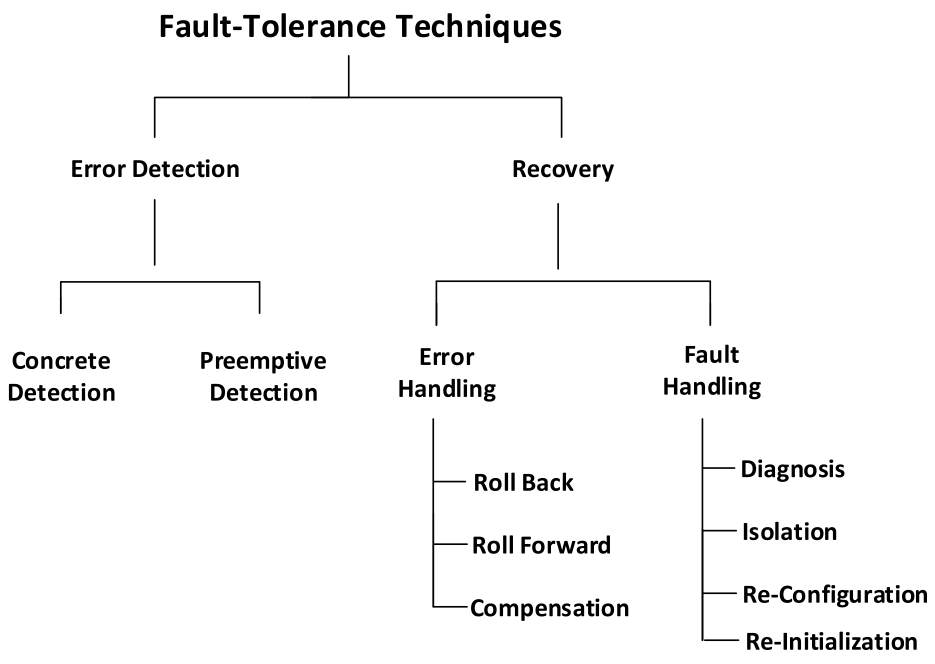


Figure 2.8: Fault-tolerance Techniques

2.3.2 Fault-Tolerance in SDN

In this section, we provide an overview of SDN architecture and discuss SDN fault-tolerance based on the SDN architecture as divided into three main layers: data plane, control plane, and application plane.

2.3.2.1 SDN Architecture Overview

SDN is a hot research topic, but there is increasing confusion regarding SDN concepts; architecture, multiple SDN networking planes, and interaction between layers through interfaces. Briefly, we discuss the SDN architecture and discuss the abstracted view of SDN planes. The SDN architecture is shown in Fig. 2.9, which comprises several abstraction layers (abstraction of well-defined planes), interfaces (Standardized APIs between planes) and well-defined planes (collection of functions and resources with the same functionality) [119].

The three distinct SDN planes are as follows:

1. **Data Plane:** The data plane (also known as the forwarding plane) is responsible for handling data packets sent by the end-user through network devices that are responsible for traffic forwarding (based on instructions received from the control plane).

   The Forwarding Information Base (FIB), also known as forwarding table and Medium Access Control (MAC) table for routers and switches. FIB is used in

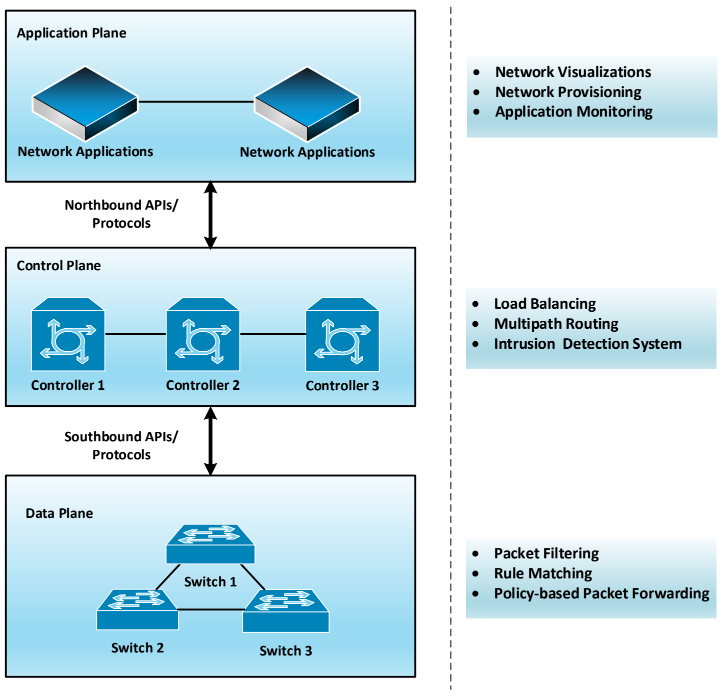


Figure 2.9: SDN High-Level Architecture: SDN Planes and Communication Interfaces

the data plane to perform IP forwarding of unlabeled packets [120].

2. **Control Plane:** The control plane is responsible for deciding on how packets must be handled and forwarded at network devices to properly cross the network. The primary purpose of the control plane is to synchronize and update forwarding tables, while packet handling policies reside in the forwarding plane.
3. **Application Plane:** The plane where applications and services that define network behavior reside. Applications that directly (or primarily) support the operation of the forwarding plane (such as routing processes within the control plane) are not considered part of the application plane.

2.3.2.2 Controller

Networks in SDN are managed by an external controller to process the flow of packets. In SDN architectures, the controller is a logically centralized entity. It is responsible for translating the SDN applications requirement, via a Northbound API, down to the SDN data layer. Furthermore, it is also responsible for providing SDN applications an abstracted view of the network (including statistics and events).

The controller enables the programming of the network to be centrally managed. Hence, the entire network and its devices can be managed efficiently regardless of the complexity of the underlying infrastructure. Moreover, SDN offers the flexibility through programming to separate the data and control planes with the logically centralized controller and by this, it is possible to modify the packet forwarding as per the network needs [103].

It is important to understand that different multi-controller architectures can exist with SDN. Bilal et al. [121] describe different types of SDN architecture and existing implementations. They further classify multi-controller SDN architecture in two broad categories (logically centralized and logically distributed architecture) and discuss in detail with an example of implementation of such designs.

SDN controller fault-tolerance issues exist and are addressed in current research, but are still far from providing the optimal solutions. Later in Section 2.3.4, we discuss the SDN controller (centralized and distributed architecture) fault-tolerance research efforts.

2.3.2.3 Southbound and Northbound APIs

All the SDN networking planes are connected through specific interfaces, that standardize and simplify intercommunication between them. Intercommunication between SDN networking planes can be achieved in two different ways depending on the SDN architecture design, and the location of network entities. On one hand, if they are placed in a different location, a network protocol is used to provide communication interaction between them. On the other hand, if the network entities reside inside the same physical or virtual location, a communication interaction between network entities is possible with APIs. This enables the flexibility to design and implement intercommunication between network entities either through network protocols and/or APIs.

The SDN architecture has two primary interfaces (which use either APIs and/or protocols), as depicted in Fig. 2.9 [122], to enable intercommunication between two different SDN Planes: the Southbound and Northbound. In SDN terminology they are often referred to as Southbound APIs and Northbound APIs.

The Southbound API is a communication interface between the data and control plane. Currently, OpenFlow is a default standard for this communication. The OpenFlow standard has been proposed to manage the communication flow between the controller and network entities [123]. Furthermore, in SDN, OpenFlow is not the only available protocol for Southbound interface [124]. Other protocols and/or APIs for Southbound interface are: Forwarding and Control Element Separation (ForCES) [125], Network Configuration Protocol (NETCONF) [126], and Extensible Messaging and Presence Protocol (XMPP) [127], but they are more rarely used.

The Northbound API is a communication interface between the control plane and the application plane. Currently, there is no standardized northbound API. Because of this, the development of network applications for SDN has not been as swift as desirable [128]. Nevertheless, most implementations use REepresentational State Transfer (REST)-based API because it is platform and language independent [14].

2.3.2.4 OpenFlow Fault-Tolerance Support in SDN

This section discusses OpenFlow fault-tolerance from the point of view of the requirement for CGNs. CGNs usually provide faster recovery against service failure (i.e recover within 50ms) [129]. If service is not able to recover within this time, then service providers may be jeopardizing their business.

OpenFlow was developed to support communication with non-proprietary FIBs. The OpenFlow protocol provides an abstraction of FIB through the OpenFlow group table concept. Moreover, the OpenFlow protocol communicates with the controller, which can trigger modifications in packets forwarding rules. This makes the FIB programmable through OpenFlow.

In SDN networks, operations rely on the proper functioning of the controller. The control plane in SDN manages the control logic of switches. The control logic is critical in SDN-based networks. This problem is minimized in the latest version of the OpenFlow protocol by a master-slave configuration at the control layer, to increase resiliency. However, we argue that a tight synchronization must be required between a master and slave configuration to maintain an identical state of this configuration or the same copy of the master controller and this causes extra overhead in networks and sophisticated network management demands. It is quite challenging, to reach the recovery time equivalent to the standards set by the CGNs, therefore, in order to enhance OpenFlow fault-tolerance support, mechanisms that not only maintain controller persistent state but also provide efficient recovery in case of controller fail-over must be developed. Another research challenge is the optimization of recovery time as per the carrier-grade requirement as well as scalability. Indeed, CGNs are a network of networks and scalability is a critical aspect. This excludes any solution not scalable, as such a solution is not of interest for such carriers.

2.3.2.5 SDN Data Plane Fault-Tolerance Support

SDN data plane fault-tolerance is related to the issues already present in traditional architectures (e.g. multiprotocol label switching technology). Due to the static nature of traditional networks, these approaches can achieve good performance upon link and node failures. However, failure detection and recovery approaches in dynamic networks such as SDN must be re-designed to adapt to the dynamics of the rapidly changing networks. Traditionally, reactive and proactive approaches were used to provide Fault-tolerance [130]. In the reactive approach, an alternative path is calculated after the fault becomes active. In proactive techniques, the resources and backup paths are pre-programmed before the occurrence of a fault (when a fault is dormant). If the fault becomes active, the pre-programmed logic starts to defend immediately and recover the system from faults. In this section, we address such failure detection and recovery approaches.

2.3.2.6 Failure Detection Approaches

The high availability of the data plane plays an important role to maintain the required communication from source to destination. To achieve high resiliency in the data plane, two steps are required: first, design and analyze the topology in the presence of known and unknown failures; and second, to design an alternative path according to the type of failures that occur in the network. In CGNs, two well-known mechanisms exist to detect failures in the data plane, namely Loss of Signal (LOS) and Bidirectional Forwarding Detection (BFD) [131]. LOS detect failures in a specific port of the forwarding device, while BFD can detect path failure between any two forwarding devices. Both methods provide failure detection at an accelerated rate, independent of the media type and routing protocols (such as Open Shortest Path First (OSPF) and Enhanced Interior Gateway Routing Protocol (EIGRP)).

2.3.2.7 Failure Recovery Approaches

In CGNs the recovery mechanism must guarantee the recovery process within 50 ms [132]. For this purpose, restoration and protection are widely used to recover from network service failures — methods based on reactive and proactive approaches. Protection is classified as a reactive technology while restoration is classified as pro-active technology. In restoration, an alternative path is only established after the occurrence of failure and resources are not reserved before the occurrence of the failure, and the paths are pre-assigned or allocated dynamically. However, in the case of protection, the alternative paths are already reserved and assigned before the occurrence of a failure. This needed no added processing (signaling) to recover from failure. In restoration, additional signaling is needed to recover from failure; in large networks, this is not often possible within the set requirement of CGNs, and thus it is not scalable. However, in protection, as a matter of fact, the additional signaling is not required, and recovery process is fast when compared to restoration, with the recovery process possible within 50 ms and suitable for CGNs.

2.3.2.8 SDN Control Plane Fault-Tolerance Support

Control plane resilience is a requirement for proper operation in networks: the controller is vital, which means that the controller must be able to process all required traffic commands in all situations. There are several approaches to enhance SDN control plane fault-tolerance. The first approach is to replicate a controller on a different control network. In the case of failure, the replicated controller takes over and manages traffic. In another approach, the controller must be embedded with mechanisms (build-in module) to self-heal from targeted attacks such as DoS, flooding and fake traffic routing and other network-related targeted attacks. However, the control plane time to recover from such attacks is critical, and ideally, recovery mechanisms must be developed to mitigate failures within the set network requirements. In addition to these, the recovery process must be efficient and must be able to self-heal during a failure event with minimum overhead. In-band and out-of-band signaling solutions have been adopted to offer SDN control plane reliability [133]. In practice, most SDN deployments use out-of-band control, where control packets are managed by a dedicated management network [134].

2.3.2.9 SDN Application Plane Fault-Tolerance Support

On an SDN network, the Application plane is the layer that has applications and services that make requests for network functions provided by the control plane and the data plane. On traditional networks, security, management, and monitoring devices or applications reside in this layer.

The application layer allows business applications to modify and influence the way the network behaves in order to provide services to customers. This requires the definition of an API, to allow third-party developers to build and sell network applications to the network operator. The development of such an API has not yet been properly addressed by the Open Network Foundation (ONF) but is required in order to guarantee interoperability between a business application and network controllers from different suppliers.

Existing SDN programming languages offer several features such as flow installation, policy definition, programming paradigm and abstraction for developing and enabling network and application fault-tolerance in SDN [135], [136], [137], [138].

2.3.3 SDN Architecture Fault-Tolerance Issues

In this section, first, we highlighted SDN fault-tolerance issues and then provide state-of-the-art research efforts focusing on such fault-tolerance issues in SDN. Furthermore, we structure them based on the three main layers of SDN architecture. These are later summarized in Table 2.6, Table 2.7, and Table 2.8.

2.3.3.1 Data Plane Issues

There are two main data plane layer issues namely: network failure detection and recovery. These issues arise either due to link or node failure. As discussed, in traditional networks, to detect network failure, a particular protocol, such as LOS and BFD, is used [139]. Also, to recover from network failure, restoration and protection approaches are widely used. However, resolving these issues in the SDN environment is challenging due to the centralized nature of the controller. For instance, in the SDN-based environment, the controller can take a longer time to detect and recover from link or node failure due to the rapidly changing abstracted view of the network (dynamic topology). Therefore, there is a need to develop mechanisms for SDN that can provide faster recovery [139].

2.3.3.2 Control Plane Issues

There are multiple SDN control plane issues. The four main issues that are critical to SDN control plane fault-tolerance can be classified as:

1. **Controller Channel Reliability:** In SDN controllers, communications with underlying devices are critical. Therefore, their availability is a must condition to protect the proper operations of a network. The controller channel must be fault-tolerant (reliable) in case of failure due to loss of switches connection, or error due to the communications protocols between the controller and underlying devices. These issues can disrupt the network and lead to several failures in the SDN network. In order to cope with these issues, controller redundancy [140], [141] and path backup are considered essential.
2. **Controller Placement and Assignment:** Controller placement (how to choose the location of controllers) and assignment (how to assign the controllers to the switches) are two significant issues [142], generally known as the controller placement problem [143].

   The controller's assignment issue (balance of controllers) in SDN is important, not only from the point of view of fault-tolerant controller design but also from the point of view of network optimization. Improper controller assignment can lead to two main problems: i) under-provisioning: When a small number of controllers are placed to handle more traffic than its capacity of processing. In this case, the controller is overloaded and possibly increases downtime and affects network performance, and ii) Over-provisioning: When more than the required

controllers are placed to handle a comparably low traffic environment. In this case, costly controllers are underutilized.
To deal with the controller placement issue, one of the strategies is to develop algorithms that can provide optimal controller placement in dynamic SDN-based networks, which is also challenging in itself [144].

3. **Inter-controller Consistency:** In order to avoid a single point of failure in SDN networks, multi-controllers architecture approaches are pursued, either in physically centralized and logically distributed, or fully distributed fashion with the coordination of different SDN controllers [145]. It is important to note that these practices increase resiliency, but there is a strict requirement for controllers consistency [146]. The level of consistency depends on stateful or stateless backup settings. The controller must maintain a persistence state to guarantee controller consistency.
4. **Multi-Controller Architecture Fail-over:** In SDN, multi-controller architectures can follow the flat/horizontal or hierarchical/vertical designs. On the one hand, in a flat architecture, the control plane has just one layer, and each controller has the same responsibilities [121]. The advantage of such architecture design is that it provides more resilience against failure, but the task of managing controllers is difficult. On the other hand, in a hierarchical architecture, the control plane has multiple layers, and each controller has different responsibilities (due to multiple level partitioning). The advantage of such design is that it provides a more straightforward way to manage controllers. Both of these multi-controller architecture approaches can be used to improve switch to controller latency or vice versa. In both designs, it is important to consider that controllers must respond to any fail-over [147] request efficiently and without affecting the performance.

2.3.3.3 Application Plane Issues

SDN enables programmability to control network devices more efficiently but this is highly dependent on the quality of software development. In order to develop reliable SDN applications, debugging (the process followed to fix bugs) and testing (verification) tools can not only advance software quality but also help in fixing software bugs as service evolves (continuous development process) [148]. To ensure the quality of software network troubleshooting, debugging and testing are considered essential [136].

Network visualization, network provisioning, and application monitoring can be conceptualized as an SDN application layer. For this reason, fault-tolerance of both network and applications can be supported at the application plane. Moreover, in order to develop fault-tolerant network applications, all the phases from application design to final application deployments must undergo proper testing. Currently, there are certain languages proposed [135] that enable the construction of fault-tolerant programs to write SDN-based fault-tolerant systems. Since fault-tolerance of both network and applications can be supported at the application plane, the two main SDN application layer issues are as follows:

1. **Software Testing:** The network behavior in a SDN-based network is controlled by a set of software programs. For proper network troubleshooting support, the SDN applications must be resilient [149]. Resilient design help to identify the root cause of the bug and the administrator can then track and isolate faults so that the system can be restored to the correct operating state.
2. **Policies Configuration:** In SDN, network management becomes more dependent on software development due to programmability. There is a risk that policies across the network can be violated due to untested errors (bugs) in the application, which can be propagated to affect SDN controllers, protocols and routing policies and eventually can lead to network service failure. Therefore, constant application monitoring is essential to avoid any violation of network policies [150].

2.3.4 SDN Fault-Tolerance Research Efforts

SDN offers greater flexibility and network automation when compared with traditional distributed systems, at the risk of the controller being a single point of failure.

Most of the research carried out has been focused on exploring these technologies rather than evaluating the associated reliability aspects. In recent years, a shift is being made towards the evaluation of SDN fault-tolerance. We reviewed the research studies carried out that address SDN data, control, and application planes fault-tolerance. The details are summarized in Table 2.6, Table 2.7, and Table 2.8. Furthermore, classification of SDN state-of-the-art research efforts according to SDN planes and controller architecture is depicted in Fig. 2.10.

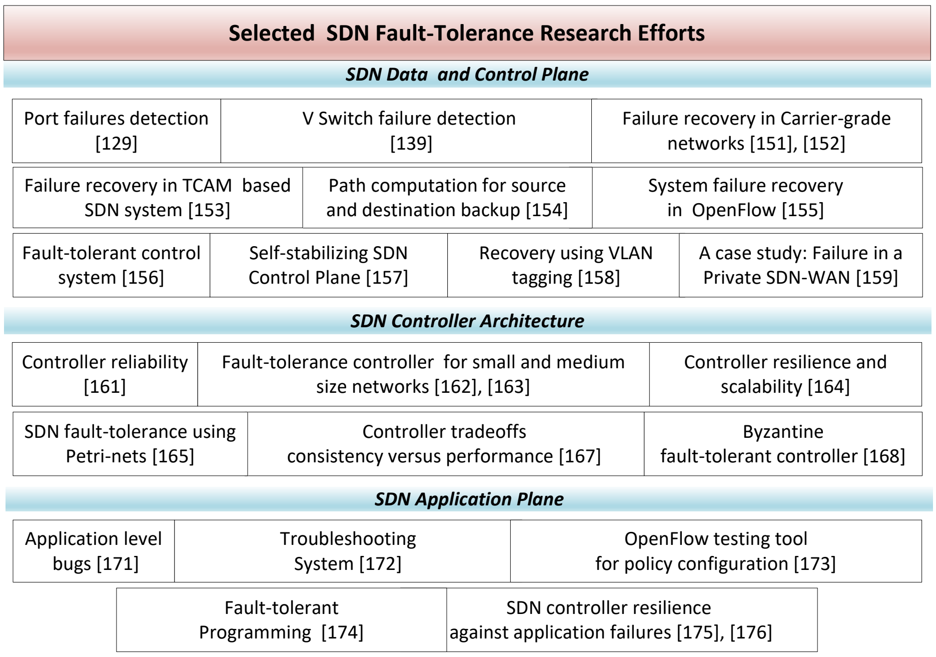

Selected SDN Fault-Tolerance Research Efforts
SDN Data and Control Plane
Port failures detection [129]
V Switch failure detection [139]
Failure recovery in Carrier-grade networks [151], [152]
Failure recovery in TCAM based SDN system [153]
Path computation for source and destination backup [154]
System failure recovery in OpenFlow [155]
Fault-tolerant control system [156]
Self-stabilizing SDN Control Plane [157]
Recovery using VLAN tagging [158]
A case study: Failure in a Private SDN-WAN [159]
SDN Controller Architecture
Controller reliability [161]
Fault-tolerance controller for small and medium size networks [162], [163]
Controller resilience and scalability [164]
SDN fault-tolerance using Petri-nets [165]
Controller tradeoffs consistency versus performance [167]
Byzantine fault-tolerant controller [168]
SDN Application Plane
Application level bugs [171]
Troubleshooting System [172]
OpenFlow testing tool for policy configuration [173]
Fault-tolerant Programming [174]
SDN controller resilience against application failures [175], [176]

Figure 2.10: SDN state-of-the-art Research Efforts: Simplified Taxonomy

2.3.4.1 Data and Control Planes

In this section, we discuss main research efforts that have addressed data and control plane fault-tolerance in the context of SDN.

Current fault-tolerance techniques are not yet proven to meet carrier-grade fault-tolerance requirement (50 ms recovery time) [129]. A research study carried out by Sharma et al. [132] provided experimental evidence that protection provides faster recovery as compared to restoration and is thus more suitable to guarantee resilience in large scalable networks [151], [152].

Adrichem et al. [139] argued that time was a critical metric in the recovery process during network failures. It is still difficult to develop mechanisms that guarantee efficient recovery. In their research study, they demonstrated that current failure recovery approaches (restoration and protection) suffered from long delays. They introduced a failover scheme based on a per-link BFD approach and showed that implementation reduced recovery time. They performed experiments to evaluate different network topologies and showed that recovery time was consistent irrespective of network size.

Mohan et al. [153] carried out a research study to provide fault-tolerance in the specific case of Ternary Content Addressable Memory (TCAM) limited SDN. They argue that proactive fault-tolerance policies provide faster failure recovery based on restoring the re-routing paths. This requires large forwarding rules to be installed on the TCAM, but TCAM has limited memory. Based on these challenges, they have developed an optimized programming formulation that determines the set of backup

Table 2.6: Selected Works on SDN Data and Control Plane Fault-tolerance Efforts

Fault-Tolerance Efforts by	**SDN Plane(s)**	**Main Purpose**
Sharma et al. [129]	Data plane	Check device port failures and consistently monitors failure detection between two forwarding links to support standardized failure detection methods such as BFD and LOS.
Adrichem et al. [139]	Data plane	Improve the speed of failure detection in Open vSwitch based implementation.
Sharma et al. [151], Sharma et al. [152]	Control plane	Meet failure recovery requirements for CGNs.
Mohan et al. [153]	Control plane	Improve TCAM and bandwidth efficiency for single link failure in SDN system by reducing number of flow rules.
Li et al. [154]	Control plane	Compute different backup according to source and destination pairs.
Kuźniar et al. [155]	Control plane	Recover system failures in OpenFlow based controller implementations.
Kim et al. [156]	Control plane	Develop a fault-tolerant system, able to recover from multiple link failures in the data plane.
Schiff et al. [157]	Control plane	Presented a model to design self-stabilizing distributed SDN control planes.
Chen et al. [158]	Data and Control plane	Enable faster recovery with low memory using VLAN tagging concept.
Jain et al. [159]	Data and Control plane	Evaluate outage and failure in a private SDN WAN.

paths to protect a flow and minimize the number of forwarding rules for the backup paths to be installed in the switch TCAM. This means that fewer rules would be required for backup paths. They proposed two algorithms, Backward Local Rerouting (BLR) and Forward Local Rerouting (FLR) [153], to improve TCAM and bandwidth usage efficiency for single link failure in SDN system.

Li et al. [154] carried research studies to enhance failure recovery in SDN with customized control. They have developed a Declarative Failure Recovery System (DFRS) based on three algorithms: backup path construction, add and subtract. The Backup paths construction algorithm creates safe backup paths based on the recovery demands. Further, it adds and subtracts the outcome of algorithms to find a minimum number of the paths to be allocated to guarantee network services during failure with minimum memory overhead [154]. Three different topologies were evaluated to test the effectiveness and scalability of DFRS. They achieved similar performance to the traditional failure protection algorithm, but with 5% less or 5% of backup rules. In the event of failure, many switches are allocated hundreds of forwarding rules for backup; this burdens the switch, affects the performance and delays failure recovery. The authors argue that the DFRS system only allocates dozens of forwarding rules to switches, as compared to the usual hundreds of forwarding rules. Thus, this leads to effective memory utilization and improved stability.

Kuźniar et al. [155] proposed Automatic Failure Recovery (AFRO) for SDN, an automated runtime system that recovers system failure in OpenFlow system. They argue that they extend the basic functionality of the controller program with additional controller agnostic modules that provide efficient recovery.

Kim et al. [156] proposed CORONET (the SDN fault-tolerant system), and they argue that their proposed system provides recovery against multipath failure in a data plane. However, since the initial published work in 2012, no significant contribution was made, although many evolutions were represented on SDN architectures and protocols.

Schiff et al. [157] presented a model to design self-stabilizing distributed control planes for SDN and argue that their proposed technique provides a mechanism to deal with key challenges of a distributed system, such as bootstrapping and in-band control. Further, they implemented a plug and play SDN distributed control plane to support automatic topology discovery and management in dynamic networks. However, it is important to note that the self-stabilizing distributed plane is still at a very early stage, and a lot more effort is needed to step forward to a proof of concept stage. The authors also claim that a feasibility study is needed to further validate their proposed model of the self-stabilizing SDN control plane. Formal proofs are required for this plug and play distributed model to be shown effective in meeting fault-tolerance requirements for SDN and future networks.

Chen et al. [158] proposed a method of protection-based recovery in SDN using VLAN tags. They argue that their proposed method provides faster recovery with low memory usage and without the participation of the controller to switch to backup paths. In their system, protection takes 20 ms while recovery on average takes 50 ms to restore from failures. Similarly, Thorat et. al. [160] proposed a proactive policy to achieve fast failure recovery using VLAN tags and claims that 99 % reduction in flow storage is achieved as well as fast failure recovery as set in CGNs.

Jain et al. [159] carried out a study to address network outages and failures. They evaluated three years of production experience with B4, their own SDN enabled WAN) that connects Google's data center. They implemented fault-tolerance policies such as customized forwarding and dynamic relocation of bandwidth and alternative link recovery using OpenFlow. Generally, control plane protection is achieved through resource replication and replicas were placed on different physical servers. They have analyzed in their study that SDN enabled WAN served more traffic than public WAN and offered cost-effective bandwidth and nearly 100 % link utilization, enabling high availability of resources. However, they admit that bottlenecks in the bridging protocol from the control plane to the data plane exist and need to be optimized to improve performance further. Improving this will offer superior fault-tolerance in future SDN based networks.

2.3.4.2 Controller Architecture

In this section, we discuss key research efforts that have addressed controller architecture fault-tolerance in the context of SDN.

Katta et al. [161] studied the fault-tolerance of the controller under crash failures. They argued that to offer a logically centralized controller, it is necessary to maintain a consistent controller state and ensure switch states consistently during controller failure. Therefore, they have proposed Ravana, an SDN based fault-tolerant protocol that provides an abstraction of the logically centralized controller. Ravana handles the entire event processing cycle and ensures total event ordering across the entire system. This enables Ravana to correctly handle switch states and replicas without the need of restoring to rollbacks. Moreover, it mitigates control messages during controller failures this helps in extending the control channel interface. Ravana provides a reliable distributed control for SDN. However, it does not provide support for richer fault models such as Byzantine failures, and it is limited to multithreaded control applications, and the scalability is also one of the tests that are not evaluated in Ravana protocol.

Botelho et al. [162], [163] carried out research studies and implemented a prototype that integrates a Floodlight-based distributed controller architecture to BFT-SMaRT (Byzantine Fault-tolerant (BFT) and State Machine Replication (SMR)), a replicated state machine library . This enables the consistency between an SDN-controller and their redundant backups stored in a shared database. In their work, three SDN applications (learning switch, load balancer, and device manager), with slight modifications, were tested to analyze the workloads these applications were generating and measure the performance. The result of their study shows that the data store is capable of handling large workloads, but to maintain a strong consistency of data there was an increase in latency and this impacted performance. Thus, the solution seems not to be scalable, they argue that an acceptable level of fault-tolerance was easy to achieve. Moreover, the authors also proposed a practical fault-tolerant SDN controller design for small and medium networks. A shared database is replicated that save all network state. This database is created using a replicated state machine, and in their previous research studies, they argue that the database meets the performance requirements for small and medium networks. They incorporate a cache in the controller that avoids failure smoothly without any additional coordination service.

Table 2.7: Selected Works on SDN Controller Fault-tolerance Efforts

Fault-Tolerance Efforts by	**Controller Architecture**	**Main Purpose**
Katta et al. [161]	Centralized	Develop reliable distributed Control plane for SDN controller.
Botelho et al. [162],[163]	Centralized	Develop fault-tolerant controller for small and medium size networks.
Fonseca et al. [164]	Centralized	Enable Control plane resilience and scalability.
Aly et al. [165]	Centralized	Petri-net based mathematical framework to enhance SDN fault-tolerance.
Tootoonchian et al. [166]	Distributed	Enable Control plane resilience and scalability.
Gonzalez et al. [167]	Distributed	Evaluate trade-off between consistency and performance in a fault-tolerant SDN platform.
ElDefrawy and Kaczmarek [168]	Distributed	Develop SDN controller that can tolerates Byzantine faults.

A master and slave controller configuration is implemented by Fonseca et al. [164] in which the solution to offer control plan resilience is provided by integrating a Control Plan Recovery (CPR) module into a standard OpenFlow controller build upon NOX OpenFlow controller. CPR is a two-phase process, consisting of replication and recovery, and offers resilience against several types of failure in an SDN enabled centrally controlled networks. Similarly, the research studies carried out by Tootonchain et al. [166] introduce HyperFlow to provide control plane resilience. HyperFlow is a distributed event-based control plane, which is physically distributed but logically centralized. This enables scalability, as well as ensures the benefits of centralized network control. They argue that HyperFlow [169] offers a scalable solution for control plane resilience in SDN enabled networks.

Aly et al. [165] used SDN-centralized architecture in which a master controller is connected to a set of slave-controllers. Based on this setup, they proposed a new Petri-net based mathematical framework for SDN fault-tolerance and named the model FTPN_{SDN}. They claim that, in order to avoid service disruption, Petri-net capability functions were used to identify the next backup controller in the event of controller failure. They also showed that the transition time needed to take over another controller was reduced by 10%. They evaluated the performance of their proposed model, comparing it with the HyperFlow reference model, and claim that they were able to reduce the 12% packet delay.

Clearly, a single controller point of failure limits scalability and we argue that the several recent research studies carried out do not yet provide a mechanism to achieve high-level performance or fault-tolerance at scale in SDN-based networks.

To deal with the challenge of SDN-controller consistency and performance, Gonzalez et al. [167] proposed a method to improve consistency and performance by using some of the approaches from the recent study carried out by Katta et al [161]. They design a mechanism to provide better consistency and performance in master-slave SDN configuration. They consider the performance metric for the SDN controller based on controller latency and throughput. Their proposed solution provides consistency and performance close to the offered by a single SDN controller. However, they emphasize that a very reliable communication channel is a must between the master controller and the data store.

ElDefrawy and Kaczmarek [168] proposed a fault-tolerance SDN controller design that tolerates Byzantine faults. However, their controller design has not yet achieved high-level performance for large-scale deployments. Further, they argued that their controller design is feasible for constructing resilient networks. In this research study, they have designed and prototyped a Byzantine-fault-tolerant distributed SDN controller to tolerate malicious faults both in control and in data plane as described in Kreutz et al. [104]. Further, they integrated the two existing SDN byzantine vulnerable controller with the BFT-SMaRt, a tool for creating byzantine fault-tolerant system [170].

2.3.4.3 Application Plane

In this section, we discuss key research efforts that have addressed application plane fault-tolerance in the context of SDN.

SDN offers the flexibility of network programmability but this raises issues of software-based troubleshooting and debugging which need to be addressed, as discussed.

Heller et al. [171] proposed a structured troubleshooting approach by exploiting

Table 2.8: Selected Works on SDN Application Plane Fault-tolerance Efforts

Fault-Tolerance Efforts by	**SDN Plane(s)**	**Main Purpose**
Heller et al. [171]	Application plane	Develop a tool to identify bugs based on the root cause of actual bugs.
Scott et al. [172]	Application plane	Develop troubleshooting system and framework for SDN.
Canini et. al. [173]	Application plane	Develop a testing tool for Open-Flow based SDN to detects any violation of network correctness policies.
Reitblatt et al. [174]	Application plane	Develop high level language to write fault-tolerant programs to implement network policies.
Chandrasekaran and Benson[175], Chandrasekaranet et al. [176]	Application plane	Make SDN controller and network resilient to SDN application failures.

the SDN layered architecture. They aim to develop a tool that would identify bugs by systematically tracking the root cause of detected failures. This would save time in diagnosing and enable the network administrator to directly fix the problems. However, they have not proposed any system or framework. In a similar way, Scott et al. [172] also studied SDN troubleshooting and proposed the SDN Troubleshooting System (STS). This system aims to optimize the debugging time by filtering events not correlated to the source of failure. They have demonstrated the feasibility of their proposed system and have tested five SDN control Open source platforms: ONOS (Java) [177], POX (Python) [178], NOX (C++) [179], Pyretic (Python) [180], and Floodlight (Java) [181]. They were able to identify seven new bugs in real-time, and debugged them using their proposed STS system, and showed that STS enhances the time-consuming process for debugging in SDN. Likewise, Canini et al. [173] built NICE, a troubleshooting tool for SDN. The state-space of the entire SDN system is explored through model checking. This approach provides a systematic way to test unmodified controller programs . This tool automates the testing of OpenFlow application based on model checking and concocts execution efficiently.

Reitblatt et al. [174] proposed FatTire, a high-level declarative language for writing fault-tolerant network programs in SDN. This high-level language aims to provide policy-based network management where SDN programmers can construct specific policies (for instance, data security and customized forwarding). Earlier work of Lui et al. [182] emphasizes that connectivity must be realized as a data plane service. This work fits together with FatTire for implementing policy abstractions. Similarly, a study by Suchara et al. [183] based on integrating fault-tolerance and traffic engineering possibly be used with FatTire. Likewise, the Flow-based Management Language (FML) [184] specify policies using a declarative language to enforce policies within the enterprise. For instance, Access Control Lists (ACLs), VLANs and policy-based routing. This differs from FatTire as it does not provide a fault-tolerance policy. Similarly, Kazemian et al. [150] introduce NetPlumber, a real-time tool for policy checking based on Header Space Analysis (HSA). The authors argue that they have applied this tool to Google's SDN and Stanford backbone and analyzed that 50-500 μs on average were required for a rule update against a single policy.

Chandrasekaran et al. [175], [176] claim that they have developed a fault-tolerant SDN controller framework called LegoSDN. The authors aim to achieve recovery of SDN application against both deterministic and non-deterministic service failures. Extending this work further, the authors develop a prototype that isolates SDN applications from one and another, as well as from the controller, by running each application securely in a sandbox. Thus, all failures are restricted to their virtual isolated space.

2.3.5 In Summary

We provided a simple background on fault-tolerance and related concepts to develop a complete understanding of the topic. Our goal was to identify SDN fault-tolerance requirements specific to the SDN architecture and discuss approaches that can be used to improve fault-tolerance in SDN.

Current SDN research efforts were structured according to the three main layers of SDN architecture and categorized according to data, control, and application planes.

While exploring the topic of fault-tolerance in SDN, we have identified that each layer has its faults and fault-tolerance issues. This means that in order to achieve fault-tolerance different aspects and features are needed to be targeted, and no single-focused technology will be able to provide the reliability expected in commercial networks.

Recent research studies show that SDN can play a pivotal role in shaping and managing future dynamic networking environments, such as cloud-native networks, 5G mobile networks [185], wireless networks [186] and optical networks [187]. However, SDN fault-tolerance is still in its infancy, and there is a broad spectrum of opportunities for the research community to develop new fault-tolerance mechanisms, standards, monitoring, debugging and testing tools to enforce fault-tolerance in such dynamic networking environments, able to ensure carrier-grade reliability. Furthermore, in Section 6.3.2, we concluded by enumerating future research directions for SDN fault-tolerance development.

2.4 Service Assurance for Hybrid networks

The Service Assurance problem addresses the policies that CSPs impose to guarantee the contractual QoS to the end-users while ensuring optimal use of resources [188]. Hence, Service Assurance can be defined as the techniques required to maintain the service (services offered require compliance with a pre-defined service quality level) and solve customer and/or network triggered problems that can lead to service degrading (for instance, failing processes and policies set by the CSP). Better Service Assurance designs reduce service downtime. Here, network management, automation, and orchestration play an important role in ensuring that service providers design Service Assurance models able to meet business needs [189]. However, when considering the usual requirements set for future networks (5G and beyond) [190], several aspects of automating network management and Service Assurance for Hybrid Networks are often overlooked.

Societies, industries, and businesses are highly dependent on reliable network connections, provided by CSPs, to support all their information exchange [191]. Due to this societal impact, one of the service provider's top priorities is to offer adequate Service Assurances. This is quite more challenging in software-driven virtualized networks than in traditional static networks due to the rapidly changing dynamic nature of such software networks, including the relocation of the point of provision of some or all service components.

A typical Hybrid Networking scenario is depicted in Fig. 2.11. Hybrid Networks are combinations of two technologies PNFs and VNFs) to provide communication services to multiple locations, and their end-users, under a single seamless network infrastructure [192]. In other words, the network where services could be dynamically created on top of legacy physical devices or relying on recent software-based technologies such as SDN, NFV, and Multi-access Edge Computing (MEC). This type of implementation can offer flexibility and ease of service provisioning. However, there is a need to design and develop Service Assurance processes that can guarantee simultaneous integrated provisioning for these virtual and physical networks.
Software-driven technologies provide a centralized management model to control devices even in multi-vendor networking environments. New SLA requirements will need to be decided/applied in real-time to support mission-critical communications (ultra-reliable and low-latency applications [193]) to deal with the dynamics of network services in 5G and beyond networks [194]. Network management, automation, and orchestration are key concepts for any Service Assurance solution.

5G networks are based on the concept of service-oriented architectures [195], [196] that is capable of supporting multiple services that could have specific performance requirements. For instance, the three main use cases of 5G are Enhanced Mobile Broad-

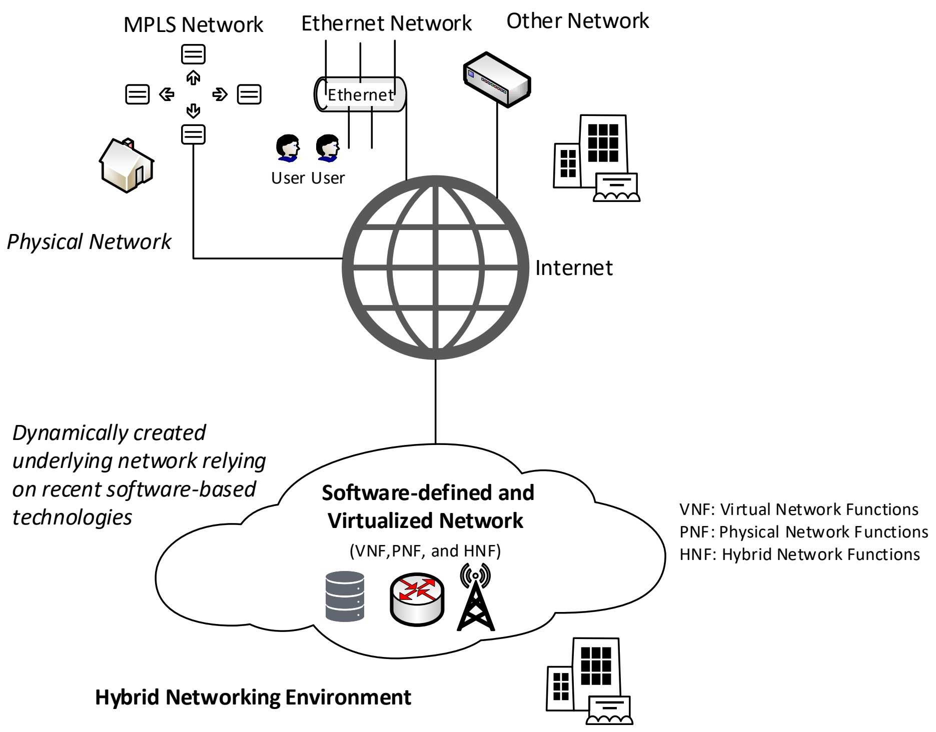

MPLS Network
Ethernet Network
Other Network
Ethernet
User User
Physical Network
Internet
Dynamically created underlying network relying on recent software-based technologies
Software-defined and Virtualized Network
(VNF,PNF, and HNF)
VNF: Virtual Network Functions
PNF: Physical Network Functions
HNF: Hybrid Network Functions
Hybrid Networking Environment

Figure 2.11: Hybrid Networking Environment

band (eMBB), Massive Machine-type Communication (mMTC), and Ultra-reliable and Low-latency Communication (URLLC) [197], [198] have different performance requirements in terms of metrics as scalability, throughput, and latency. Supporting multiple services with distinct performance objectives under a single monolithic architecture would be costly. 5G network addresses this problem with the E2E slicing concept [199] to ensure that each network slice can not only support the contracted (subscription and service level agreements) network services as well as it meets the specific performance objectives by dynamically allocating shared infrastructure resources.

In 5G, network slicing is considered one of the key enablers to support multiple verticals [200], [201] simultaneously, by delivering dedicated and isolated slice services over a shared information communication technology infrastructure. These verticals have explicit SLA requirements that must be guaranteed by mechanisms like Service Assurance [202]. Furthermore, Service Assurance must take into account inter-dependence across various business relationships. The impact of this on Service Assurance is discussed in Section 2.4.1. Next, we discuss Hybrid Network applicability scenarios. Hybrid Networks challenges and opportunities are then discussed and conclusions are drawn in the final section.

2.4.1 Applicability

5G is the critical next-generation networks technology, supports a multitude of vertical stockholders. Network slicing [49], [203] is being considered to support these new markets and business models. In 5G networks [204], [48], a network slice is an independently managed and isolated environment of network functions and infrastructure resources. 5G network slicing has been introduced to support multi-tenancy, multi-vendor, and multi-domain Hybrid Networks. As shown in Fig. 2.12, network services can then be sliced and deployed within the same underlying infrastructure [49], [48]. Note that proper performance metrics for assuring network services are critical; different performance metrics need to be defined to support specific scenarios.

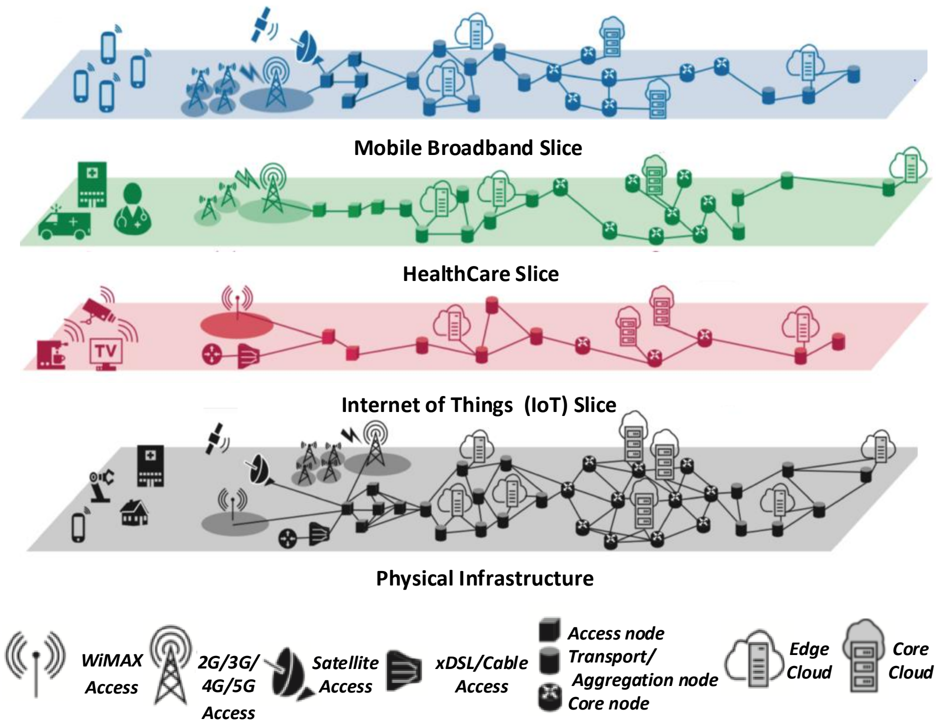


Figure 2.12: Multi-tenancy, Multi-vendor and Multi-domain Networking Scenarios (Network Slicing) [48]

This concept of slicing in 5G brings different KPIs requirements per slice, depending on the services to be supported. Service Assurance must be designed to manage these different KPIs on each supported service. In multi-domain virtualized networking environments, new services and products, with their associated SLAs, will evolve substantially. The challenges are increasing many-fold, with diverse end-user service metrics needing to be considered simultaneously.

All tasks inside operators must support integration within the OSS stack, and the same applies for all layers of Hybrid Networks, in order to enable policy-based management that defines SLAs [205]. This can enable smooth and rapid network

provisioning and Service Assurance across multi-domain networks.
Besides slicing, with 5G, the Telecom business environment is expected to grow in overall complexity. Platform ecosystems will emerge, both in specific verticals [206] or based on more general ecosystem platforms, as depicted in Fig. 2.13. Fig. 2.13 illustrates the network/service provider's roles with enabled access to Value-added Connectivity (VAC) across typical actors to offer network services in a multi-domain environment. There

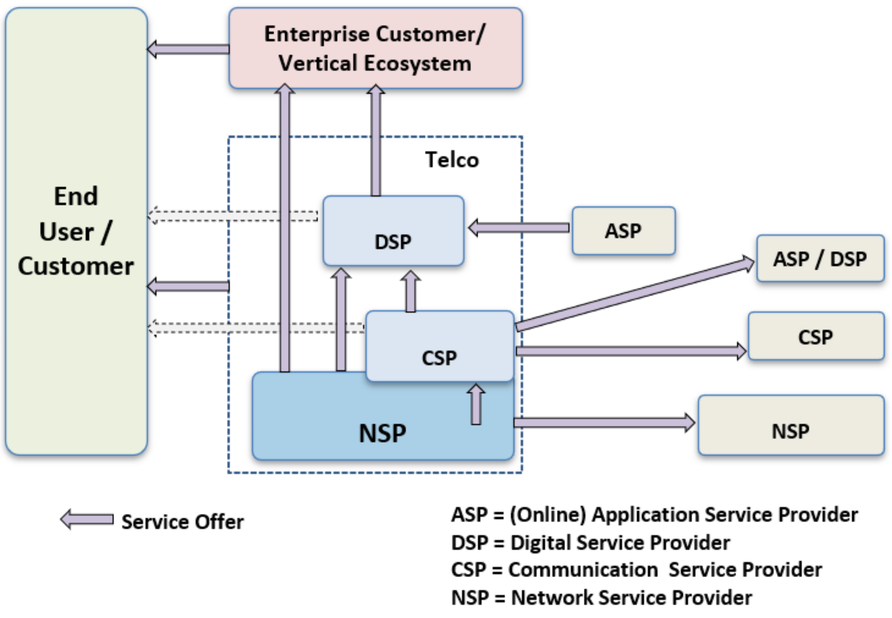

Enterprise Customer/ Vertical Ecosystem
End User / Customer
Telco
DSP
ASP
ASP / DSP
CSP
CSP
NSP
NSP
Service Offer
ASP = (Online) Application Service Provider
DSP = Digital Service Provider
CSP = Communication Service Provider
NSP = Network Service Provider

Figure 2.13: Service Assurance Needed in Multi-actor Environment

will be an increased focus on Business to Business (B2B) service offerings requiring automation and standardization of common capabilities and APIs within and across verticals. Service Assurance will evolve to consider many kinds of (interdependent) business relationships. The CSP role of today (Business to Consumer (B2C) focused) typically addresses the market of fixed / mobile broadband and the Telco Voice and Short Message Service (SMS) services for the consumer market. 5G on the other hand, enables specialized services ranging from mission-critical to consumer critical services, will expand. The Network Service Provider (NSP) role (B2B focused) will become more distinct and VAC between any two end-points will need frequently to be delivered on-demand across multiple domains. NSPs will need to open up to all actors in a transparent way to allow access to end-to-end VAC across vertical domains. The Service Assurance analysis and solution proposals must address this complex multi-actor environment with interdependent business relationships, requiring a completely different set of trust considerations.

Until recently, Service Assurance has been supported mostly by reactive processes.

However, data-driven operations powered by artificial intelligence and automation, enable now a shift from reactive to proactive processes [9].

Reactive processes are triggered by an event due to data abnormalities and then a pre-defined procedure is activated to resolve the issue, for instance, rerouting of the network service due to QoS violations. In Proactive processes data abnormalities are controlled without a triggering event, for instance, pre-empt issues to prevent predicted SLA violations.

It is important to note that, in order to support Service Assurance of Hybrid Networks, CSPs must strategically transform their Service Assurance processing primarily to Proactive monitoring models. Proactive monitoring models enable "zero-touch" automation where no human intervention is required. Furthermore, on the identification of any network related fault/service abnormality, the Proactive process automatically instantiates to resolve the service-related network issues. However, it is quite challenging for CSPs to model Service Assurance for Hybrid Networks [207], [208] because it requires widespread advancement in automation tools, protocols, and standard data models, across different technical areas [9].

Software-based hybrid systems inherently represent a key value for the delivery of better Service Assurance systems, providing more flexibility through increased degree of freedom. For instance, virtualization can enable live migration of service components to deal with potential SLAs violations, while slicing can provide configurable services for specific tenants and better fulfillment of elasticity policies that can help to cope with unexpected load peaks more easily. In this regard, it is important to identify any SLAs breached or risk of such breaches, so that they can be reported quickly to the (usually pro-active) fault-management system [209].

Recent 5G Public Private Partnership Projects (5G-PPP) [210], [211] considered the Service Assurance problem and provided dynamic monitoring solutions and control loops that are triggered based on the services' SLAs and propagate events to orchestration systems, with minimum delay, even in a multi-domain scenario. These recent projects have touched on Service Assurance aspects. However, existing solutions to address these Service Assurance problems are still in their infancy and there is a need to develop a unified Service Assurance model within the scope of Hybrid Networks.

2.4.2 Service Assurance for Hybrid Networks: Requirements and Key Consideration

In this section, we discuss requirements, and key considerations for Service Assurance for Hybrid Networks by considering and discussing key enablers and key technologies that can potentially roll out Service Assurance for Hybrid Networks.

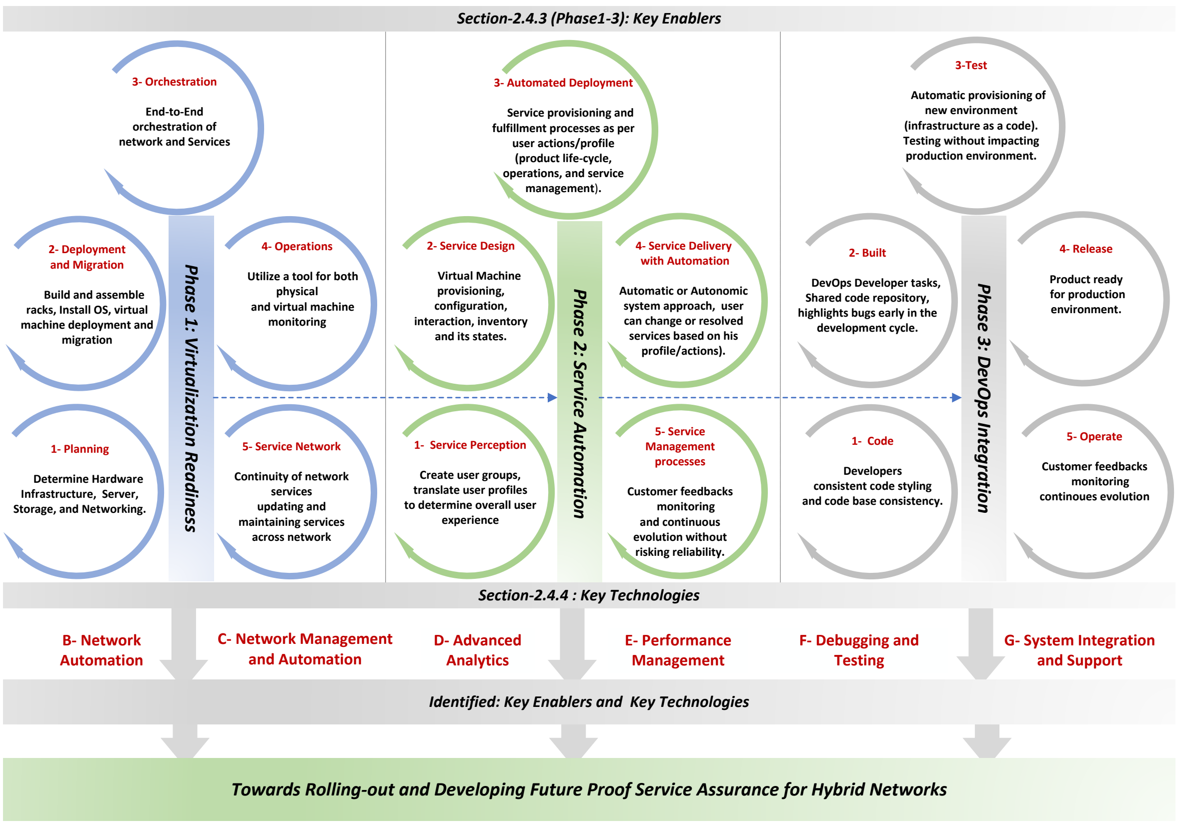
2.4.2.1 Key Enablers and Key Technologies

Here, we consider and discuss three main phases of rolling out Service Assurance and developing future proof Service Assurance for Hybrid Networks are as follows: 1) Virtualization Readiness, 2) Service Automation, and 3) Development and Operation (DevOps) (a set of practices that combines software development (Dev) and IT operations (Ops)) integration. The details of these phases are as follows:

1. **Virtualization Readiness**
   The first step is virtualization; in this phase, the main focus is to build a virtual infrastructure (utilizing modern virtualization approaches) that can provide services and assure performance equivalent to the performance achieved with hardware-based infrastructure, without compromising the customer QoS expectations.
2. **Service Automation**
   In the second phase, the main focus is on service provisioning (product lifecycle, operations, and service management) processes without human intervention and without risking reliability.
3. **DevOps Integration**
   In the last phase, these automated solutions must be coupled with DevOps practices, in order to support a higher level of debugging and software testing to enable deploying new services and products.

As shown in Fig. 2.14, the Phases 1, 2, and 3 (the key enablers) have multiple stages and development lifecycles. We discuss briefly these details in this Section 2.4.2.1 (Phase:1-3); we then identify and discuss topics in Section 2.4.2.2 (designated as key technologies). The identified key enablers, as well as the key technologies, are mandatory prerequisites to the roll-out of Service Assurance for Hybrid Networks, considering the trend towards network slicing and softwarization [203] of telecommunication systems.

Phase-1: We identify the virtualization readiness first and foremost as the phase necessary for creating a Hybrid Networking environment to support federated hardware infrastructures including storage, server, and networking [212]. Furthermore, as shown in Fig. 2.14, deployment, and migration policies for VMs, orchestration, operations, and servicing networks are components of this phase that have their own peculiarities. As such, all these aspects require detailed knowledge and attention before implementing a Hybrid Network. Virtualization readiness aims to maximize data-center operations and it is mandatory because it provides an abstracted view of (some) hardware to create flexible and dynamic environments without worrying about vendor-specific pieces of equipment. Generally, the virtualization environment is deployed (hopefully) over COTS equipment. Besides this, virtualization should be planned properly for the



Section-2.4.3 (Phase1-3): Key Enablers
3- Orchestration
End-to-End orchestration of network and Services
2- Deployment and Migration
Build and assemble racks, Install OS, virtual machine deployment and migration
4- Operations
Utilize a tool for both physical and virtual machine monitoring
1- Planning
Determine Hardware Infrastructure, Server, Storage, and Networking.
5- Service Network
Continuity of network services updating and maintaining services across network
Phase 1: Virtualization Readiness
3- Automated Deployment
Service provisioning and fulfillment processes as per user actions/profile (product life-cycle, operations, and service management).
2- Service Design
Virtual Machine provisioning, configuration, interaction, inventory and its states.
4- Service Delivery with Automation
Automatic or Autonomic system approach, user can change or resolved services based on his profile/actions).
1- Service Perception
Create user groups, translate user profiles to determine overall user experience
5- Service Management processes
Customer feedbacks monitoring and continuous evolution without risking reliability.
Phase 2: Service Automation
3-Test
Automatic provisioning of new environment (infrastructure as a code). Testing without impacting production environment.
2- Built
DevOps Developer tasks, Shared code repository, highlights bugs early in the development cycle.
4- Release
Product ready for production environment.
1- Code
Developers consistent code styling and code base consistency.
5- Operate
Customer feedbacks monitoring continoues evolution
Phase 3: DevOps Integration
Section-2.4.4 : Key Technologies
B- Network Automation
C- Network Management and Automation
D- Advanced Analytics
E- Performance Management
F- Debugging and Testing
G- System Integration and Support
Identified: Key Enablers and Key Technologies
Towards Rolling-out and Developing Future Proof Service Assurance for Hybrid Networks

Figure 2.14: Propose Rolling-out Approach.

long-term, keeping business objectives, solutions, and customer service expectations in mind. Furthermore, it should also consider the agility to deploy new on-demand services on the fly with the right skills, management tools, and trusted standard interfaces across the network.

Phase-2: We consider the service automation phase in Hybrid Networking after virtualization readiness. Identified key enablers (phase: 1-3) rely on underlying technologies required to seamlessly integrate the physical and virtual networks as shown in Fig. 2.14. These key enablers will bring a new set of challenges to the six technologies discussed in Section 2.4.2.2, that will be important for Hybrid Networks Service Assurance. CSPs identified the processes that required manual intervention are limiting them to achieve a higher level of orchestration, Service Assurance, and automation. To overcome these limitations, CSPs offer of new services, and increase CSP profitability, service automation is essential for E2E service over Hybrid Networks. Machine learning and artificial intelligence approaches are considered vital to simplify and improve service automation.

Currently, due to the high Total Cost of Ownership (TCO) [213] and long times needed for testing and deploying a service, operators are interested to migrate from PNFs to VNFs. However, this new environment brings new challenges. For instance, Hybrid Network brings new complexity of managing physical and virtual infrastructure simultaneously and dealing with an increase in operational cost. To handle simplified operations, unified orchestrators and container-based applications are important to manage this delicate complexity. Automation can simplify operation and overcome this delicate complexity as well as brings agility, faster innovation needed to capitalize on new services. Furthermore, it promises to reduce the time to market for new products and services.

Phase-3: We recognize the need for the DevOps Integration phase in Hybrid Networking to provide system integration support to deliver, evolve, and improve, applications at a faster pace. DevOps [214], [215] is a culture for promoting the softwarization of the telecommunication system. The traditional siloed, waterfall-based Service Assurance model is not suitable to support a Hybrid Networking environment encompassing NFV [216], SDN [217], MEC technologies [218]. DevOps models are no longer traditionally siloed and quite often are tightly integrated into the application development lifecycle. Integrating DevOps methodologies are essential to test and validate E2E services in PNFs and VNFs based deployments. DevOps oriented automation enables faster innovation to the software development and infrastructure management processes. The best DevOps practices include continuous integration, continuous delivery, microservices, infrastructure as code, monitoring and logging, communication, and collaboration.

DevOps would be used for Service Assurance (design, automation, lifecycle management, and business integration) by automating manual tasks and maintaining repeatable processes with better transparency while maintaining stability and integrity, in order for operators to offer zero-touch Service Assurance (ensure uninterrupted business continuity and provide significant reduction of unplanned downtime and associated cost).

DevOps based Service Assurance is a promising technology towards rolling out Service Assurance and developing future proof Service Assurance for Hybrid Networking along with the discussed three key enablers and six technologies as depicted in Fig. 2.14.

2.4.2.2 Key Technologies

1. **Network Automation**
   Network automation is vital because it reduces TCO, and further reduces the cost for Information technology staff as the network expands [219]. It supports the processing of standard workflows based on automated tasks, such as interface shutdown, virtual machine failure, and others. There are protocols (such as

NETCONF, RESTCONF) and data formats and models (such as Yet Another Next Generation Language (YANG), Yet Another Markup Language (YAML), JavaScript Object Notation (JSON), and Extensible Markup Language (XML)) that are being used to process such standard workflows [9]. These protocols are used to simplify network management and provide automation through programmability (configuring network devices directly through custom scripting using programming languages). There are several IT configuration management tools and automation platforms (such as Puppet, Chef, Ansible, and Salt are used to deploy, configure, manage, and automate servers [220]). Currently, these protocols, configuration management tools, automation platforms, data models, and formats are evolving (still going through standardization acceptance, testing and development phases) and potentially can be applied in software-driven networks to offer more powerful network automation features. Furthermore, network automation using machine learning and artificial intelligence can make closed-loop automation possible with Hybrid Networks. A more detailed analysis of Network automation using machine learning and artificial intelligence is described in [221].

Considering the need for network automation to support Service Assurance systems for future networks, two important concepts exist: automatic and autonomic [222]. An automatic system relies on the execution of a certain rule, previously configured, and does not require human intervention in known scenarios. This type of system is robust and efficient to automate, but such a system is not capable to optimize system performance in non-familiar situations. As a result, an automatic system requires configuration changes when the network environment (topology) changes. Thus, they are not truly fully automated Service Assurance systems. Autonomic systems, on the other hand, offer efficient self-management using advanced data analysis technologies, such as machine learning and artificial intelligence, to drive cognitive and behavior-centric approaches [223]. Based on this data analysis, an autonomic decision is invoked to enable self-healing in order to restore optimized network services to their best operational condition. Thus, autonomic systems offer full automation and enhanced Service Assurance, and are thus a requirement for Hybrid Networks [224].

2. **Network Management and Orchestration**

   Network management is challenging [225], as traditional management follows a per box approach to configure network devices, with significant human intervention involved. Other challenges that make network management difficult[226] are: coupled data and control planes, closed propriety interfaces, and vendor-specific devices. Systems and processes must be standardized to support network management for Hybrid Networks. Therefore, network management must be addressed holistically.

To address the management and orchestration aspects for emerging software-driven networks, two major frameworks, Open source Management and Orchestration (OSM-MANO) and Open Network Automation Platform (ONAP), have recently attracted significant attention from both academia and industry [227]. OSM-MANO is an ETSI initiative for the development of management and orchestration Open source software for software-driven networks, and is associated with the development of 5G management and orchestration. ONAP is a project initiated by the Linux Foundation to provide cloud providers and developers a platform to enable real-time automation and orchestration of both PNFs and VNFs, from instantiating new services to supporting life-cycle management of products. OSM-MANO has been adopted by major European network operators while ONAP has more support in North America and Asia. However, Service Assurance aspects in both frameworks are not yet mature and keep on evolving. Automation is one of the features provided by a network orchestrator; it is essential that the merits of both technologies get integrated to achieve enhanced Service Assurance support. For instance, it is important to develop a comprehensive platform capable of providing orchestration, management, and automation of network and edge computing services for network operators, cloud providers, and enterprises. Real-time, orchestration and automation of physical and virtual network functions can enable rapid automation of new services and their life-cycle management that are considered critical for 5G and B5G networks.

3. **Advanced Analytics**

   The traditional existing Service Assurance methods are defined to deal with static networks and cannot deal with the dynamicity that virtualization brings forward. In order to operate in a dynamic control environment, it is necessary to have the data required to process a decision in order to assure the health of the network continuously. This requires advanced data analysis tools [228] that can perform data optimization automatically in order to enhance Service Assurance, an essential requirement for autonomic systems. Two major aspects are as follows:

   - Determining Root Cause Analysis: To be able to automatically detect root causes of failure, artificial intelligence, and machine learning techniques and algorithms [229] are used to gather data to be able to localize faults, instantiate self-healing processes and improve data quality while making decisions to offer network automation. Furthermore, combining network operation and management aspects into one unified framework with the use of data analytics can provide better data correlation and identification of root causes in the network. This helps reduce operating costs as well as network downtime.

- Monitoring: To operate in a dynamic control environment continuously and monitor the health of the network, it is necessary to have the data needed to process a decision. This requires advanced data analysis tools that can collect automatically the data, analyze it, and recommend actions to orchestration systems. It is essential to continuously monitor data exchange through a closed-loop feedback system (collecting and analyzing the data for network optimization and implement changes through automation) to improve service reliability and resilience [210]. This enables operators to create and maintain a self-healing network in real-time thus advancing the automation of network management (automatic versus autonomic systems) and offering high-class service delivery/experience to end-users and enhances business services (end-user Quality of Experience (QoE) impacts assessment [230]).

4. **Performance Management**
In multi-domain, multi-technology and multi-vendor scenarios, performance remains an issue. Performance management functions must detect abnormal behavior reactively or proactively, based on collecting and analyzing QoS metrics to ensure that network performance in multi-domain networks is not compromised. In order to enhance performance capabilities in traditional networks, metrics such as packet loss, latency, link utilization, Mean Opinion Score (MOS) [231], and QoE analytics are statistically analyzed in order to evaluate the strategy and design better service experience for customers. However, in Hybrid Networks, these metrics must be analyzed using integrated data analytics tools controlled by advanced machine learning and artificial intelligence techniques, able to consider physical aspects at the same level than virtualization constraints, at the logical and infrastructure levels. These advanced data analytics enable E2E view across networks, and thus can significantly improve the performance of the system.

5. **System Integration and Support**
Recently, two distinct methods of developing software systems, Micro-services and DevOps [232], attracted significant attention from enterprises. Both aim to offer greater agility and operational efficiency. Both methods were not just limited to transforming monolithic applications into decomposed services (Micro-services), but rather they are meant to support a culture of agile development with virtualized instantiations. Furthermore, Micro-services and DevOps practices are better when deployed together because Micro-services bring additional

productivity to DevOps (enable enterprises to employ common tools for both software development (Dev) and IT operations (Ops)).
However, DevOps practices bring several new challenges that need to be addressed before its widespread deployment [233]. For instance, moving from monolithic to Micro-services approaches can be risky without enhanced automation (handling appropriately the granularity that Micro-services bring forward), and resilient designs.
The culture of agile development can be critical for Service Assurance design. A new Service Assurance design for Hybrid Networks must be realized considering DevOps and Micro-services approaches. This includes network service as well as Service Assurance testing and validation.

6. **Debugging and Testing**
 Software-driven networks explore programmability to control network devices more efficiently. However, this highly depends on the quality of software development, debugging, and testing tools that support software quality and fix software bugs.
For CSPs, any product developed must be ideally designed to offer error-free service. To ensure the quality of software both testing and debugging are essential for software-defined network functions [136]. Testing is used to find different defects in codes such as errors, bugs through manual and automated processes. Debugging is a multi-step process to identify the problem and remove it (from software or system). Some basic strategies that could be used to achieve efficient testing and debugging in Hybrid Networks are as follows:
   - White-Box: In white-box/clear box testing, all the internal implemented structures are known to the tester. The tester can determine appropriate outputs through testing all inputs.
   - Black-Box: In black-box testing, the internal implemented structures are not known to the tester. The tester can test a few possible numbers of inputs to determine the appropriate output.
   - Gray-Box: In grey-box testing, a combined approach based on black and white box principles is applied. In the gray-box, the internal structure of the component is partially known but not for all the components.

   Considering Hybrid Networks Service Assurance, the preferred approach would be white-box testing because if we know the internal structures of the network components, it becomes easier to analyze network behavior, even if networks are changing rapidly. This would provide enhanced trust on automation; flexibility to manage, troubleshoot and configure rapidly changing dynamic networks; and limit the failures to a minimum. Unfortunately, realistically both multi-domain and legacy networks with closed interfaces restrict the CSP to offer limited network management and automation, and will impose black or grey-box approaches. A more detailed comparative analysis of debugging support is described in [104].

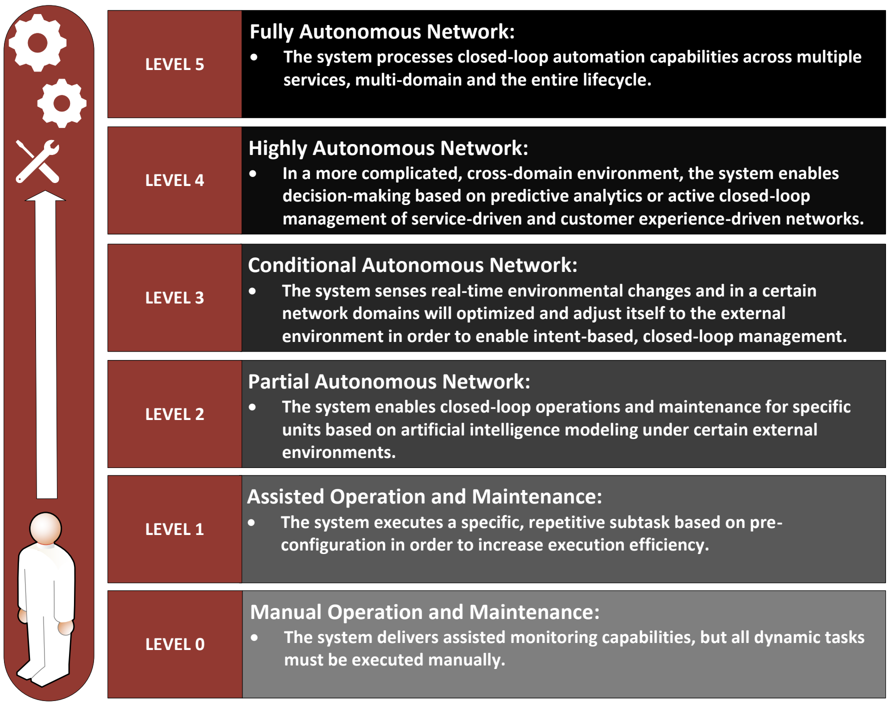
2.4.3 Discussion

5G is considered technology to revolutionize network services with innovation. This requires CSPs to improve their infrastructure level capabilities to support 5G innovation by reimagining the OSS/BSS system that includes embracing automation and artificial intelligence in Service Assurance [234]. The maturing of the Service Assurance for Hybrid Networks (as identified and discussed key enablers and key technologies (Section 2.4.2) means that appropriate tools are positioned to achieve strategic goals for assuring machine learning-based QoS assurance for 5G networks [235]. However, developing novel strategies, architectures, and standards to support QoS assurance for 5G networks is still in its infancy.

One of the key aspects of a new kind of Service Assurance system is service orchestration. Service orchestration relies upon an assurance solution that can generate high-quality data to enable the provision of network services in a multi-vendor environment across the networks. CSPs need to implement a new platform parallel to their existing siloed platform to support the new kind of assurance that relies on data analytics and aggregation from multiple sources in multi-domain networking environments. However, there is a long road ahead to replace the traditional networks with highly virtualized hybrid networks based on 5G [234].

5G standalone deployments are operationally complex as the core is based on a cloud-native architecture. It is important to note that cloud-native communications networks are more complex than traditional physical, and current virtualized networks. From the perspective of cloud-native that relies on a multi-vendor ecosystem, it is important that interoperability and data sharing must be optimal and standardized among multi-vendor systems. The TM Forum highlighted that their open digital architecture and Open APIs (enable plug-and-play interoperability of components within the IT systems) can assist CSPs to manage E2E service assurance [236].

As we discuss in Section 2.4.2.2 the concept of the autonomic system (a decision is invoked to enable self-healing in order to restore optimized network services to their best operational condition with fully automated mode). Similarly, TM Forum autonomous network Group identifies six levels of advancements as shown in Fig. 2.15 in order to move toward zero-touch operations and management [236]. Furthermore, they argue that most of the CSPs Service Assurance systems are operating at Level-2 and therefore,


LEVEL 5

Fully Autonomous Network:

- The system processes closed-loop automation capabilities across multiple services, multi-domain and the entire lifecycle.

LEVEL 4

Highly Autonomous Network:

- In a more complicated, cross-domain environment, the system enables decision-making based on predictive analytics or active closed-loop management of service-driven and customer experience-driven networks.

LEVEL 3

Conditional Autonomous Network:

- The system senses real-time environmental changes and in a certain network domains will optimized and adjust itself to the external environment in order to enable intent-based, closed-loop management.

LEVEL 2

Partial Autonomous Network:

- The system enables closed-loop operations and maintenance for specific units based on artificial intelligence modeling under certain external environments.

LEVEL 1

Assisted Operation and Maintenance:

- The system executes a specific, repetitive subtask based on pre-configuration in order to increase execution efficiency.

LEVEL 0

Manual Operation and Maintenance:

- The system delivers assisted monitoring capabilities, but all dynamic tasks must be executed manually.

Figure 2.15: Automation: Six stages of advancement [234]

this the significant area for CSPs to develop, transform and modernize it in order to accelerate the deployment of fully automated E2E orchestration of 5G and B5G services.

2.4.4 In Summary

Effective Service Assurance practice enables CSPs to identify and resolve various issues and faults, and thus prevents customers from service quality degradation and offers CSPs a high return on investment. This is achieved by higher utilization of resources and increased utility for the customers by offering predictable QoE. Due to the advent of new technologies such as SDN, NFV, MEC, and the concept of network slicing in 5G, CSPs are interested to invest in new Service Assurance models for Hybrid Networks in order to reduce cost, support new services and enhance the customer experience. Networks are evolving, and in the near future, there will be many networks with multiple needs and more demanding applications and different Service Assurance requirements. Therefore, there is also the need to re-think and consider developing Service Assurance systems to fulfill the real-time needs of Hybrid Networks. However, CSPs are reluctant to upgrade all their infrastructure to new Service Assurance models oriented to Hybrid Networks, because they have to invest a huge amount of money for networking infrastructure, and would remain worried for return on investments. Therefore, it is critical for CSPs to tackle Service Assurance challenges and topics addressed above in order to accelerate the Service Assurance development for Hybrid Networks.

In order to automate future networks without human intervention (transforming fully automated networks), both in real-time and in predictive networking environments, next-generation Service Assurance must be equipped with proactive monitoring and advanced technologies such as machine learning, advanced analytics, and artificial intelligence with well-accepted trusted standard interfaces across different business partners.

Effective Service Assurance systems must be based on a unifying model and integrated with the OSS to track performance, for the purpose of meeting the SLA requirements and to identify and manage network failures.

It is now time for service providers to address the challenges of Service Assurance and transform their Service Assurance model to meet the requirements of Hybrid Networks. Otherwise, the promised benefits of virtualization will lag and service providers may not be able to enhance their services most appropriately. Furthermore, in Section 6.3.3, we concluded by enumerating the future research directions of the Service Assurance development for Hybrid Networks.

Chapter 3

SDN Control Network

Chapter Outline: This chapter focuses on the management and automation concepts of future reliable and dynamic networks. A few research studies have been carried out that highlighted SDN reliability (fault-tolerance) aspects. Considering fault-tolerance aspects, we explore new SDN resilience and reliability mechanisms and proposed out-of-band, Self-Healing, and centralized hybrid SDN control network design. In addition to that, there is an added in-band control mechanism to support resilience in SDN to guarantee higher reliability.

3.1 Background

In SDN data and control planes are connected through a Controller. This controller is called the "brain of SDN", while the switches and network devices are regarded as the "hands of SDN". Furthermore, the network controller is often referred to as a network operating system comprised of a control plane and has a centralized view of the network. Therefore, controller availability and liveness are always essential to enforce reliable communications. In communication networks, control plane resilience is an essential requirement for ensuring the proper operation of the network functions. In SDN the controller handles packet flows and routes network traffic. Hence, the controller must be always operating.

In-band and out-of-band signaling solutions have been adopted to offer SDN control plane reliability [237]. In practice, most SDN deployments use out-of-band control, where control packets are managed by a dedicated management network [134]. In a study carried out by [238], the authors have presented a model for the design of an in-band distributed control plane to build an adaptive SDN control network. Due to the split architecture of SDN, apart from the data plane failure, the controller must also be resilient against failures of control messages against targeted network attacks. There are some approaches highlighted that offer resiliency for SDN controllers. The first approach is to replicate a controller on a separate control network. In the case of failure, the other controller takes over and manages traffic unless prone to further failure. In another approach, the controller must be embedded with mechanisms (Build-in Module) to Self-Heal from targeted attacks such as DoS, flooding, fake traffic routing, and other network-related targeted attacks [239].

We argue that the control plane recovery time to recover from such attacks is critical and ideally recovery mechanisms must be developed to mitigate failures within the set requirements of network performance. For instance, in the case of CGNs, recovery time must not exceed 50 ms, while in other cases may vary depending on the reliability requirements. Besides these, the recovery process must be efficient and must be able to Self-Heal during an event of a failure with minimum overheads. We argue that covering these aspects enforces the SDN control plane to implement a flexible and reliable communication process. Thus, we believe that this guarantees network management and automation for SDN and future networks.

3.2 Proposed Methodology

In this section, we considered data and control plan resiliency to offer reliability in SDN. One of the important aspects that SDN must offer is scalability. Therefore, we explore SDN control network suitability to meet the reliability and resiliency requirement for CGNs. In CGN the recovery mechanism must ensure the recovery process within 50 ms [240]. For this purpose, Restoration and Protection are widely used to recover from system failures. In Restoration, an alternative path is only established after the occurrence of failure, and network resources are not reserved before the event of a failure but recovery paths may be pre-assigned or allocated dynamically. Also, additional signaling is required to recover from failure. Therefore, this solution is not scalable because of not meeting the requirements of CGNs [241], [242].

However, in the case of Protection, alternative paths are already reserved and assigned before the occurrence of a failure. Moreover, in protection, no additional signaling is required. Thus, the recovery process is fast as compared to the restoration method. Due to no additional signaling requirement, the recovery process in the protection method is possible within 50 ms. Thus, it is a scalable and more appropriate solution for CGNs. In this study, we aim to apply restoration and protection techniques to our proposed hybrid control plane (handling both legacy and SDN switches). This enables us to build a flexible design that guarantees reliability and resiliency of data and control plane in SDN and future networks.

3.3 Theoretical and Practical Implications

Due to the associated risks with SDN described in [239], existing research studies do not guarantee optimal reliability. The out-of-band control has a cost constraint while in-band control adds complexity and challenges to design and bootstrap. Moreover, restoration is not scalable while protection is. Eventually, these solutions add complexity to the design of SDN-based reliable and resilient systems. Due to these intricate balances, the current solutions offered are either costly or complex. Our proposed hybrid SDN control plane, therefore, provides a direction towards designing resilient and reliable SDN control networks.

Furthermore, the multi-tenancy nature of future network environments adds a fundamental problem to the issue of network dynamics, namely that of the varying veracity of information. This variation (of accuracy) can stem from a variety of reasons, such as lack of proper information visibility. For instance, due to infrequent intervals for retrieving the exact values to keep the signaling overhead lower. This veracity of information forces any control framework to deal with uncertainty. Thus, risk management becomes a key instrument to address this change. We also aim to extend our proposed approach further to address network dynamics through risk management. This means that network dynamics can be better dealt with by defining a risk assessment matrix indicating the severity of each risk and a possible reliable approach for Self-Healing in real-time SDN environments.

3.4 Fault Detection and Mitigation

The Restoration and Protection techniques have been proposed to recover from network service failure for both in-band and out-of-band control. The study carried out by [237] discussed the challenges of fast in-band OpenFlow-based network recovery. Moreover, in the normal operation of in-band control in the presence of a dormant fault, the switch flood control traffic out of port to all nodes by default. However, a problem arises when the fault becomes active: in this case, the neighboring switch forwards packets where another switch may drop because of intermediate switch disconnection.

Due to the fact that in in-band control the data and control traffic is sent over the same link, the procedure is not simple as compared to out-of-band control. In out-of-band control, control messages are sent over a separate network. This results in simplified switch implementation. Since switches cannot interfere with control messages, this increases reliability.

3.5 Fault-Tolerance and Self-Healing

We believe a network system is referred to be fault-tolerance if it can Self-Heal in the presence of Failures, Errors, and Faults. A failure happens if a system is unable to implement the system function and a service deviates from the correct service. One of the examples of service failure is that a system is not capable of restoring from one state to another state. The cause of error refers to when the sequence of one or more external system state deviates from the real service. A hypothesized cause of the error refers to a fault. For instance, a software bug, Human-made error, or hardware power failure. In fault-tolerance, the error detection and system recovery strategies are used to avoid failures [243]. In "Telecommunication" Self-Healing refers to re-routing Traffic due to a fiber cut e.g. Synchronous Optical Networks (SONET) and Synchronous Digital Hierarchy (SDH). However, in SDN, adapting to real-time demands and network dynamics. Self-Healing minimizes human intervention due to advanced monitoring based on the constant intake of live data from the network environments. Moreover, in the case of abnormal data behavior, the system automatically invokes the Self-Healing process for recovery.

3.6 Proposed Resilient Design to Offer Reliability in SDN Networks

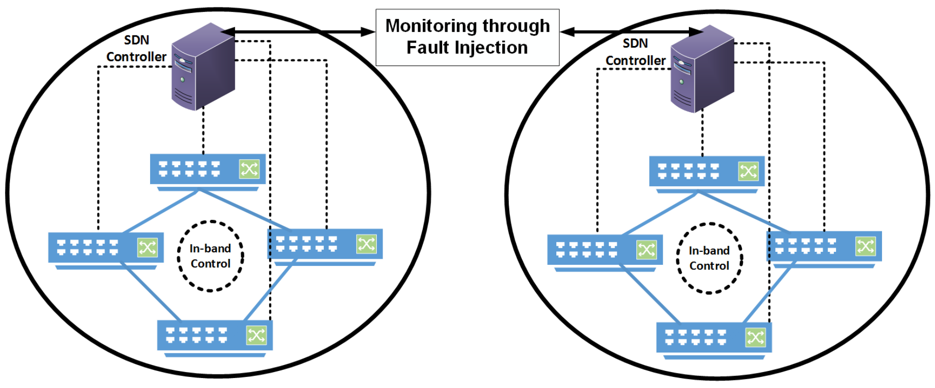


Figure 3.1: Resilient Design through Hybrid Control Plane with Reliability

As depicted in Fig. 3.1, we propose out-of-band control in SDN with additional in-band control to offer fault-tolerance. This resilient design ensures reliability between the controller and underlying devices. For ensuring additional reliability a backup controller takes over in the case of any network failure. To guarantee proper operation of the Controller-to-Controller communication we propose to monitor this communication through Fault injection mechanisms which ensure guaranteed reliable communication with improved performance. We argue that the Fault injection is important to evaluate the dependable communication [244]. This involves the study of fault, error, and failures. Moreover, removing these errors or minimizing them is a basic requirement that must be fulfilled to develop highly reliable fault-tolerant SDN networks. Therefore, we included Fault injection in our proposed design to monitor Controller-to-Controller communications.

3.7 In Summary

In this chapter, we highlighted SDN reliability concerns and proposed a resilient design to offer future proof reliable communication in a software-defined network-based environment. In our proposed design we have considered the possibility of legacy as well as the standalone deployment of SDN with redirection of traffic flow in the event of failure of the primary SDN controller.

In the next chapter, we set up a scenario-based experiment in which a critical and reliable communications environment is implemented using a software-defined network-based environment. In this line, using SDN to redirect the necessary data flows in the event of a failure. To achieve zero downtime between failures, an SDN Recovery Mechanism was implemented and evaluated. However, we have not evaluated SDN controller channel reliability itself, as it requires implementing Fault injection approaches in order to ensure the reliability of the SDN controller channel (Controller-to-Controller communications). Furthermore, in Section 6.2.1, we discussed and outlined the directions for future research concerning the proposed resilient design that could offer reliability in SDN control networks.

Chapter 4

Reliable 5G Networks

Chapter Outline: This chapter addresses the role and impact of 5G key enablers technologies in reliable communications. In this regard, first, it compares the use of two virtualization technologies, namely containers and unikernels, for VNF instantiation in computational limited resources, envisioning the deployment of VNFs close to the end-nodes contributing to facilitate resilient communication networks in critical scenarios. Second, it addresses the concern that the probability of failure increases with the hardware limitation, imposing the development of failure detection and recovery mechanisms. In this line, we developed a failure detection and recovery mechanism able to ensure VNF reliability by dynamically instantiating a backup VNF before failure and using SDN to redirect the necessary data flows. Finally, the results showed that our mechanism was able to ensure near-zero downtime, showcasing the feasibility of the solution.

4.1 5G Critical and Reliable Communications Overview

The 5G telecommunications network aims to enable a fully connected society and empower socio-economic transformations in several ways that are unimagined today. To achieve such an all-connected society, the density/volume of traffic as well connectivity are expected to grow tremendously [245]. This prediction of traffic increase occurs because 5G not only will better support human-centric applications (e.g., virtual/augmented reality and ultra-high-definition video) but will also support the demands of machine-to-human and Machine-to-Machine Communications (M2M). With this large range of possible applications, 5G networks need to support a broad number of KPIs requirements [246].

To better achieve this variety of services and applications, new technologies are being introduced into edge and core networks, namely SDN and NFV. With an architecture that makes use of these two technologies, 5G will not only provide better performance for the end-user but will also provide a more flexible and dynamic provision of communications for different scenarios [247]. Another two promising enabling technologies are Fog and Edge computing. They are highly virtualized platforms that provide compute, storage, and networking services between end devices and traditional Cloud Computing data-enters [248]. Nevertheless, while Edge computing pushes the provided capabilities directly into devices like Programmable Automation Controllers (PACs), Fog computing pushes the capabilities to a device near to the end-user.

Fog/Edge computing integrated with SDN and NFV can significantly reduce the latency in network processing and provide fast instantiation and reliable connectivity to the end user [249] Critical and Reliable communications were and still are a major concern to mobile operators, since they are crucial for the secure operation of Machine-type communications, such as monitoring and control systems, vehicle-to-vehicle coordination, and cloud-based systems [250].

In this line, SDN and NFV play a major role in Critical and Reliable scenarios, offering the ability to instantiate the necessary optimal network behavior at the right time and place.

4.2 Problem Identification

SDN decouples the control plane from the data plane. With the decoupling of these two planes, the behavior of the network can be controlled by a logically centralized controller that has a full view of the network, providing a high level of programmability and allowing for dynamic network (re)configuration [119] NFV enables the virtualization of network functions, which can be implemented on VM or other virtualization platforms, on COTS servers [251].

Being reliability one of the major concerns when designing network architectures, making use of SDN, NFV, and fog computing, new types of services that have strict resilience and reliability KPIs can be developed or improved. For example, in [252] it is advocated that SDN can significantly improve data-plane reliability when incorporated in Long-term Evolution (LTE) / Evolved Packet Core (EPC) networks. SDN can also be a viable solution for providing redundant communications in Smart Grids environments as proposed in [253]. In [254], a new architecture of wireless SDNs is introduced which provides reliable connectivity and services to IoT applications. SDN can also be used for self-healing based frameworks for 5G networks, as proposed in [255] and can also be implemented as a link failure recovery mechanism [256], [257] proposed a method that uses active probing to detect and manage failures in an OpenFlow-based data center network exploiting load balancing among equal cost multiple paths. In [258], an algorithm that can minimize the physical resources consumption while guaranteeing the required high reliability with a polynomial-time complexity was proposed.

However, none of these studies combined SDN with virtualized network function deployment in constrained resources for reliability-demanding critical scenarios.

4.3 Architecture Overview: Description and Deployment

This section presents the motivational scenario, followed by the framework architecture overview and its building blocks description and deployment.

4.3.1 Scenario-based Experiment

Our scenario considers a critical and reliable communications environment where an ISP and/or Cloud Computing Service Provider (CCSP) hosts the NFs on behalf of an industrial corporation. The ISP/CCSP needs to ensure not only the placement of the NF (or VNF) near to the end-users but also reliable communications with a near-zero downtime between version updates and/or failures for those NFs. Due to the critical nature of these VNFs, they are managed as black boxes in order to prevent tampering, which means that the CCSP cannot modify in any way the operation of the VNFs. In this line, for our proof-of-concept framework, a Virtual Firewall (vFW) will be used as an example of VNF. The use-case scenario is depicted in Fig. 4.1

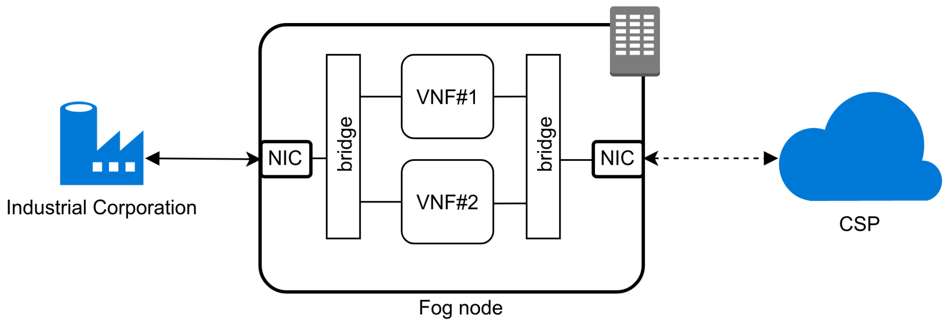


Figure 4.1: Use case scenario.

As stated, to cope with the reliability and latency requirements imposed by the industrial corporation, the CCSP needs to instantiate the VNFs close to the corporation. For this, the CCSP may explore fog computing, however, due to hardware resource limitations the probability of failure increases [259]. To overcome such failure events, the CCSP requires enhanced failure detection and recovery mechanisms to obtain near-zero downtime of VNFs failure and/or updates.

4.3.2 Building block and deployment

The fog node architecture is depicted in Fig. 4.2. Next, the building blocks of the framework architecture are presented, while describing their deployment.

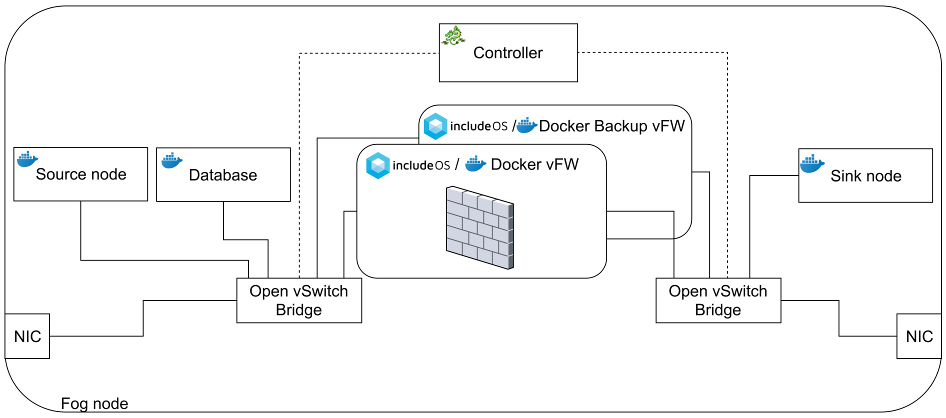


Figure 4.2: Fog node deployment for proof-of-concept evaluation.

4.3.2.1 Fog node

The fog node is a computing system that plays the part of a mini-cloud, located at the edge of the network. It was deployed in an APU2C4[1], a single-board computer developed by PC Engines capable of providing storage, compute, and networking services. This device has an AMD GX-412TC System-on-a-Chip (SOC), Central Processing Unit (CPU) with 4 cores, 4GB DDR3-1333, Dynamic Random Access Memory (DRAM), and a 16 GB Solid State Device (SSD) for storage. The OS installed was Ubuntu 17.10 with the 4.13.0-21-generic kernel. After the OS installation, the system was turned into a type-1 hypervisor with KVM[2].

4.3.2.2 Source node

The source node simulates industrial critical communication. For proof-of-concept purposes, it was deployed in a Docker container, however, our framework is independent of the source node implementation. In this line, in a commercial situation, the source node can be a REST service but its implementation is not in the scope of this work. The container uses the ubuntu nettools image created by Robertxie[3].

[1]https://pcengines.ch/apu2c4.htm
[2]https://www.linux-kvm.org/
[3]https://hub.docker.com/r/robertxie/ubuntu-nettools/

4.3.2.3 Database

The database stores different versions of the VNF. When a VNF update is available, the database is updated with the new version. It runs as a Docker container, with an image similar to the one used by the source node, which runs a simple python Hypertext Transfer Protocol (HTTP) server[4].

4.3.2.4 Sink node

The sink node is a Docker container instantiated with the same image as the source node. This container does not fit any purpose other than receiving packets to test the VNF.

4.3.2.5 VNF

The VNF represents the service request by the corporation and is hosted by the ISPand/or CCSP. For proof-of-concept purposes, a vFW will be used and evaluated over different virtualization techniques. In this line, the vFW will be instantiated as a Docker container and as an IncludeOS unikernel. Regarding to the Docker vFW, it is an Alpine Docker container[5] with the iptables package[6]. The vFW filters the packets according to the iptables rules specified in a bash file. When the Docker vFW is instantiated, it downloads the current-state file from the database, launching the container with the last updated version. The CCSP provides secure 3rd party interfaces to the corporation, allowing the latter to provide new versions of its VNFs, without the former tampering with them. However, such interfaces are out-of-scope from this work. Concerning to the IncludeOS vFW, it is a single-tasking OS [260], compiled into a binary file via a C++ code where the firewall rules are programmed. The hypervisor, before launching the unikernel VM with QEMU (a generic and Open source machine emulator and virtualizer)[7], downloads the latest updated binary to the database container, ensuring that the unikernel is running with the latest policies applied.

4.3.2.6 Open vSwitch bridges

In our proposal, the network bridges ensure communication between services, while seamlessly redirecting the traffic to the required VNF. In order to enable the on-the-fly

[4]https://docs.python.org/3/library/http.server.html
[5]https://hub.docker.com/_ /alpine/
[6]https://pkgs.alpinelinux.org/package/edge/main/x86/iptables
[7]https://www.qemu.org/

data-path updates, we used the Open vSwitch (OvS) [8] (version 2.10.90), allowing flow-based rules to be applied to the bridges flow tables via the OpenFlow protocol.

4.3.2.7 SDN controller

The SDN controller manages and controls OvS bridges using the OpenFlow, by implementing therein flow-based rules. This allows the controller to dynamically redirect the network traffic from one VNF to another in a failure or update event. Finally, Ryu[9] was used as the SDN controller, and an SDN application was developed to run along Ryu to allow the traffic redirection upon network triggers (explained in next section).

4.4 Design and Development: Recovery Mechanisms

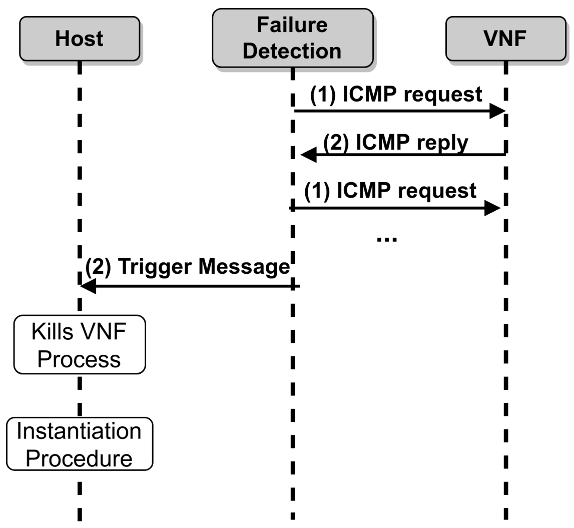
Our framework is supported by a recovery mechanism in order to achieve a near-zero downtime. In this line, our recovery mechanism can be divided into 3 steps: 1) failure detection; 2) VNF re-instantiation; and 3) data-path update. While Fig. 4.3a depicts the failure detection and re-instantiation mechanisms, Fig 4.3b shows the sequence messages of the data-path update.

4.4.1 Failure detection and VNF re-instantiation

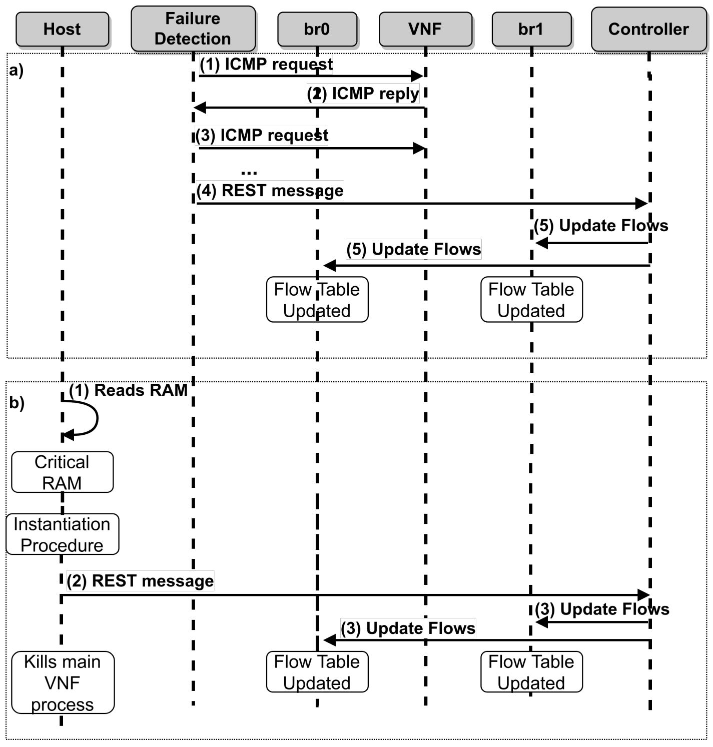
As the ISP and/or CCSP need to cope with the possibility of software failures, a failure detection mechanism was developed for verifying the status of operation of the NF, in this case, the firewall. This mechanism is based on the heartbeat software developed by the Linux-HA (High-Availability Linux)[10].

Fig. 4.3a illustrates the failure detection procedure, which is described as follows. The failure mechanism periodically sends Internet Control Message Protocol (ICMP) requests (1) (every $100ms$), and if the VNF responds by sending an ICMP reply (2), the failure mechanism does nothing and proceeds to keep sending ICMP requests (3). When the VNF fails to respond within a $50ms$ time-frame, a trigger is sent to the host (4). Note that both the ICMP request periodicity and the time-frame for response were chosen for proof-of-concept evaluations and different values can be used in order to optimize the mechanism. After receiving the trigger, the host kills the processes of the VNF to enforce the re-instantiation procedure. The virtualization technology (i.e., unikernel or container) in which the VNF (i.e, vFW) is going to be re-instantiated is sent in the trigger message. This failure mechanism is an illustrative approach with the sole purpose of triggering dynamic re-adjustments of the system. Other more refined

[8]OvS: https://www.openvswitch.org/
[9]Ryu: https://osrg.github.io/ryu/
[10]http://www.linux-ha.org/



(a) Fail detection and re-instantiation sequence messages.



(b) Data-path update via SDN interfaces.

Figure 4.3: Failure detection and recovery mechanism.

solutions can be used, including monitoring and operation assessment mechanisms more tailored towards commercial environments.

4.4.2 Data-path update

To minimize and achieve a near-zero downtime of the service, SDN was used for data-path updates on the fly. As such, our failure detection mechanism informs the SDN controller of failure and requests a flow redirection to the new VNF. To prevent disconnection between the source and sink nodes during a VNF fail detection event, one of two cases is required: *case a)* a backup VNF for resilience is already running from the beginning, or *case b)* a new VNF is dynamically and pre-emptively instantiated upon the trigger of a monitoring mechanism. For this last case, the monitor mechanism can explore KPIs, such as memory usage, which when reaches the memory limit increases the probability of failure. Fig. 4.3b depicts the high-level sequence message for a data-path update upon failure detection.

Case a) VNF already instantiated as backup:

1. The failure detection mechanism sends ICMP requests to the firewall;
2. The VNF replies to those requests;
3. Another ICMP request is sent;
4. Since the firewall does not respond to the ICMP request, meaning that an error occurred, a REST message is sent to the SDN controller;
5. After receiving the trigger message, the controller updates the flow tables of both OvS bridges, offloading the traffic to the backup VNF.

Case b) VNF dynamically instantiated:

1. The host reads the Random Access Memory (RAM) consumption of the main VNF; When the RAM consumption reaches critical values (80% of total RAM allocated to the VNF, although other values can be configured), the instantiation procedure of the backup VNF begins;
2. When the backup VNF is fully operational, a REST message is sent to the SDN controller;
3. After receiving the trigger message, the controller updates the flow tables of both OvS bridges, offloading the traffic to the backup VNF.

Finally, note that once the OvS bridges update their flow tables, the packets that were previously being sent to the main VNF, are now forwarded to the backup VNF, ensuring the reliability of the service.

4.5 Evaluation and Discussion

In this section, the proposed framework is evaluated in terms of throughput and delay. First, both virtualization technologies (i.e., Docker containers and IncludeOS unikernels) are compared in terms of image size, instantiation delay, traffic latency, and both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) throughput. Finally, the performance of the recovery mechanism was also evaluated in terms of throughput impact and delay. The experiments were run 10 times, with the average results being presented with a confidence interval of 95%.

4.5.1 VNF image size

Table 4.1 compares the image size of the two VNFs. It can be seen that the Docker FW is approximately 5 times larger than the IncludeOS FW. The reason for this difference has to do with the versatility of the VNF. The Docker FW runs as a tinyOS[11] while the unikernel has limited capabilities, having the sole purpose of running the C++ code compiled into the image. It should be noted that the actual size of the firewall program is also different for both implementations. While the Docker VNF uses the iptables package ($1.54 MBytes$), the IncludeOS FW uses $0.5 Mbytes$, since the unikernel needs a minimum of $2.7 MBytes$ to be able to run. As for memory, both implementations have 256 MB of RAM allocated.

4.5.2 Instantiation delay

For measuring the instantiation of both VNFs, it was considered the time it took from the launch command until the firewall was ready with all policies implemented. The time was measured immediately before the instantiation command and immediately after the last rule was applied, measured in milliseconds (ms) using the Unix epoch time[12]. Table 4.1 shows the instantiation time for both Docker FW and IncludeOS FW. Here, Docker was 44% faster than IncludeOS, mainly because containers do not contain a guest OS, decreasing the boot-time and creation delay.

4.5.3 Latency

For this evaluation, the latency refers to the Round Trip Time (RTT) of a packet. This test intends to evaluate how the VNF affects the latency in an end-to-end connection and was measured using the ping tool from the Network Tools Package[13]. The experiment

[11]TinyOS: http://www.tinyos.net
[12]Unix epoch time: https://www.epochconverter.com/
[13]https://network-tools.com/

was run 10 times from both the source node and from the sink node, with results being presented in Table 4.1.

From Table 4.1, the latency of the end-to-end connection when the firewall was instantiated in an unikernel was approximately $0.79ms$, as for the Docker instance, the end-to-end latency was $0.27ms$, almost 3 times less than unikernel. It can be deducted that this latency difference happens due to the time packets were being processed by the firewall since the ICMP packets travel the same path in both implementations.

Table 4.1: VNFs metrics for both virtualization technologies.

VNF	Size (MBytes)	Instantiation delay(s)	Traffic latency (ms)
Docker Firewall	16.4	3.02 ± 0.05	0.27 ± 0.02
IncludeOS Firewall	3.2	4.44 ± 0.10	0.79 ± 0.03

4.5.4 Traffic Traversing VNF

Three different types of traffic were evaluated for each VNF. The first was TCP traffic and the other two were UDP traffic - $100Mbps$ target bandwidth and $1Gbps$ target bandwidth. These tests were performed for 10 seconds using the iPerf3 tool[14]. The iPerf server was started on port 80 of the sink container and the client initiated the connection from the source node container with a buffer length of $1400Bytes$.

4.5.4.1 TCP Traffic

Fig. 4.4 shows the results after executing the iPerf 20 times for each virtualization technology using TCP packets. The Docker FW obtained better performance in both metrics, with an average throughput of $120Mbps$ compared to $100Mbp$ from the IncludeOS. The container VNF did not register TCP segments retransmissions, opposed to the unikernel where an average of 1.10 TCP segments was retransmitted, which translates into 0.013% of the total TCP packets sent. Notwithstanding, the small percentage of retransmitted packets, this difference can be justified by the latency delay introduced by the IncludeOS FW.

[14]iPerf3: https://iperf.fr/

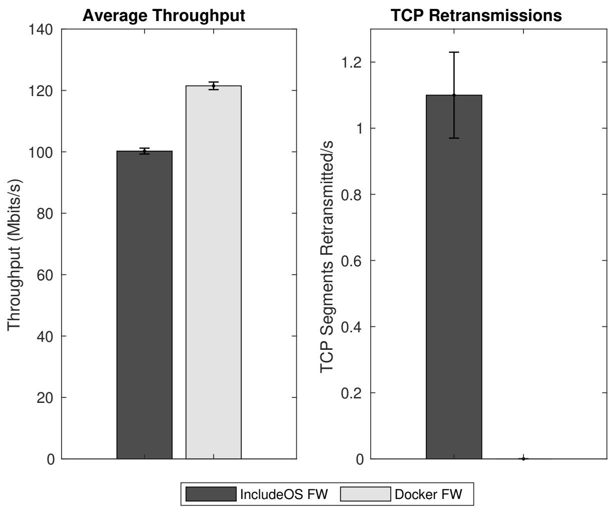


Figure 4.4: TCP Performance

4.5.4.2 UDP Traffic

Fig. 4.5 shows the results after performing the iPerf 20 times for each virtualization technology using UDP packets with both target bandwidths - 100*Mbps* and 1*Gbps*.

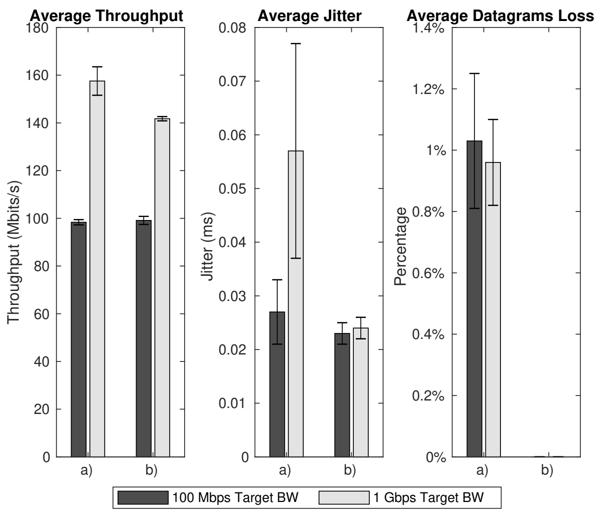


Figure 4.5: UDP Performance: a) IncludeOS FW; b) Docker FW

Contrary to the TCP performance, the IncludeOS FW had approximately the same performance as the Docker FW with 100*Mbps* target bandwidth, and it only stayed behind in the datagram loss since the container did not lose any datagrams.

When the target bandwidth was set to 1*Gbps* on the iPerf client, the performance of the unikernel improved in comparison to the container, achieving 160*Mbps* of throughput, which is 20*Mbps* better than the container.

In both technologies, the target bandwidth of $1Gbps$ was not reached due to the size of the UDP packets ($1400Bytes$). In contrast, the average Jitter in the Docker FW stayed constant and in the unikernel case, it doubled ($0.06ms$). The datagram loss stayed equal from the previous setup - 1% for IncludeOS and zero losses for Docker. These losses in the unikernel occurred for the same reason as the TCP retransmissions, namely the delay introduced by the IncludeOS FW.

Finally, for both virtualization technologies, the fact that both the host and sink node were instantiated in the same system board could impact the observed throughputs, limiting the maximum throughput by edging the computational power of the system board used as a fog node.

4.5.5 Traffic Traversing with VNF Failure

To evaluate the reliability mechanism proposed in Section 4.4, the same iPerf (a tool for network performance measurement and tuning) was used. For the following experiments, a failure was induced at Time = 3s of the experiment running time. The results with TCP traffic are presented in Fig. 4.6.

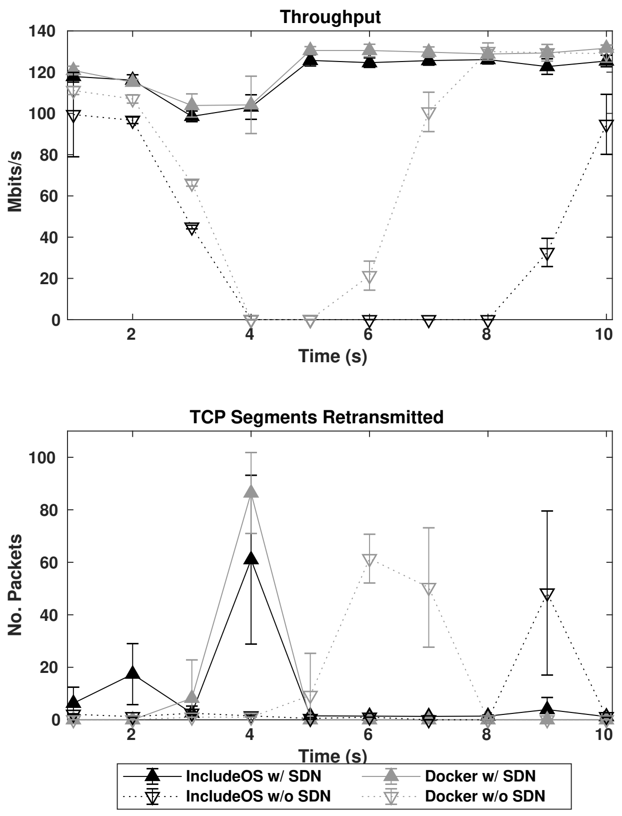


Figure 4.6: TCP performance with SDN recovery mechanism.

It is noted that the SDN recovery mechanism significantly improved the service as there was only a slight decrease in throughput when the flows were updated in the OvS bridges (this is evident at Time = 4s). When the same test was performed without the SDN recovery mechanism, the throughput dropped to zero when the failure was induced and only returned to baseline values when the re-instantiation occurred (Time = 7s for Docker FW and Time = 10s for IncludeOS FW).

Similarly to the previous test, when the iPerf was performed with UDP packets, throughput dropped to zero when the SDN recovery mechanism was not active and it was only restored at Time = 7s for the Docker FW and at Time = 10s for the IncludeOS FW. This occurred in both target bandwidths (100$Mbps$ and 1$Gbps$).

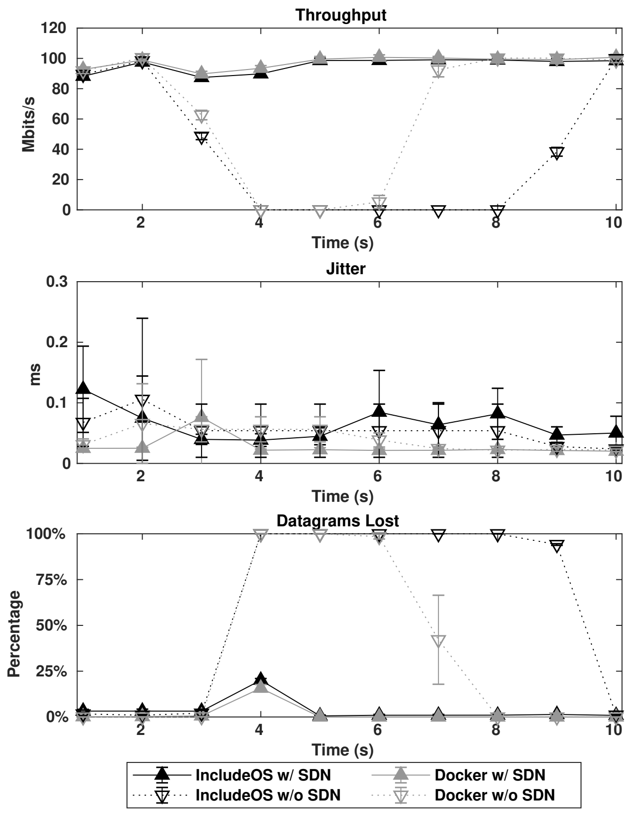


Figure 4.7: UDP Performance with SDN recovery mechanism for 100Mbps.

When the SDN recovery mechanism was active, for 100$Mbps$ target bandwidth - Fig. 4.7 - the redirection of traffic to the backup VNF was almost unnoticed, with only a slight increase in datagram loss seen at Time = 4s. As for UDP traffic with 1$Gbps$ target bandwidth - Fig. 4.8 - the redirection of traffic to the backup FW is clear at Time = 4s, where a slight decrease of throughput was noticeable as well as roughly 25% of UDP datagrams were lost. Note that the above experiments were performed with two VNFs running in parallel. To evaluate the difference of performance when the backup VNF is dynamically instantiated, an iPerf configured with UDP packets with 1$Gbps$ target bandwidth was performed during 25 seconds.

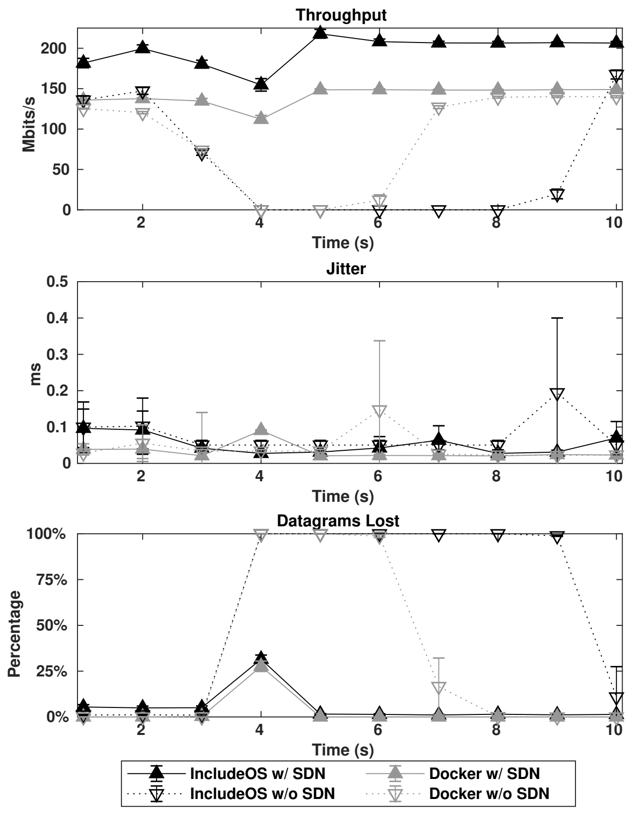


Figure 4.8: UDP performance with SDN recovery mechanism for 1Gbps.

The primary VNF was instantiated as an IncludeOS unikernel and the backup VNF instantiated as a Docker container since previous experiments showed that it instantiates roughly 2 seconds faster than the unikernel (Table 4.1). Fig. 4.9 presents such results.

At Time = 10s the primary VNF reached a critical point where 80% of the allocated memory was being used, which triggered the instantiation of the backup VNF. The primary VNF crashed at Time = 15s and the OvS flow tables were updated at that time. Even though that there was a slight decrease in throughput in comparison to when two instances were running in parallel, there is not a shattering difference for these proof of concept architecture. As for datagram losses, both implementations lost roughly the same amount of datagrams. As depicted in Fig. 4.7 , Fig. 4.8, and Fig. 4.9, jitter had large variations due to changes in routes and congestion.

Due to the nature of the data-center environment, where in this case resources are restricted, one could argue that the difference in throughput is acceptable, as the backup VNF is only instantiated when it is strictly necessary.

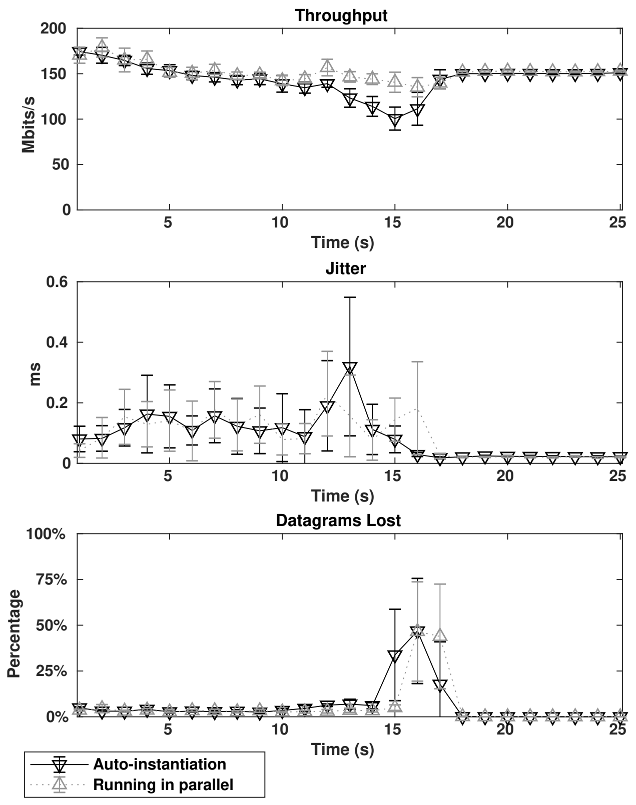


Figure 4.9: SDN recovery mechanism with dynamic instantiation

4.6 In Summary

This work presented a comparison of performance between two distinct types of virtualization platforms - unikernels and containers - when instantiated as a VNF with intelligent recovery strategies. Failure detection and a recovery mechanism were developed, with SDN being used to update the data path to mitigate the downtime of the VNF in a failure event, and consequently improving the reliability of services instantiated as VNFs. In this line, the proposed mechanism was able to achieve near-zero downtime for the evaluated service.

Results showcased that VNFs instantiated as Docker containers, have better performance (by lowering end-to-end latencies and instantiation time) when compared to VNFs instantiated as IncludeOS unikernels. In terms of throughput impact, containers perform slightly better than the unikernel for TCP traffic. Concerning UDP traffic up to $100Mbps$ of bandwidth, both virtualization techniques presented the same throughput. Notwithstanding, when stressing the VNF with $1Gbps$ of UDP traffic bandwidth, the IncludeOS unikernel presented an increase of 11% over Docker containers. Finally, regarding TCP retransmissions and UDP datagram losses, Docker containers performed better than IncludeOS unikernels. Furthermore, in Section 6.2.2, we briefly discussed and outlined the directions for future research regarding critical and reliable communications in software-defined and virtualized networks.

Chapter 5

Auto-scaling Testing

Chapter Outline: Recently, the OSM-MANO platform has attracted significant interest from academia and industry, potentially providing NFV management and orchestration solutions for ETSI compliant NFV implementations. OSM-MANO has a built-in Monitoring Module (MON), a Policy-management Module (POL), and a Prometheus time-series database for metrics collection, which are the main components of ensuring Service Assurance in OSM-MANO. In this work, auto-scaling testing is performed with OSM-MANO after instantiating network service using the OSM-MANO community-developed VNFs and the Grafana tool for data visualization. The Prometheus database collects the metrics from the MON exporter, and scaling actions are defined in POL. Finally, we performed auto-scaling testing and demonstrated the results using the Grafana dashboard. The results showed that the auto-scaling policy was automated successfully after the Virtual Infrastructure Manager (VIM) and VNF metrics triggered threshold violation alarms (already defined) and scaling actions were performed successfully as defined in POL.

5.1 NFV, OpenStack, and OSM-MANO Overview

NFV is considered a promising technology to decouple network functions from the underlying infrastructure, allowing for software-based deployment of network functions directly onto commodity hardware. In the NFV approach, scaling is the ability to extend or reduce the allocation of resources to VNFs. Scaling-in is performed to release unused resources, while scaling-out is performed to handle added loads. In this regard, auto-scaling refers to the automatic scaling of VNFs to enhance resource utilization and reduce operating costs [261].

The auto-scaling concept originated in Cloud Computing. Scaling is a key function of VNFs lifecycle management and auto-scaling enables the communications providers to automatically resize network services that are composed of network functions according to the network conditions (traffic load) [262]. Auto-scaling can accommodate more requests as compared to the legacy monolithic deployments that do not support auto-scaling features [263].

ETSI proposed an architecture for NFV that has three main parts [216]: NFVI, VNFs, and NFV-MANO as shown in Fig. 5.1. The OSM-MANO focuses on the NFV-MANO stack, as depicted in Fig. 5.1. The NFV orchestrator, VNF manager, and VIM are part of the NFV-MANO stack. VIM provides lifecycle management of virtualized resources, the VNF manager provides lifecycle management of VNFs directly or through their own EMS, and the NFV orchestrator provides general resource orchestration and an end-to-end service lifecycle, including multiple VNFs, while also supporting the interaction with both generic and vendor-specific VNF managers. The details of ETSI-NFV functional blocks are provided in Chapter 2.

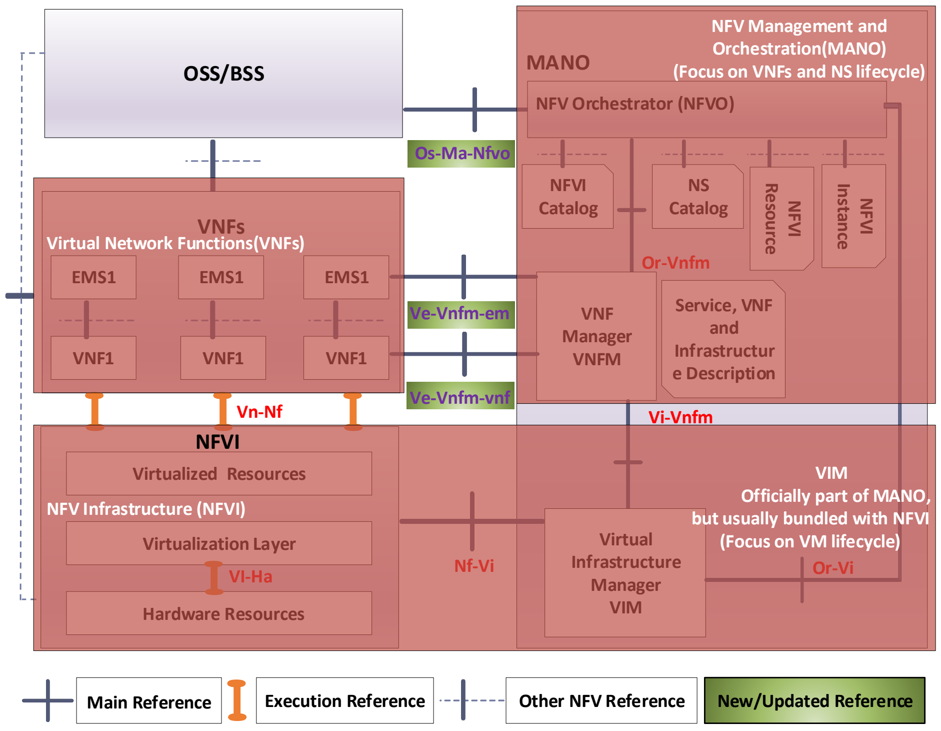


Figure 5.1: ETSI NFV Architecture and OSM-MANO Placement

OpenStack is a cloud platform offering abstracted network, storage, and computing services. The OSM-MANO OpenStack platform is used as a VIM. Other Cloud Computing platforms, such as Amazon Web Services (AWS) or VMware Cloud, are also supported by OSM-MANO. For VNFs to interact across different platforms, they must be defined in a standardized and simple way. OSM-MANO provides the possibility to define VNFs in a unified catalog through descriptor files, ensuring the interoperability of the VNFs across different networks. As mentioned earlier, VIM is already covered by OpenStack distribution, and, in practice, the NFV-MANO part focuses on the VNF manager and orchestrator.

5.2 ETSI Compliant NFV Management and Orchestration Frameworks

In this section, we provide a brief overview and comparisons of the state-of-the-art platforms offering management and orchestration solutions for 5G and beyond networks [264], [265]. Popular platforms that provide NFV-MANO ETSI compliant implementation are as follows:

5.2.1 OSM-MANO

OSM-MANO is a collaborative Open source project hosted by ETSI to develop NFV management and the orchestration software stack aligned with ETSI NFV information models to meet the commercial deployment of NFV networks [227]. Apart from providing the management and orchestration (MANO) component to the NFV, OSM-MANO can be used to facilitate the management, design, and development of VNFs and network services. The OSM-MANO solution is quite closely associated with 5G, and OSM-MANO can accelerate with the implementation of 5G. The OSM-MANO service orchestrator provides VNFs and network service lifecycle management as well as supporting YANG data modeling language. However, it can support a single administrative domain. Currently, the ETSI NFV-MANO community is releasing OSM-MANO eleven version (the most recent version verify to date) [266].

5.2.2 ONAP

The ONAP is led by the Linux Foundation [267]. ONAP is a result of merging two Open source management and orchestration projects (OPEN-O [268] and OpenCOMP [269]). The ONAP platform implements a unified architecture for creating, orchestrating, monitoring, and managing physical and virtual network functions. ONAP is supported by the largest network operator and cloud technology providers. ONAP supports Topology and Orchestration Specification for Cloud Applications (TOSCA) and YANG data modeling languages and can be integrated into VIM, the Virtual Network Function Manager (VNFM), and the SDN controllers as well as other legacy devices, which leads to cost savings. ONAP supports multiple administrative domains. Currently, the ONAP community is releasing Honolulu (8.0), the eighth release of the Linux Foundation ONAP project (the most recent version to date).

5.2.3 OPEN-BATON

Open-Baton project is led by Fraunhofer Fokus Institute and the Technical University of Berlin. The Open-Baton is an Open source implementation of NFV Orchestrator (NFVO)compliant with ETSI NFV-MANO and TOSCA model specifications [270]. It is a vendor-independent platform designed to provide extensible and customized network services across different vendor-specific networks (enable interoperability between different vendor-specific solutions). Open-Baton can orchestrate across multiple administrative domains and supports different cloud infrastructure implementation (e.g. OpenStack, Docker containers, and Linux containers). However, the Open-Baton community is quite small which limits the maintainability of the project.

5.2.4 OPENSTACK TACKER

Tacker is an Open source project initiated by the OpenStack community to enable NFV orchestration for the OpenStack platform based on ETSI NFV-MANO architectural framework [271]. In Tacker generic VNFM and NFVO can be built to deploy and manage VNFs and network services using cloud infrastructure. Tacker supports auto-scaling and TOSCA NFV profile Using a heat translator and allows mapping of SFC. Tacker provides limited interoperability with other VIMs because tacker components are directly integrated to support OpenStack. Tacker supports VNFs operation and deployment only with a single administrative domain.

5.2.5 X-MANO

The X-MANO project [272] is initiated to deal with the challenges of cross-domain NFV orchestration includes confidentiality, multi-domain lifecycle management, and consistent information exchange between standard interfaces. X-MANO supports VNFs end-to-end network service delivery across multiple administrative and technological domains without exposing details of third-party implementations. X-MANO introduces the concept of programmable network service to address multi-domain lifecycle management requirements and supports domain-specific scripting language to allow developers to a flexible multi-domain network service descriptor to build and deploy customized network services across multiple domains.

In Table 5.1, we have provided a brief comparison of the selected projects focusing on NFV-MANO stack compliance with the ETSI NFV reference framework. Most of the NFV management and orchestration solutions offered have followed ETSI NFV-MANO

Table 5.1: Selected Projects Focusing on NFV-MANO Stack Compliance with the ETSI NFV Reference Framework.

NFV-MANO PLATFORM	NFV-MANO FRAMEWORKS				
	Multi-domain	VNFM	VIM	NFVO	OSS/BSS
ONAP	Yes	Yes	Yes	Yes	Yes
OSM-MANO	No	Yes	Yes	Yes	No
OPEN BATON	No	Yes	Yes	Yes	No
TACKER	No	Yes	No	Yes	No
X-MANO	Yes	No	No	No	No

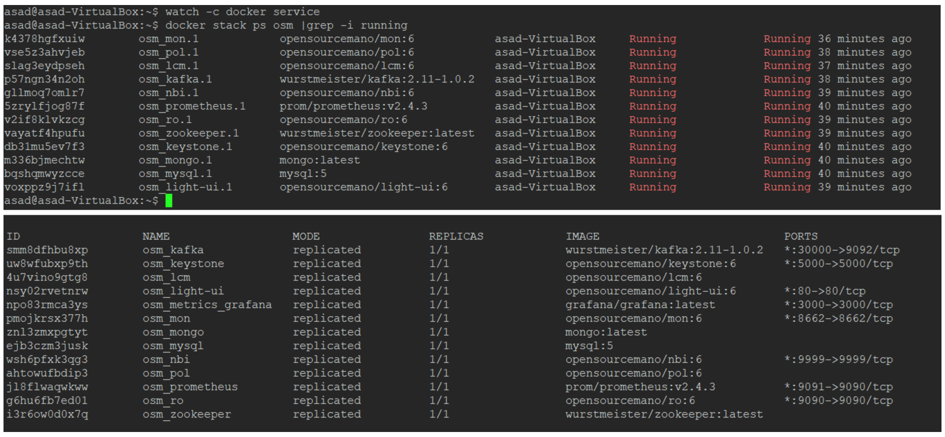
specification which means that the ETSI NFV-MANO reference framework is generally preferred. OSM-MANO and ONAP have attracted significant interest from industry and academia. On one hand, ONAP is progressed and ready for deployment in production environments [265] supported by a larger group of global service providers but one of the disadvantages of ONAP is deployment complexity. On the other hand, OSM-MANO is progressing fast and increasing support from global service providers and deployment is easy as compared to ONAP. ONAP and OSM-MANO orchestrator's performance could determine their adaptation in near future production environments.

5.3 Configuring to Develop with OSM-MANO

In OSM-MANO there are more than ten modules run in separate docker containers except for the juju (an Open source charmed Operator framework) controller which uses LXC. Filip et al.[273] studied and evaluated the performance comparison of containers and unikernel (lightweight virtualizations technologies). The details of the OSM-MANO modules are as follows [266]:

- kafka: Kafka bus is used for OSM-MANO communications and this module relies on the zookeeper module.
- zookeeper: This module is used by Kafka to manage clusters.
- nbi: The northbound interface of OSM-MANO. It relies on the mongo database, Kafka bus, and optionally uses the keystone for authentication.
- keystone: This module stores the users, projects, role permissions and it relies on the mysql module.
- lcm: This module provides LCM. For resource orchestration, it uses ro module and juju for configuration and relies on the mongo module.

- ro: This module provides resource orchestration, or VIM deployment, and relies on the mysql module.
- light-ui: This module provides a web user interface and communicates with nbi.
- mon: This module performs OSM-MANO monitoring and relies on mongo and mysql modules.
- mongo: This module provides a common non-relational database for OSM-MANO.
- mysql: This module provides a relational database server that is used for modules such as ro, keystone, mon, and pol.
- pol: Policy manager for OSM-MANO.
- prometheus: This module is used for monitoring.
- grafana: This module is used for data visualization.



```
asad@asad-VirtualBox:~$ watch -c docker service
asad@asad-VirtualBox:~$ docker stack ps osm |grep -i running
k4378hgfxuiw        osm_mon.1           opensourcemano/mon:6              asad-VirtualBox     Running             Running 36 minutes ago
vse5z3ahvjeb        osm_pol.1           opensourcemano/pol:6              asad-VirtualBox     Running             Running 38 minutes ago
slag3eydpseh        osm_lcm.1           opensourcemano/lcm:6              asad-VirtualBox     Running             Running 37 minutes ago
p57ngn34n2oh        osm_kafka.1         wurstmeister/kafka:2.11-1.0.2     asad-VirtualBox     Running             Running 38 minutes ago
gl1moq7omlr7        osm_nbi.1           opensourcemano/nbi:6              asad-VirtualBox     Running             Running 39 minutes ago
5zrylfjog87f        osm_prometheus.1    prom/prometheus:v2.4.3            asad-VirtualBox     Running             Running 40 minutes ago
v2if8klvkzcg        osm_ro.1            opensourcemano/ro:6               asad-VirtualBox     Running             Running 39 minutes ago
vayatf4hpufu        osm_zookeeper.1     wurstmeister/zookeeper:latest     asad-VirtualBox     Running             Running 39 minutes ago
db31mu5ev7f3        osm_keystone.1      opensourcemano/keystone:6         asad-VirtualBox     Running             Running 40 minutes ago
m336bjmechtw        osm_mongo.1         mongo:latest                      asad-VirtualBox     Running             Running 40 minutes ago
bqshqmwyzcce        osm_mysql.1         mysql:5                           asad-VirtualBox     Running             Running 40 minutes ago
voxppz9j7ifl        osm_light-ui.1      opensourcemano/light-ui:6         asad-VirtualBox     Running             Running 39 minutes ago
asad@asad-VirtualBox:~$
```

```
ID                  NAME                  MODE                REPLICAS            IMAGE                             PORTS
smm8dfhbu8xp        osm_kafka             replicated          1/1                 wurstmeister/kafka:2.11-1.0.2     *:30000->9092/tcp
uw8wfubxp9th        osm_keystone          replicated          1/1                 opensourcemano/keystone:6         *:5000->5000/tcp
4u7vino9gtg8        osm_lcm               replicated          1/1                 opensourcemano/lcm:6
nsy02rvetnrw        osm_light-ui          replicated          1/1                 opensourcemano/light-ui:6         *:80->80/tcp
npo83rmca3ys        osm_metrics_grafana   replicated          1/1                 grafana/grafana:latest            *:3000->3000/tcp
pmojkrsx377h        osm_mon               replicated          1/1                 opensourcemano/mon:6              *:8662->8662/tcp
znl3zmxpgtyt        osm_mongo             replicated          1/1                 mongo:latest
ejb3czm3jusk        osm_mysql             replicated          1/1                 mysql:5
wsh6pfxk3qg3        osm_nbi               replicated          1/1                 opensourcemano/nbi:6              *:9999->9999/tcp
ahtowufbdip3        osm_pol               replicated          1/1                 opensourcemano/pol:6
jl8flwaqwkww        osm_prometheus        replicated          1/1                 prom/prometheus:v2.4.3            *:9091->9090/tcp
g6hu6fb7ed01        osm_ro                replicated          1/1                 opensourcemano/ro:6               *:9090->9090/tcp
i3r6ow0d0x7q        osm_zookeeper         replicated          1/1                 wurstmeister/zookeeper:latest
```

Figure 5.2: Running OSM-MANO Modules

In order to successfully configure OSM-MANO, it is important to visualize that all OSM-MANO modules are up and running (by using the watch docker service command) as shown in Fig. 5.2.

5.3.1 OSM-MANO Fault-Management: Service Assurance Components

There are three main modules of OSM-MANO Service Assurance and, multiple optional tools for monitoring and data visualization as shown in Fig. 5.3. The details are as follows.

- MON: MON module is responsible for collecting Virtual Deployment Unit (VDU), VNF metrics, and infrastructure metrics of external components including VIMs, WAN Infrastructure Managers (WIMs), and SDN controllers. It stores metrics in the Prometheus Time Series Data Base (TSDB). Furthermore, it manages and evaluates alarms define in VNF/VDU descriptor. MON sends notifications through Kafka bus when alarms are triggered.

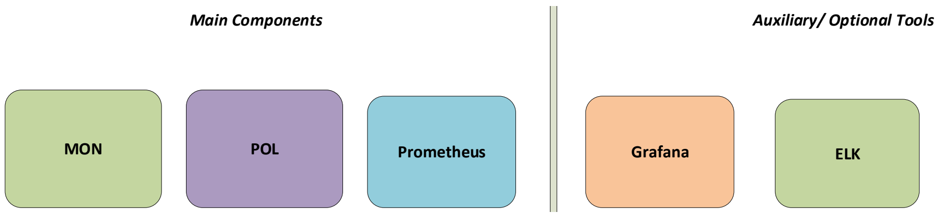
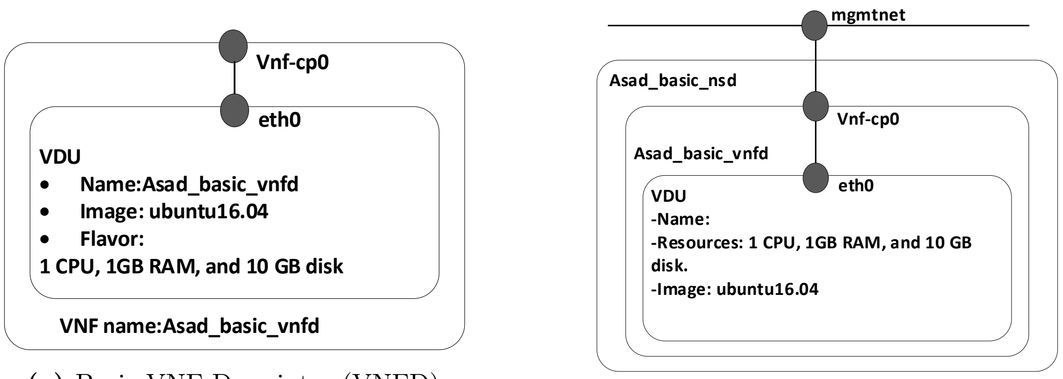


Figure 5.3: OSM-MANO Service Assurance Components

- POL: POL module is responsible for configuring alarms and auto-scaling groups for VNFs. Furthermore, it listens to MON for corresponding VNF alarms and scaling policies defined in VNF Descriptor (VNFD). When a network service is created it check the corresponding VNF descriptors to evaluate policies and execute the corresponding decision.
- Prometheus: Prometheus is OSM-MANO, TSDB, collected metrics are stored in Prometheus. It can be easily integrated with data visualization tools to show interactive graphs for any metric. However, at present it doesn't support multitenancy; other projects need to be explored.

  Besides these main modules, OSM-MANO provides support for Grafana and ELK (ELK is the acronym for three Open source projects: Elasticsearch, Logstash, and Kibana) as auxiliary Open source tools to visualize data with charts and graphs. These tools allow end-users to create complex dashboards and easily show interactive graphs for any metric.

5.3.2 Creating Basic Virtual Network Function and Network Service Descriptors



(a) Basic VNF Descriptor (VNFD)

(b) Basic Network Service Descriptor (NSD)

Figure 5.4: OSM-MANO: Network Service Creation

This section provides an overview of creating basic VNFD and NS Descriptor (NSD) descriptors in OSM-MANO.

In order to deploy the network service using the CLI in OSM-MANO, network functions are defined with the required specifications in the VNF descriptor compatible with OSM-MANO. After creating and validating a package for each descriptor the network service is instantiated. Descriptors are defined in Yet Another Markup Language (YAML) programming language, information model tree representation for VNFD is also provided by the OSM-MANO community [274]. Code 5.1 and 5.2 depict the programing structure to create basic VNF and NS descriptors in OSM-MANO. The graphical representation of VNF and NS descriptors is illustrated in Fig. 5.4. However, it is possible to create customized VNF and NS descriptors. Furthermore, Multi VDU in VNF can be defined in a single VNF.

Code 5.1: Creating Basic VNF Descriptor (VNFD)

```
\\ Program name: VNFD.yaml
/* Creating basic VNFD/
vnfd:
- id: Asad_basic_vnfd
 name: Asad_basic_vnfd
 ...
 mgmt-interface:
 cp: vnf-cp0
 vdu:
 - id: Asad_basic_vnfd-VM
 name: Asad_basic_vnfd-VM
 vm-flavor:
 vcpu-count: 1
 memory-mb: 1024
 storage-gb: 10
 image: ubuntu16.04
 interface:
 - name: eth0
 virtual-interface:
 type: VIRTIO
 ...
 external-connection-point-ref: vnf-cp0
 connection-point:
 - name: vnf-cp0
```

Code 5.2: Creating Basic NS Descriptor (NSD)

```
  \\ Program name: VNFD.yaml
  /* Creating basic NS descriptor/
nsd:
- id: Asad_basic_nsd
 name: Asad_basic_nsd
 ...
 constituent-vnfd:
 - member-vnf-index: 1
 vnfd-id-ref: Asad_basic_vnfd
 vld:
 - id: mgmtnet
 name: mgmtnet
 type: ELAN
 mgmt-network: true
 vnfd-connection-point-ref:
 - member-vnf-index-ref: 1
 vnfd-connection-point-ref: vnf-cp0
 vnfd-id-ref: Asad_basic_vnfd
```

5.4 Testing Environment Set-Up and Results

We tested auto-scaling in OSM-MANO with two VMs installed on a single laptop [275]. Detail specifications are as follows: System details 8 CPU's, 16 GB RAM, Processor: Intel(R) Core (TM) i7-8550U CPU @ 1.80GHz (8 CPUs) TOSHIBA, TECRA A50-E, VM1: DevStack, and VM2: OSM-MANO Version 6. The OSM-MANO interaction with VIM and VNFs connectivity is illustrated in Fig. 5.5. To quickly bring up a complete OpenStack environment in a single machine DevStack VIM is configured to test VNFs auto-scaling using OSM-MANO.

When VNFD and NSD packages are created, it is possible to import them using OSM-MANO Graphical User Interface (GUI) and deploy network service using OSM-MANO GUI. Fig. 5.6 shows the created network service VNFD and NSD through user-friendly GUI. OSM-MANO allows the flexibility to create VNFD and NSD descriptors through both GUI and CLI.

Generally, scaling is initiated to increase the capacity of the VNFs in advance based on demand forecasting or in order to meet the SLAs requirements for an end-to-end service level management system [276]. Scale-out/in (horizontal scaling) is the ability

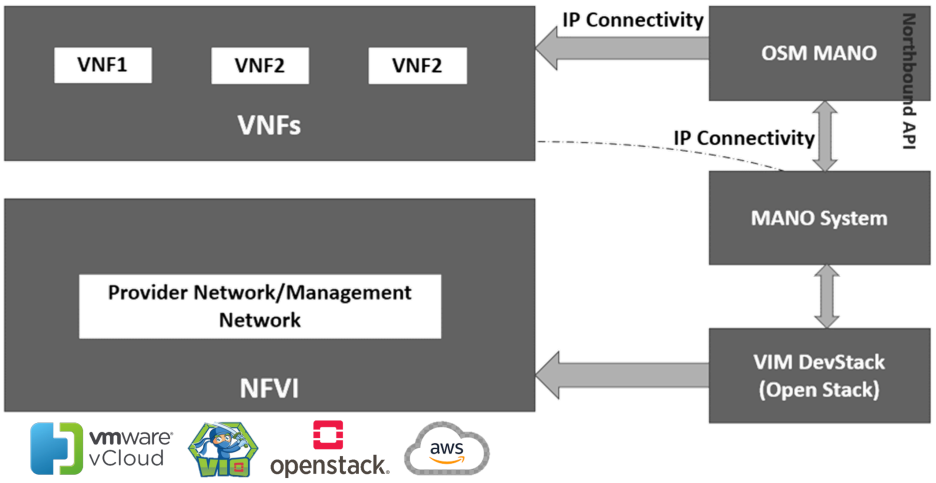
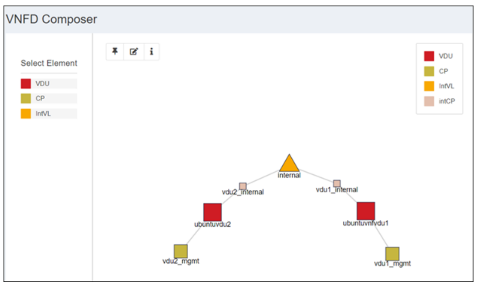
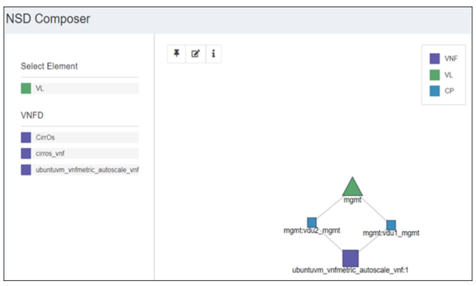


Figure 5.5: OSM-MANO Interaction With VIM



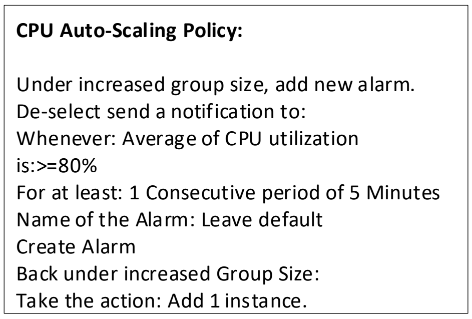
(a) Graphical Virtual Network Function Descriptor



(b) Graphical Network Service Descriptor

Figure 5.6: OSM-MANO: VNFD and NSD GUI Composer

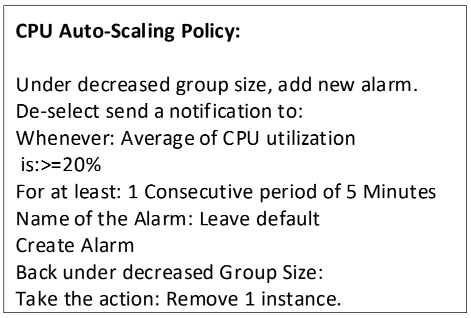
to scale by adding and removing instances in a virtualized network comprised of VNFs. A typical example of scale-out/in policy is depicted in Fig. 5.7.



CPU Auto-Scaling Policy:

Under increased group size, add new alarm.
De-select send a notification to:
Whenever: Average of CPU utilization
is:>=80%
For at least: 1 Consecutive period of 5 Minutes
Name of the Alarm: Leave default
Create Alarm
Back under increased Group Size:
Take the action: Add 1 instance.

(a) Scale-out



CPU Auto-Scaling Policy:

Under decreased group size, add new alarm.
De-select send a notification to:
Whenever: Average of CPU utilization
is:>=20%
For at least: 1 Consecutive period of 5 Minutes
Name of the Alarm: Leave default
Create Alarm
Back under decreased Group Size:
Take the action: Remove 1 instance.

(b) Scale-in

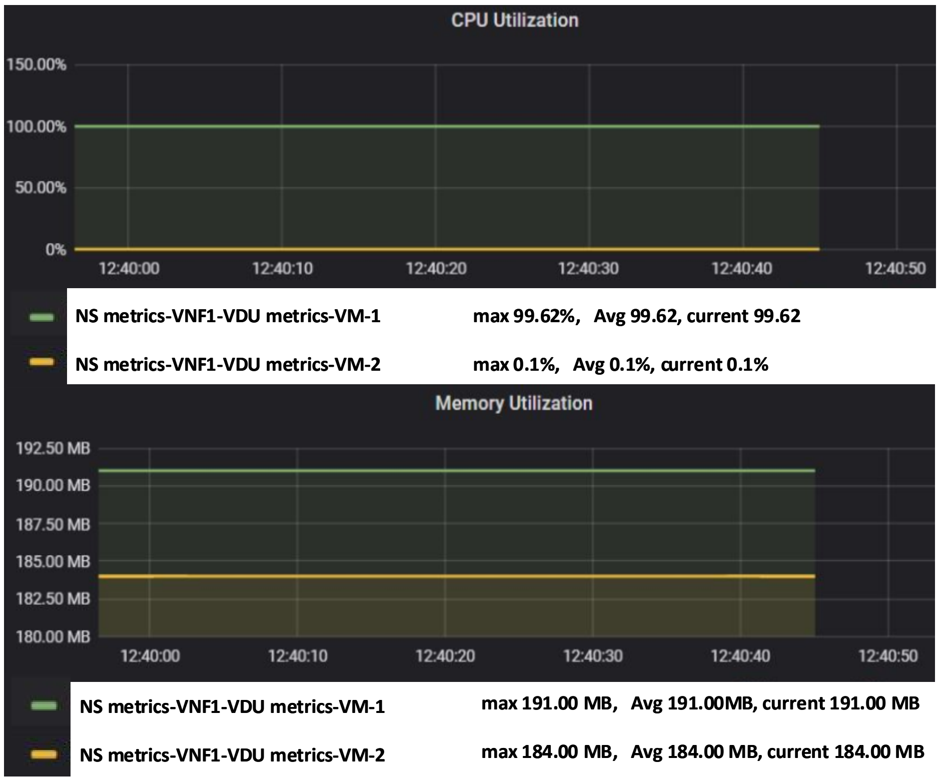
Figure 5.7: Defining Auto-Scaling Policy

Our proposed auto-scaling flow in OSM-MANO works as follows: First of all, threshold and scaling actions are defined at the VNF descriptor. POL creates alarms through MON. MON configures the alarms locally and starts its evaluation by default every 30 seconds. When a defined metric threshold is crossed, MON puts a notification in the bus. Scaling actions are triggered based on the received notification. Lifecycle Management (LCM) module receives scaling requests and proceeds with instantiation.

Using the auto-scaling feature, service instances are automatically added or removed based on the defined policy of scale-out/in, thus new VNFs are instantiated on the fly, this provides the ease of creating and delivering new network services effectively.

Each NFV instance consists of only one VDU and has a Cirros operating system (as a test image). Each VDU is composed of 1 CPU, 1 hard disk of 10 GB, and 2048 MB of RAM Memory. As already defined in our scaling policy when CPU or RAM usage reaches 80% of the total availability from the VM (after performing the stress tests) triggering alarm triggers for the OSM-MANO MON framework, which, in turn, initiated the instantiation of a new VNF. After the instantiation of the new VNF, the load can be distributed which will cause a decrease in CPU usage. As shown in Fig. 5.8, we performed auto-scaling testing and demonstrated the results using the Grafana dashboard. The results showed that the auto-scaling policy was automated successfully after the VIM and VNF metrics triggered threshold violation alarms (already defined) and scaling actions were performed successfully as defined in POL.

Auto-scaling can increase the number of VNFs during peak load and decrease capacity during idle time thus auto-scaling features can enhance performance as well as reduce network operating costs.



🟩	NS metrics-VNF1-VDU metrics-VM-1	max 99.62%, Avg 99.62, current 99.62
🟨	NS metrics-VNF1-VDU metrics-VM-2	max 0.1%, Avg 0.1%, current 0.1%

🟩	NS metrics-VNF1-VDU metrics-VM-1	max 191.00 MB, Avg 191.00MB, current 191.00 MB
🟨	NS metrics-VNF1-VDU metrics-VM-2	max 184.00 MB, Avg 184.00 MB, current 184.00 MB

Figure 5.8: Auto-Scaling using OSM-MANO

5.5 Experimental Lesson Learned and Research Gaps

We can summarize the main conclusions of our experimental work as follows:

- The initial installation and deployment of the DevStack (OpenStack) and OSM-MANO based setup in a single machine was challenging.
- Data-driven templates YAML for creating VNFD and NSD should be carefully configured (follow indentation rules) in order to run VNFD and NSD instances appropriately.
- VNFs in OSM-MANO should be developed using OSM-MANO standard procedure with a proper indication of interfaces and connection points.
- Currently auto-scaling in OSM-MANO cannot be initiated dynamically.
- New proposal for VNF indicators is underway now with using Simple Network Management Protocol (SNMP).
- Prometheus does not support multi-tenancy; other projects need to be explored.

Furthermore, in Section 6.2.3, we briefly discussed and outlined the future research directions for auto-scaling testing using OSM-MANO.

CHAPTER 6

Conclusions and Future Directions

Chapter Outline: This chapter concludes by highlighting future research directions in terms of reliability covering aspects of fault-tolerance and Service Assurance using SDN, NFV, and Cloud Computing based implementations.

6.1 Review of Achievements

This thesis explored NFV and SDN technologies with a touch of Cloud Computing for evolving software-defined and virtualized networks. The requirements, concepts, design goals, main architectural impairments, and state-of-the-art research efforts are discussed. Furthermore, we review fault-tolerance in the scope of SDN and envisage Service Assurance for 5G and B5G Hybrid Networks.

In a nutshell, this work explored and validated solutions in terms of reliability, fault-tolerance, and service assurance using SDN, NFV, and Cloud Computing based implementations to discover the feasibility and reliability of software-defined and virtualized networks. Furthermore, OSM-MANO mechanisms were integrated envisioning standardized management and orchestration of network resources and services. As presented in Chapter 1, this study contributes to national and intentional projects, as well to the research community through publications, scientific journals, public talks, and international conference proceedings.

As result, the thesis answers initial self-imposed questions as follows.

6.1.0.1 Fulfillment of Research Questions

Q1.What are the implementation challenges and architectural impairments of software-defined and virtualized networks key enablers and key technologies namely, SDN and NFV?

In this thesis, Chapter 2, first provides a detailed background of NFV, SDN, and Service Assurance for evolving 5G and B5G Hybrid Networks to establish a comprehensive understanding of the subject, ranging from the basics to more advanced topics. In addition to that, Chapter 2, addresses comprehensively the implementation challenges and architectural impairments of software-defined and virtualized networks key enablers and key technologies namely, SDN and NFV. Furthermore, Chapter 2, highlights the state-of-the-art research efforts, work in progress associated with evolving software-defined and virtualized networks key enablers and key technologies.

Q2. How Service Assurance would be devised to support evolving 5G and B5G Hybrid Networks, a mix of both physical and virtualized networks?

Next-generation communication networks (network functions) are expected to be implemented on Open source virtualized infrastructures instead of current proprietary-based implementations to support the 5G vision. Realizing this phenomenon a complex Hybrid networking environment (involving both legacy and virtual technologies) will co-exist. Hybrid Network Services will be based on SDN, Cloud, NFV, and legacy-based infrastructures. Therefore, Chapter 2 also identifies and envisages the Hybrid Networks environment and discusses the implementation of Service Assurance for 5G and Beyond 5G (B5G) Hybrid Networks their requirements, design goals, key considerations, and challenges (details are provided in Chapter 2 and Chapter 6).

Q3. Which virtualization technologies, namely containers and unikernels, envisioning the deployment to facilitate resilient communication networks in critical scenarios (5G environments)?

For reliable communications, Chapter 3, and Chapter 4 presents an overview of the design for the critical and reliable use case scenario with containers and unikernel VNFs based on virtualization technologies. Results are presented and evaluated using containers and unikernel virtualization technologies (VNFs performance for metrics such as instantiation times, latency, throughput, and others). Finally, the results showed that our developed failure detection mechanism and recovery mechanism achieve higher reliability and were able to ensure near-zero downtime when a failure occurs, showcasing the feasibility of the solution.

Q4. How to implement Service Assurance components and test auto-scaling using OSM-MANO to meet the network requirements (management and orchestration) of 5G?

In Chapter 5, Service Assurance components and auto-scaling were implemented and tested using OSM-MANO. VNFs were managed by NFV-MANO, allowing VNFs to be instantiated, managed, scaled in or out, and terminated when no longer required. Mechanisms were integrated envisioning standardized management and orchestration of network resources to deployment and maintain NFV services across the network. Finally, we performed auto-scaling testing and demonstrated the results using the Grafana dashboard. The results showed that the auto-scaling policy was automated successfully after the VIM and VNF metrics triggered threshold violation alarms (already defined) and scaling actions were performed successfully.

6.2 Discussion

In this section, we briefly discuss and outline the future research direction for Chapter 3: SDN Control Network, Chapter 4: Reliable 5G Networks, and Chapter 5: Auto-scaling Testing.

6.2.1 SDN Control Network

Some of the Fault-tolerant techniques adapted from the traditional networks are being applied to SDN but we believe that there is a need to develop new reliability and resiliency mechanisms for SDN. In SDN the control plane is programmable and this enables efficient network monitoring. Thus, this ensures the greater flexibility to develop a more efficient Fault-tolerant mechanism to mitigate network faults.

It is hoped that the concept of proposed hybrid SDN control network should fulfill the fault-tolerant requirements for future heterogeneous networks made up of OpenFlow and traditional switches as well as provide support to the important concept of network slicing in 5G and beyond 5G networks. Due to the policy-based network management in SDN, these evolving challenges potentially will be resolved through SDN in the near future. Moreover, to design and develop new fault-tolerant policies for SDN, different network management protocols used to monitor connectivity such as Loss of Signal (LOS), Bidirectional Forwarding Detection (BFD), and Link Layer Discovery Protocol (LLDP) must be explored, modified, and optimized to adapt the dynamics of the SDN and future network more efficiently. Moreover, software debugging may fit well together to enforce the correctness of flow rules.

We also believe that hybrid mode is a more appropriate solution to offer more efficient fault-tolerance in SDN control networks. One of the possible implementations is that all network devices are connected to the controller directly with a separate network and in the case of a broken communication link the self-activation of in-band control takes over to ensure proper operation of the network. However, the cost to maintain such a separate network is an issue but hybrid control offers a more resilient design that guarantees improved reliability in future SDN-based networks.

6.2.2 Reliable 5G Networks

As future work, more refined solutions including monitoring and operation assessment mechanisms tailored towards commercial environments can be implemented to further improve this type of service. Also, other types of VNFs, such as Domain Name Systems (DNSs), Content Delivery Networks (CDNs), and IDS can be compared between these two different virtualization technologies. Not only other VNFs can be tested, but different containers and unikernels can also be evaluated.

Another compliment to the proposed architecture is taking into account the scalability of the VNFs running in the data center. Since applications and services can have fluctuating resource requirements, depending on external factors such as time of day, load demand, and network conditions, it is crucial to prepare these critical and reliable VNFs with dynamic scalability features according to performance requirements.

6.2.3 Auto-scaling Testing

This work presented a comparison of the state-of-the-art ETSI compliant NFV management and orchestration frameworks that can provide management and orchestration solutions for the 5G and B5G Networks.

Results showcased that the auto-scaling policy was automated successfully after the VIM and VNF metrics triggered threshold violation alarms already set and scaling actions were performed successfully as defined in POL.

At present auto-scaling in OSM-MANO is initiated statically. For future work, auto-scaling in OSM-MANO must be initiated dynamically, and more refined solutions including monitoring tailored towards supporting NFV commercial environments can be implemented to further improve this type of service.

We aim to develop mechanisms to collect different VNFs metrics using OSM-MANO MON architecture as well as assess the reliability of VNFs according to the performance requirements for critical communication scenarios.

6.3 Future Research Directions

Finally, in this section, we overall conclude enumerating future directions for NFV, SDN fault-tolerance, and Hybrid Networks Service Assurance development.

6.3.1 NFV: Future Research Directions

Indeed, NFV has the potential to grow significantly and eventually transform traditional proprietary-based network appliances to non-proprietary and open-standard-based network and service deployments. This is going to shift the trend towards the softwarization of telecommunications systems [277]. The purpose is to move network management functionalities closer to the software development; this brings three main advantages 1) Eases the network management for administrators. 2) Enables administrators to run automatic automation in the fail-over scenario. 3) Facilitates service providers to add new services on the fly without worrying about hardware compatibility issues. However, to obtain these benefits, NFV must support intelligence and programmability, and network automation.

In this section, we outline the future directions for NFV development from the perspective of NFV role in intelligent programmable networks, including network programmability and automation, the softwarization of telecommunications systems, and finally integration of NFV with other technologies.

6.3.1.1 Network Softwarization

Network Softwarization (NetSoft) is a paradigm to enhanced network programmability which leads to enhanced automation. The term NetSoft refers to the networking industry transformation for designing, deploying, implementing, and maintaining network devices/ network elements through software programming. This enables flexibility to re-design network services to optimize cost and enable self-management capabilities to manage network infrastructure. Furthermore, the term "NetSoft" was first introduced at the academic conference Network softwarization in 2015, to include broader interests regarding NFV, SDN, network virtualization, MEC formerly known as Mobile Edge Computing, Cloud Computing, and IoT technologies [278].

To sum up, NFV architecture must be designed to support the softwarization of telecommunications systems, and we believe that this is an important area for future NFV-related research.

6.3.1.2 Integration of NFV with other Technologies

Another important research direction for NFV is the integration of NFV with other technologies. Over the past years, the integration of NFV with other technologies, such as SDN, Cloud Computing, and 5G has attracted significant attention from both the academic research community and industry. These technologies have been proposed to fulfill different demands of future networks. Similarly, some of the studies also identified IoT, ICN [1], [8], MEC [218], [279], [280] fog computing [279], [281] and Cloud Radio Access Network (C-RAN) [282], [283] as a use case for NFV and discussed it in the perspective of future NFV research direction.

However, several challenges arise when integrating these technologies with NFV. Firstly, the standardization of these emerging technologies is underway, and therefore, at this stage, a time-resilient integration is quite challenging. Secondly, NFV commercial deployment is still at its early stages (as identified and discussed in Section 2.2.3), hence integrating NFV with other new technologies is naturally difficult. Finally, the dynamic and multi-domain nature of future networks imposes new security risks and threats. Addressing the security aspect while integrating NFV with other technologies is very challenging, specifically to ensure data privacy and trustworthiness between different network domains. In order to get the full benefit of NFV, the main integration challenges must be addressed appropriately before deployment.

NFV integration with SDN and Cloud Computing is reasonably accepted due to the complementary features and distinctive approaches followed by each technology toward providing solutions to today's and future networks [284], [285]. For instance, NFV provides functions/service abstractions (i.e., virtualization of network functions) supported by ETSI [64], SDN provides network abstractions supported by ONF [286] and Cloud Computing provides computing abstraction (i.e., a shared pool of configurable storage resources) supported by the Distributed Management Task Force (DMTF) [287].

These technologies have distinct features and when combined can provide a technological platform that integrates and combine technological concepts to fulfill the needs of new business verticals (as handled in 5G networks). These three main technologies provide a complete abstraction solution for networking (network, computing, and storage). Therefore, the industry is looking forward to this integration and believes that the integration of these technologies will result in the creation of fully programmable networks [8].

SDN, NFV, and Cloud Computing technologies are complementary to each other but are independent and can be deployed alone or together. A combination of these technologies together in a network architecture is more desirable [15]. Moreover, integrating NFV with these technologies adds value (flexibility and agility) to telecommunications systems. Furthermore, it will offer freedom to CSPs to create and manage network services without worrying about vendor-specific networking device configurations. Thus, this integration of NFV with other technologies offers enhanced privileges to CSPs to set up network services almost effortlessly.

6.3.2 SDN Fault-tolerance: Future Research Directions

In this section, we outline future directions for SDN fault-tolerance development from the perspective of its role in future intelligent programmable networks. The term programmability refers to control the set of actions, rules, or enforcing a policy by software intelligently. The programmability empowers to utilize multi-vendor hardware/devices with enhanced flexibility. Moreover, it enables customized scripting through programming languages to facilitate network administrators to enforce policy-based configuration on network devices/functions through APIs. Therefore, programmability is a pre-requisite for enabling network automation (a practice in which software automatically configure and test network devices) in a communications network[9].

A programmable network is flexible and re-configurable because most of the protocol stacks are implemented in software. Therefore, network upgrades to replace or configure the network protocols are possible without the interruption of the network operations [288], [289].

The term Network softwarization refers to the "networking industry transformation for designing, deploying, implementing and maintaining network devices/network elements through software programming" [216].

6.3.2.1 Data Plane Programmability for Network Softwarization

A few research studies included in Table 2.6 [290] focused on SDN data plane fault-tolerance using traditional failure detection approaches (i.e., BFD and LOS) and recovery approaches (i.e., restoration and protection), that were able to ensure carrier-grade reliability. However, due to the emergence of new data plane specifications such as Programming Protocol-independent Packet Processors (P4) [291], and Protocol-Oblivious Forwarding (POF) [292] new paths toward the development of novel strategies and standards to support fault-tolerance opened up. Data plane programmability in SDN is the next step towards supporting a fast-growing trend of network programmability [10] and network softwarization (softwarization of future networks and services) [293]. Traditionally, the network data plane was designed to be configurable but with fixed forwarding logic (packet processing with pre-defined logic). However, the SDN programmable data plane should provide the flexibility to modify forwarding logic (customized packet processing). Concerning the SDN data plane, there are still error detection and recovery issues, which require careful consideration. For instance, the current probe-based testing solution takes a long time to generate probe packets [294], [295], making consistency between control plane policies and data plane forwarding behaviors difficult. Furthermore, additional new pipelines are required in the switch data path for collecting traffic statistics [296]; this process itself can cause errors.

Due to these challenges, the idea of data plane programmability has attracted significant interest from both academia and industry [297]. Recent research studies addressed SDN data plane programmability. New data plane specification (eg., P4 and POF) has been evolved which extend the feature of SDN beyond OpenFlow specifications [298]. These new data plane specifications can optimize fault-management in SDN, thus improving SDN architecture fault-tolerance and reliability aspects. we believe that data plane programmability is an important area for future SDN development.

6.3.2.2 Controller Architecture for Mission Critical Communications

SDN Controller fault-tolerance research studies included in Table 2.7 were focused on designing fault-tolerant SDN controller in scenarios where parameters such as throughput, packet loss, latency, jitter, and redundancy are more flexible than mission-critical communications (industrial networks and intelligent systems) [299], [300], where these parameters have more stringent demands. Mission-critical applications are common in different sectors including military, hospital, automotive safety, and air-traffic control systems [301]. Unfortunately, the research and development of fault-tolerant SDN controller for mission-critical applications have been overlooked. Not even the SDN fault-tolerant controller research efforts (other than mission-critical communication) are still not yet fully developed. Scalability, performance, and data consistency in SDN multi-controller architectures is still an area of intense investigation [302], [303]. There is a need to develop fault-tolerant SDN control networks for mission-critical applications, where designing SDN controller for mission-critical applications is of significant importance and quite challenging hence, we believe that this topic should be addressed comprehensively in future research.

6.3.2.3 Software Tools for SDN Applications Development

SDN fault-tolerance research studies included in Table 2.8 were focused on developing software tools for troubleshooting, writing fault-tolerant programs, and detecting any network policy violations in the application plane. However, the developed fault-tolerant software tools are still having many shortcomings, for instance, incomplete repair mechanisms and high overhead for recovery [304]. Due to the diversity of network protocols for SDN Southbound and Northbound APIs, and the underway standardization of these diverse protocols, the new SDN applications development has not been accelerated. Hence, the developed software tools have not been comprehensively tested and developed to support the diversity of network protocols used in SDN networks. There is a need to develop improved software tools in order to enable application plane fault-tolerance in future SDN deployments.

6.3.3 Service Assurance for Hybrid Networks: Future Research Directions

Guaranteeing assurances in virtual services is more difficult than in a legacy network, which is static and usually well defined. It is now the correct time to develop a new Service Assurance model (considering the dynamic nature of the new end-to-end services) to replace existing strategies in legacy networks (mostly static end-to-end services). We need new Service Assurance systems to adapt to network dynamics and manage the veracity (trustworthiness) of information in real-time environments to reflect and update the current network state well within effective response times in Hybrid Networks. Operators need to design, rethink, and re-shape Service Assurance processes according to the new Service Assurance requirement for Hybrid Networks. This brings new challenges as well as research and development opportunities.

6.3.3.1 Standardized Automation

There is a large uncertainty among CSPs about how to transform their infrastructure to support new Service Assurance models coping with software-driven networks. CSPs are hesitant to migrate to Proactive processes because it requires major changes in CSP's infrastructure and extensive trusted coordination across the communication Ecosystem. While taking this into consideration, we believe that is it a matter of operators to unify together to design and standardize new Service Assurance models for future networks. If CSPs have to wait for years to advance their OSS to properly support Hybrid Networks, then this implies that we cannot benefit from virtualization [305]. Indeed, this is something CSPs are concerned, because new services keep evolving, and more and more network devices are connecting to the Internet. This situation can cause performance bottlenecks and service degradation in the evolved Hybrid Networks if Service Assurance models are not upgraded [306]. At present, Service Assurance still requires significant human intervention. It seems clear that future Service Assurance needs to evolve and rely on virtualization, dynamic control, big data collection, and data analytics engines using advanced machine learning and artificial intelligence techniques to automate self-healing processes with high accuracy[307], [308].

6.3.3.2 Improved Monitoring

Another important aspect is how the monitoring system keeps track of performance in a dynamic environment with rapidly changing network demands and services, while also guaranteeing data consistency as services change. It is quite challenging to design a unified model that can integrate multiple networks and assure services with standardized service element interfaces. There is a need to develop agile software platforms and intelligent interfacing with data engines and other resources. Operators also need to consider defining service elements such as physical, logical, virtual, and communication reference interfaces to establish their relationship to different technical networking domains to ensure optimal performance in dynamic networks.

CSPs have the opportunity to design a new Service Assurance model for Hybrid Networks considering Cloud Computing existing service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Application as a Service (AaaS) (these services are deployed as public, private and hybrid clouds) in order to generate new revenue streams. But all these aspects need to be seamlessly included in the monitoring processes.

6.3.3.3 Standardized Software Development

Harmonization of the communications between DevOps teams is another key challenge, which includes defining the business processes and determining the correct tools required to meet the processes. Insufficient communication between DevOps teams can become extremely problematic and transiting to DevOps can be risky for enterprises if not handled appropriately.

Another difficulty is that there is no well-established telecom standardization of DevOps practices. DevOps practices are still in their infancy, and new DevOps tools keep evolving. At present, due to these reasons, commercial transiting in Telecoms into DevOps is complex. Furthermore, structuring DevOps practices to high-reliability systems (networks) is still an area of intense investigation [309].

CSPs have the opportunity, as well as the challenge to develop a unified technology platform that can integrate third-party network application functions/virtual network functions/ application functions and combine software-driven technologies such as SDN, NFV, and MEC in order to fulfill the needs of new business verticals (as handled in 5G networks). The ETSI defined OSM-MANO architecture, an orchestration system that is being developed over the last years to create a unified technology platform for third-party network application functions but still has a very basic Service Assurance, and these efforts should be pursued.

References

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. “Network Function Virtualization: State-of-the-Art and Research Challenges”. In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 236–262. DOI: 10.1109/COMST.2015.2477041.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. “Software-Defined Networking: A Comprehensive Survey”. In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76. DOI: 10.1109/JPROC.2014.2371999.

[3] R. Jain and S. Paul. “Network virtualization and software defined networking for cloud computing: a survey”. In: *IEEE Communications Magazine* 51.11 (2013), pp. 24–31. DOI: 10.1109/MCOM.2013.6658648.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”. In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376. DOI: 10.1109/COMST.2015.2444095.

[5] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos. “A Survey of Information-Centric Networking Research”. In: *IEEE Communications Surveys Tutorials* 16.2 (2014), pp. 1024–1049. DOI: 10.1109/SURV.2013.070813.00063.

[6] Nisha Panwar, Shantanu Sharma, and Awadhesh Kumar Singh. “A survey on 5G: The next generation of mobile communication”. In: *Physical Communication* 18 (2016). Special Issue on Radio Access Network Architectures and Resource Management for 5G, pp. 64–84. ISSN: 1874-4907. DOI: 10.1016/j.phycom.2015.10.006.

[7] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. “SIMPLE-fying Middlebox Policy Enforcement Using SDN”. In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 27–38. ISSN: 0146-4833. DOI: 10.1145/2534169.2486022.

[8] Bo Yi, Xingwei Wang, Keqin Li, Sajal k. Das, and Min Huang. “A comprehensive survey of Network Function Virtualization”. In: *Computer Networks* 133 (2018), pp. 212–262. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2018.01.021.

[9] Edelman Jason, Lowe Scott, and Oswalt Matt. “Network Programmability and Automation: Skills for the Next-Generation Network Engineer”. In: 1st ed. USA, O’Reilly Media, 2018, pp.151–176.

[10] Gooley Ryan. “Programming and Automating Cisco Networks”. In: 1st ed. USA,Cisco Press, 2016, pp.1–64.

[11] Nick Feamster, Jennifer Rexford, and Ellen Zegura. “The Road to SDN: An Intellectual History of Programmable Networks”. In: *SIGCOMM Comput. Commun. Rev.* 44.2 (Apr. 2014), pp. 87–98. ISSN: 0146-4833. DOI: 10.1145/2602204.2602219.

[12] B. Jennings, S. V. Der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, and J. Strassner. “Towards autonomic management of communications networks”. In: *IEEE Communications Magazine* 45.10 (2007), pp. 112–121. DOI: 10.1109/MCOM.2007.4342833.

[13] S. Dobson, F. Massacci, S. Denazis, P. Nixon, A. Fernandez, F. Saffre, D. Gaiti, N. Schmidt, E. Gelenbe, and F. Zambonelli. “A survey of autonomic communications”. In: *ACM Trans. Auton. Adapt. Syst.* 1.2 (Dec. 2006), pp. 223–259.

[14] William Stallings. "Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud". In: 1st ed. USA,Addison-Wesley Professional, 2015, pp.80–85.

[15] Y. Li and M. Chen. "Software-Defined Network Function Virtualization: A Survey". In: *IEEE Access* 3 (2015), pp. 2542–2553. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2015.2499271.

[16] Stefano Vissicchio, Laurent Vanbever, and Olivier Bonaventure. "Opportunities and Research Challenges of Hybrid Software Defined Networks". In: *SIGCOMM Comput. Commun. Rev.* 44.2 (Apr. 2014), pp. 70–75. ISSN: 0146-4833. DOI: 10.1145/2602204.2602216. URL: https://doi.org/10.1145/2602204.2602216.

[17] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. "Are we ready for SDN? Implementation challenges for software-defined networks". In: *IEEE Communications Magazine* 51.7 (2013), pp. 36–43. DOI: 10.1109/MCOM.2013.6553676.

[18] *Delivering High Availability in Carrier Grade NFV Infrastructures.* VMWare, Inc. CA 94304, USA. URL: https://www.vmware.com/files/pdf/techpaper/vmware-vcloud-nfv-high-availability.pdf.

[19] Margaret Chiosi, Don Clarke, Peter Willis, Andy Reid, James Feger, Michael Bugenhagen, Waqar Khan, Michael Fargano, Chunfeng Cui, Hui Deng, et al. "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action". In: *SDN and OpenFlow World Congress.* Vol. 48. sn. 2012.

[20] *ETSI GS NFV 002 V1.2.1:Network Functions Virtualisation (NFV); Architectural Framework.* RGS/NFV-002. ETSI (ISG). Doc.Ref: RGS/NFV-002,Sophia-Antipolis Cedex France, 2014. URL: https://docbox.etsi.org/ISG/NFV/open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf.

[21] *ETSI GS NFV 002 V1.1.1:Network Functions Virtualisation (NFV); Architectural Framework.* DGS/NFV-0010. ETSI (ISG). Doc.Ref: DGS/NFV-0010,Sophia-Antipolis Cedex France, 10.2013. URL: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf.

[22] Malathi Veeraraghavan, Takehiro Sato, Molly Buchanan, Reza Rahimi, Satoru Okamoto, and Naoaki Yamanaka. "Network Function Virtualization: A Survey". In: *IEICE Transactions on Communications* E100.B.11 (2017), pp. 1978–1991. DOI: 10.1587/transcom.2016NNI0001.

[23] *The Global Goals for Sustainable Development.* Accessed: Dec. 30, 2020. GlobalGoals. URL: https://www.globalgoals.org/.

[24] B. Bitner and S. Greenlee. *z/VM – A Brief Review of Its 40 Year History.* IBM Corporation, 2012. URL: http://www.vm.ibm.com/vm40hist.pdf.

[25] *Understanding Virtualization.* Accessed: Dec. 30, 2020. RedHat Corporation. URL: https://www.redhat.com/en/topics/virtualization.

[26] J. E. van der Merwe, S. Rooney, L. Leslie, and S. Crosby. "The Tempest-a practical framework for network programmability". In: *IEEE Network* 12.3 (1998), pp. 20–28. ISSN: 0890-8044. DOI: 10.1109/65.690958.

[27] M. R. Macedonia and D. P. Brutzman. "MBone provides audio and video across the Internet". In: *Computer* 27.4 (1994), pp. 30–36. ISSN: 0018-9162. DOI: 10.1109/2.274996.

[28] R Hinden and J Postel. *IPv6 Testing Address Allocation. Internet Engineering Task Force.* Tech. rep. January 1996. RFC 1897, obsoleted by RFC 2471 on" 6bone Phaseout, 1996.

[29] Joe Touch. "Dynamic Internet overlay deployment and management using the X-Bone". In: *Computer Networks* 36.2 (2001). Theme issue: Overlay Networks, pp. 117–135. ISSN: 1389-1286. DOI: 10.1016/S1389-1286(01)00172-4.

[30] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. "PlanetLab: An overlay testbed for broad-coverage services". In: *ACM Comput. Commun. Rev.* 33.3 (2003), pp. 3–12.

[31] Larry Peterson, Tom Anderson, Dan Blumenthal, Dean Casey, David Clark, Deborah Estrin, Joe Evans, Dipankar Raychaudhuri, Mike Reiter, Jennifer Rexford, Scott Shenker, and John Wroclawski. "GENI design principles". English (US). In: *Computer* 39 (Sept. 2006), pp. 102–105. ISSN: 0018-9162. DOI: 10.1109/MC.2006.307.

[32] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. "In VINI veritas: realistic and controlled network experimentation". In: *ACM SIGCOMM Computer Communication Review* 36.4 (2006), pp. 3–14.

[33] Ying Zhang. "Network function virtualization". In: 1st ed. USA,Wiley-IEEE Press, 2018, pp.22–36.

[34] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks". In: *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)* (2018), pp. 1–1993. DOI: 10.1109/IEEESTD.2018.8403927.

[35] Maria Napierala, Lawrence Kreeger, Thomas Narten, David Black, Luyuan Fang, and Eric Gray. *VLAN Aggregation for Efficient IP Address Allocation, document RFC 3069, Internet Requests for Comments, RFC Editor, Feb. 2001.* URL: https://www.rfc-editor.org/rfc/pdfrfc/rfc3069.txt.pdf.

[36] V. Moreno and K. Reddy. "Network virtualization". In: 1st ed. USA,Cisco Press, 2006, pp.35–53.

[37] Mallik Mahalingam, Dinesh Dutt, Kenneth Duda, Puneet Agarwal, Lawrence Kreeger, T Sridhar, Mike Bursell, and Chris Wright. *Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks.* Tech. rep. IETF, 2014.

[38] Ramesh K. Sitaraman, Mangesh Kasbekar, Woody Lichtenstein, and Manish Jain. "Overlay Networks: An Akamai Perspective". In: *Advanced Content Delivery, Streaming, and Cloud Services.* John Wiley & Sons, Ltd, 2014. Chap. 16, pp. 305–328. ISBN: 9781118909690. DOI: 10.1002/9781118909690.ch16.

[39] Maria Napierala, Lawrence Kreeger, Thomas Narten, David Black, Luyuan Fang, and Eric Gray. *Problem Statement: Overlays for Network Virtualization, document RFC 7364, Internet Requests for Comments, RFC Editor, Oct. 2014.* URL: https://www.rfc-editor.org/rfc/pdfrfc/rfc7364.txt.pdf.

[40] Jim Doherty. "SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization". In: 1st ed. USA, Addison-Wesley Professional, 2016, pp.3–34.

[41] *VMware vSphere Documentation.* Accessed: Dec. 30, 2020. VMware. URL: https://docs.vmware.com/en/VMware-vSphere/index.html?topic=%2Fcom.vmware.vsphere.vm_admin.doc_50%2FGUID-CEFF6D89-8C19-4143-8C26-4B6D6734D2CB.html.

[42] Rajendra Chayapathi, Syed F Hassan, and Paresh Shah. "Network Functions Virtualization (NFV) with a Touch of SDN". In: 1st ed. USA,Addison-Wesley Professional, 2016, pp.37–62.

[43] Hyungro Lee. "Virtualization Basics: Understanding Techniques and Fundamentals". In: *School of Informatics and Computing, Indiana University* 815 (2014).

[44] *Learn about hypervisors, system virtualization, and how it works in a cloud environment.* Accessed: Dec. 30, 2020. IBM Corporation. URL: https://developer.ibm.com/articles/cl-hypervisorcompare/.

[45] Portnoy Matthew. "Virtualization Essentials, 2nd Edition". In: 2nd ed. Sybex, USA, 2016, pp.21–36.

[46] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. "An updated performance comparison of virtual machines and Linux containers". In: *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* 2015, pp. 171–172. DOI: 10.1109/ISPASS.2015.7095802.

[47] Pavan Sutha Varma Indukuri. “Performance comparison of Linux containers (LXC) and OpenVZ during live migration”. MSc, dissertation. Blekinge Institute of Technology, Sweden, 2016.

[48] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca, and J. Folgueira. “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges”. In: *IEEE Communications Magazine* 55.5 (2017), pp. 80–87. DOI: 10.1109/MCOM.2017.1600935.

[49] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina. “Network Slicing in 5G: Survey and Challenges”. In: *IEEE Communications Magazine* 55.5 (2017), pp. 94–100. DOI: 10.1109/MCOM.2017.1600951.

[50] *Unikernels*. Accessed: Dec. 30, 2020. URL: https://wiki.xenproject.org/wiki/Unikernels.

[51] Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. “Unikernels: Library Operating Systems for the Cloud”. In: *SIGPLAN Not.* 48.4 (Mar. 2013), pp. 461–472. ISSN: 0362-1340. DOI: 10.1145/2499368.2451167.

[52] Michael J De Lucia. *A Survey on Security Isolation of Virtualization, Containers, and Unikernels*. Tech. rep. US Army Research Laboratory Aberdeen Proving Ground United States, 2017.

[53] *What is a Container*. Accessed: Dec. 30, 2020. Docker, Inc. URL: https://www.docker.com/resources/what-container.

[54] *Projects Open source work on unikernels*. Accessed: Dec. 30, 2020. URL: http://unikernel.org/projects/.

[55] Pekka Enberg. “A Performance Evaluation of Hypervisor, Unikernel, and Container Network I/O Virtualization”. MSc, dissertation. University of Helsinki, 2016.

[56] Network Operators. “Network functions virtualization, an introduction, benefits, enablers, challenges and call for action”. In: *SDN and OpenFlow SDN and OpenFlow World Congress*. 2012.

[57] *Network functions virtualisation*. Accessed: Dec. 30, 2020. ETSI 650, route des Lucioles 06921 Sophia-Antipolis Cedex France. URL: http://www.etsi.org/technologies-clusters/technologies/nfv.

[58] *Network functions virtualisation completes first phase of work*. Accessed: Dec. 30, 2020. ETSI 650, route des Lucioles 06921 Sophia-Antipolis Cedex France. URL: https://www.etsi.org/news-events/news/864-2015-01-press-etsi-network-functions-virtualisation.

[59] Ken Gray and Thomas D Nadeau. “Network Function Virtualization”. In: 1st ed. USA,Morgan Kaufmann, 2016, pp.49–76.

[60] *2018-NFV Report Series Part 3: State of the VNF Ecosystem*. Accessed: Dec. 30, 2020. SDxCentral. URL: https://www.sdxcentral.com/reports/next-gen-data-center-networking-download-2018.

[61] *NFV Proofs of Concept*. Accessed: Dec. 30, 2020. ETSI 650, route des Lucioles 06921 Sophia-Antipolis Cedex France. URL: http://www.etsi.org/technologies-clusters/technologies/nfv/nfv-poc.

[62] *ETSI GS NFV 003, V1.3.1:Network Functions Virtualization (NFV); Architectural Framework*. RGS/NFV-003ed131. ETSI (ISG). Doc.Ref: DGS/NFV-0010,Sophia-Antipolis Cedex France, 1.2018. URL: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.03.01_60/gs_nfv003v010301p.pdf.

[63] Prabhu Krish and Donovan John. “Building the Network of the Future”. In: 1st ed. USA,Chapman and Hall/CRC, 2017, pp.49–66.

[64] *Network functions virtualisation*. Accessed: Dec. 30, 2020. ETSI 650, route des Lucioles 06921 Sophia-Antipolis Cedex France. URL: https://www.etsi.org/technologies/nfv.

[65] J. Gil Herrera and J. F. Botero. “Resource Allocation in NFV: A Comprehensive Survey”. In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 518–532. ISSN: 1932-4537. DOI: 10.1109/TNSM.2016.2598420.

[66] W. Yang and C. Fung. “A survey on security in network functions virtualization”. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. 2016, pp. 15–19. DOI: 10.1109/NETSOFT.2016.7502434.

[67] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)”. In: *IEEE Network* 28.6 (2014), pp. 18–26. ISSN: 0890-8044. DOI: 10.1109/MNET.2014.6963800.

[68] *ETSI GS NFV-PER 001 V1.1.1:Network Functions Virtualisation (NFV); NFV Performance and Portability Best Practises*. DGS/NFV-PER001. ETSI (ISG). Doc.Ref: DGS/NFV-PER001 ,Sophia-Antipolis Cedex France, June.2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/001/01.01.01_60/gs_nfv-per001v010101p.pdf.

[69] *ETSI GS NFV-PER 001 V1.1.2:Network Functions Virtualisation (NFV); NFV Performance and Portability Best Practises*. RGS/NFV-PER001ed112. ETSI (ISG). Doc.Ref: RGS/NFV-PER001ed112, Sophia-Antipolis Cedex France, Dec.2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-PER/001_099/001/01.01.02_60/gs_nfv-per001v010102p.pdf.

[70] *Hardware Acceleration for Network Services*. WP-HW-ACCERELATION-NS-1/2017. Netronome Systems, Inc. CA 95054, USA, Jan.2017. URL: https://www.netronome.com/media/documents/WP_Hardware_Acceleration.pdf.

[71] *Manage the future now*. 4AA5-6431ENW. Hewlett-Packard Development Company, L.P. Jan.2015. URL: http://www.5gamericas.org/files/6414/3274/6235/White-Paper-Transforming-OSS-for-NFV.pdf.

[72] *Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure*. ATIS-I-0000044. ATIS. 1200 G Street, NW Suite 500 Washington, DC 20005, Oct.2013. URL: https://access.atis.org/apps/group_public/download.php/20398/Operational%20Opportunities.pdf.

[73] V. A. Cunha, I. D. Cardoso, J. P. Barraca, and R. L. Aguiar. “Policy-driven vCPE through dynamic network service function chaining”. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. June 2016, pp. 156–160. DOI: 10.1109/NETSOFT.2016.7502463.

[74] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. “A survey on service function chaining”. In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2016.09.001.

[75] R. Szabo, M. Kind, F. J. Westphal, H. Woesner, D. Jocha, and A. Csaszar. “Elastic network functions: opportunities and challenges”. In: *IEEE Network* 29.3 (May 2015), pp. 15–21. ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113220.

[76] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. “Virtual Network Embedding: A Survey”. In: *IEEE Communications Surveys Tutorials* 15.4 (Fourth 2013), pp. 1888–1906. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.013013.00155.

[77] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba. “On tackling virtual data center embedding problem”. In: *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. May 2013, pp. 177–184.

[78] M. Falkner, A. Leivadeas, I. Lambadaris, and G. Kesidis. “Performance analysis of virtualized network functions on virtualized systems architectures”. In: *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. Oct. 2016, pp. 71–76. DOI: 10.1109/CAMAD.2016.7790333.

[79] J. Anderson, H. Hu, U. Agarwal, C. Lowery, H. Li, and A. Apon. “Performance considerations of network functions virtualization using containers”. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*. Feb. 2016, pp. 1–7. DOI: 10.1109/ICCNC.2016.7440668.

[80] P. Veitch, M. J. McGrath, and V. Bayon. "An instrumentation and analytics framework for optimal and robust NFV deployment". In: *IEEE Communications Magazine* 53.2 (Feb. 2015), pp. 126–133. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045400.

[81] E. Hernandez-Valencia, S. Izzo, and B. Polonsky. "How will NFV/SDN transform service provider opex?" In: *IEEE Network* 29.3 (May 2015), pp. 60–67. ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113227.

[82] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, and R. Boutaba. "Elastic virtual network function placement". In: *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. Oct. 2015, pp. 255–260. DOI: 10.1109/CloudNet.2015.7335318.

[83] G. Galante and L. C. E. d. Bona. "A Survey on Cloud Computing Elasticity". In: *2012 IEEE Fifth International Conference on Utility and Cloud Computing*. Nov. 2012, pp. 263–270. DOI: 10.1109/UCC.2012.30.

[84] R. Mijumbi, J. Serrat, J. Gorricho, S. Latre, M. Charalambides, and D. Lopez. "Management and orchestration challenges in network functions virtualization". In: *IEEE Communications Magazine* 54.1 (2016), pp. 98–105. ISSN: 0163-6804. DOI: 10.1109/MCOM.2016.7378433.

[85] *ETSI GS NFV 001 V1.1.1:Network Functions Virtualisation (NFV); Management and Orchestration*. DGS/NFV-MAN001. ETSI (ISG). Doc.Ref: DGS/NFV-MAN001, Sophia-Antipolis Cedex France, Dec.2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf.

[86] K. Katsalis, N. Nikaein, and A. Edmonds. "Multi-Domain Orchestration for NFV: Challenges and Research Directions". In: *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*. Dec. 2016, pp. 189–195. DOI: 10.1109/IUCC-CSS.2016.034.

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

[88] M. Bagaa, T. Taleb, and A. Ksentini. "Service-aware network function placement for efficient traffic handling in carrier cloud". In: *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. Apr. 2014, pp. 2402–2407. DOI: 10.1109/WCNC.2014.6952725.

[89] S. Mehraghdam, M. Keller, and H. Karl. "Specifying and placing chains of virtual network functions". In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. Oct. 2014, pp. 7–13. DOI: 10.1109/CloudNet.2014.6968961.

[90] M. Bouet, J. Leguay, and V. Conan. "Cost-based placement of vDPI functions in NFV infrastructures". In: *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. Apr. 2015, pp. 1–9. DOI: 10.1109/NETSOFT.2015.7116121.

[91] O. Houidi, O. Soualah, W. Louati, M. Mechtri, D. Zeghlache, and F. Kamoun. "An Efficient Algorithm for Virtual Network Function Scaling". In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–7. DOI: 10.1109/GLOCOM.2017.8254727.

[92] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku. "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure". In: *The 16th Asia-Pacific Network Operations and Management Symposium*. Sept. 2014, pp. 1–6. DOI: 10.1109/APNOMS.2014.6996545.

[93] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. May 2015, pp. 98–106. DOI: 10.1109/INM.2015.7140281.

[94] B. Addis, D. Belabed, M. Bouet, and S. Secci. "Virtual network functions placement and routing optimization". In: *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*. Oct. 2015, pp. 171–177. DOI: 10.1109/CloudNet.2015.7335301.

[95] *ETSI GS NFV 003 V1.4.1:Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV*. RGS/NFV-003ed141. ETSI (ISG). Doc.Ref: RGS/NFV-003ed141, Sophia-

Antipolis Cedex France, Aug.2018. URL: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.04.01_60/gs_nfv003v010401p.pdf.

[96] B. Chatras and F. F. Ozog. "Network functions virtualization: the portability challenge". In: *IEEE Network* 30.4 (July 2016), pp. 4–8. ISSN: 0890-8044. DOI: 10.1109/MNET.2016.7513857.

[97] Tom R. Halfhill. *Beyond the Data Center: How Network-Function Virtualization Enables New Customer-Premise Services.* The Linley Group. Feb.2016. URL: https://www.nxp.com/docs/en/white-paper/NXP-NFV-WHITEPAPER.pdf.

[98] Luis M. Contreras, Paul Doolan, Håkon Lønsethagen, and Diego R. López. "Operational, organizational and business challenges for network operators in the context of SDN and NFV". In: *Computer Networks* 92 (2015). Software Defined Networks and Virtualization, pp. 211–217. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2015.07.016.

[99] *Survey: The Top 5 Challenges Preventing SDN/NFV Deployment.* Accessed: Dec. 30, 2020. Netcracker. URL: https://www.netcracker.com/insights/general/survey-top-5-challenges-preventing-sdn-nfv-deployment.html.

[100] R. Mersh. *The Business End of NFV.* broadband forum, Fremont, CA 94538, USA. URL: http://www.intercomms.net/issue-29/pdfs/articles/broadband-forum.pdf.

[101] P. Veitch, M. J. McGrath, and V. Bayon. "An instrumentation and analytics framework for optimal and robust NFV deployment". In: *IEEE Communications Magazine* 53.2 (Feb. 2015), pp. 126–133. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7045400.

[102] Hewlett Packard. *Network functions virtualization.* Tech. rep. 4AA5-1114ENW. Hewlett Packard Enterprise Development LP: Hewlett Packard, June 2017. URL: https://h20195.www2.hpe.com/V2/getpdf.aspx/4AA5-1114ENW.

[103] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks". In: *IEEE Communications Surveys Tutorials* 16.3 (Third 2014), pp. 1617–1634. ISSN: 1553-877X. DOI: 10.1109/SURV.2014.012214.00180.

[104] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. "Software-Defined Networking: A Comprehensive Survey". In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76. DOI: 10.1109/JPROC.2014.2371999.

[105] F. Hu, Q. Hao, and K. Bao. "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation". In: *IEEE Communications Surveys Tutorials* 16.4 (Fourthquarter 2014), pp. 2181–2206. ISSN: 1553-877X. DOI: 10.1109/COMST.2014.2326417.

[106] Taimur Bakhshi. "State of the art and recent research advances in software defined networking". In: *Wireless Communications and Mobile Computing* 2017 (2017), pp. 1–35.

[107] Greg Stanley. *Fault Management - the Overall Process and Life Cycle of a Fault.* [Accessed:12-Dec-2019]. 2019. URL: https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/Fault-Management/fault-management.htm.

[108] Kjetil Nørvåg. "An introduction to fault-tolerant systems". In: *Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim* (2000).

[109] A. Lara, A. Kolasani, and B. Ramamurthy. "Simplifying network management using Software Defined Networking and OpenFlow". In: *2012 IEEE International Conference on Advanced Networks and Telecommunciations Systems (ANTS).* Dec. 2012, pp. 24–29. DOI: 10.1109/ANTS.2012.6524222.

[110] Maarten van Steen and Andrew S Tanenbaum. *Distributed systems.* 3rd ed. Prentice-Hall, 2017, pp. 422–499.

[111] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr. "Basic concepts and taxonomy of dependable and secure computing". In: *IEEE Transactions on Dependable and Secure Computing* 1.1 (Jan. 2004), pp. 11–33. ISSN: 1545-5971. DOI: 10.1109/TDSC.2004.2.

[112] Saurabh Hukerikar and Christian Engelmann. “Resilience Design Patterns: A Structured Approach to Resilience at Extreme Scale”. In: *Supercomputing Frontiers and Innovations* 4.3 (2017). ISSN: 2313-8734.

[113] Jerome H Saltzer and M Frans Kaashoek. *Principles of computer system design: an introduction*. Morgan Kaufmann, 2009.

[114] Ravi Jhawar and Vincenzo Piuri. “Chapter 9 - Fault Tolerance and Resilience in Cloud Computing Environments”. In: *Computer and Information Security Handbook*. Ed. by John R. Vacca. 3rd Edition. Boston: Morgan Kaufmann, 2017, pp. 165–181. ISBN: 978-0-12-803843-7. DOI: 10.1016/B978-0-12-803843-7.00009-0.

[115] Moin Hasan and Major Singh Goraya. “Fault tolerance in cloud computing environment: A systematic survey”. In: *Computers in Industry* 99 (2018), pp. 156–172. ISSN: 0166-3615. DOI: 10.1016/j.compind.2018.03.027.

[116] Far B.H. *Fault Tolerant Software Systems: Techniques (Part 4a)*. Accessed: Dec. 30, 2020. [PowerPoint Slides], 2018. URL: https://slideplayer.com/slide/10093304/.

[117] Antonio Bucchiarone, Henry Muccini, and Patrizio Pelliccione. “Architecting Fault-tolerant Component-based Systems: from requirements to testing”. In: *Electronic Notes in Theoretical Computer Science* 168 (2007). Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006), pp. 77–90. ISSN: 1571-0661. DOI: doi.org/10.1016/j.entcs.2006.11.004.

[118] John Dobson, Ian Sommerville, and Guy Dewsbury. “Introduction: Dependability and Responsibility in Context”. In: *Responsibility and Dependable Systems*. Ed. by Guy Dewsbury and John Dobson. London: Springer London, 2007, pp. 1–17. ISBN: 978-1-84628-626-1. DOI: 10.1007/978-1-84628-626-1_1.

[119] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, J Hadi Salim, David Meyer, and Odysseas Koufopavlou. *Software-defined networking (SDN): Layers and architecture terminology*. RFC 7426. RFC Editor, Jan. 2015. URL: https://www.rfc-editor.org/rfc/pdfrfc/rfc7426.txt.pdf.

[120] Glenn Warnock and Amin Nathoo. *Alcatel-Lucent Network Routing Specialist II (NRS II) Self-Study Guide: Preparing for the NRS II Certification Exams*. John Wiley & Sons, 2011.

[121] Othmane Blial, Mouad Ben Mamoun, and Redouane Benaini. “An overview on SDN architectures with multiple controllers”. In: *Journal of Computer Networks and Communications* 2016 (2016), pp. 1–6.

[122] Anderson Santos da Silva, Paul Smith, Andreas Mauthe, and Alberto Schaeffer-Filho. “Resilience support in software-defined networking: A survey”. In: *Computer Networks* 92 (2015), pp. 189–207. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2015.09.012.

[123] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. “OpenFlow: enabling innovation in campus networks”. In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74.

[124] Wolfgang Braun and Michael Menth. “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices”. In: *Future Internet* 6.2 (2014), pp. 302–336. ISSN: 1999-5903. DOI: 10.3390/fi6020302.

[125] E. Haleplidis, J. Hadi Salim, J. M. Halpern, S. Hares, K. Pentikousis, K. Ogawa, W. Wang, S. Denazis, and O. Koufopavlou. “Network Programmability With ForCES”. In: *IEEE Communications Surveys Tutorials* 17.3 (thirdquarter 2015), pp. 1423–1440. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2439033.

[126] Rob Enns, Martin Bjorklund, Juergen Schoenwaelder, and Andy Bierman. *Network configuration protocol (NETCONF)*. Tech. rep. IETF, 2011.

[127] Peter Saint-Andre. *Extensible messaging and presence protocol (XMPP): Core*. Tech. rep. IETF, 2011.

[128] R. Jain and S. Paul. “Network virtualization and software defined networking for cloud computing: a survey”. In: *IEEE Communications Magazine* 51.11 (Nov. 2013), pp. 24–31. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6658648.

[129] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. “OpenFlow: Meeting carrier-grade recovery requirements”. In: *Computer Communications* 36.6 (2013). Reliable Network-based Services, pp. 656–665. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2012.09.011.

[130] Jue Chen, Jinbang Chen, Fei Xu, Min Yin, and Wei Zhang. “When Software Defined Networks Meet Fault Tolerance: A Survey”. In: *Algorithms and Architectures for Parallel Processing*. Ed. by Guojun Wang, Albert Zomaya, Gregorio Martinez, and Kenli Li. Cham: Springer International Publishing, 2015, pp. 351–368. ISBN: 978-3-319-27137-8.

[131] Dave Katz and Dave Ward. *Bidirectional forwarding detection (BFD)*. RFC 5880. RFC Editor, June 2010. URL: https://www.rfc-editor.org/rfc/pdfrfc/rfc5880.txt.pdf.

[132] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester. “Software defined networking: Meeting carrier grade requirements”. In: *2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)*. Oct. 2011, pp. 1–6. DOI: 10.1109/LANMAN.2011.6076935.

[133] A. U. Rehman, R. L. Aguiar, and J. P. Barraca. “A Proposal for Fault-Tolerant and Self-Healing Hybrid SDN Control Network”. In: *INForum 2017 - 9th Simpósio de Informática*. Oct. 2017, pp. 47–52.

[134] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. “Achieving High Utilization with Software-driven WAN”. In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 15–26. ISSN: 0146-4833. DOI: 10.1145/2534169.2486012.

[135] C. Trois, M. D. Del Fabro, L. C. E. de Bona, and M. Martinello. “A Survey on SDN Programming Languages: Toward a Taxonomy”. In: *IEEE Communications Surveys Tutorials* 18.4 (Fourthquarter 2016), pp. 2687–2712. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2553778.

[136] G. N. Nde and R. Khondoker. “SDN testing and debugging tools: A survey”. In: *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. 2016, pp. 631–635. DOI: 10.1109/ICIEV.2016.7760078.

[137] Joshua Reich, Christopher Monsanto, Nate Foster, Jennifer Rexford, and David Walker. “Modular sdn programming with pyretic”. In: *Technical Reprot of USENIX* (2013).

[138] Ryan Beckett, Xuan Kelvin Zou, Shuyuan Zhang, Sharad Malik, Jennifer Rexford, and David Walker. “An Assertion Language for Debugging SDN Applications”. In: *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. HotSDN ’14. Chicago, Illinois, USA: ACM, 2014, pp. 91–96. ISBN: 978-1-4503-2989-7. DOI: 10.1145/2620728.2620743.

[139] N. L. M. v. Adrichem, B. J. v. Asten, and F. A. Kuipers. “Fast Recovery in Software-Defined Networks”. In: *2014 Third European Workshop on Software Defined Networks*. Sept. 2014, pp. 61–66. DOI: 10.1109/EWSDN.2014.13.

[140] L. Sidki, Y. Ben-Shimol, and A. Sadovski. “Fault tolerant mechanisms for SDN controllers”. In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 173–178. DOI: 10.1109/NFV-SDN.2016.7919494.

[141] Keisuke Kuroki, Nobutaka Matsumoto, and Michiaki Hayashi. “Scalable OpenFlow controller redundancy tackling local and global recoveries”. In: *Proceedings of the Fifth International Conference on Advances in Future Internet, Barcelona, Spain*. 2013, pp. 25–31.

[142] T. Yuan, X. Huang, M. Ma, and J. Yuan. “Balance-Based SDN Controller Placement and Assignment with Minimum Weight Matching”. In: *2018 IEEE International Conference on Communications (ICC)*. May 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422637.

[143] G. Wang, Y. Zhao, J. Huang, and W. Wang. “The Controller Placement Problem in Software Defined Networking: A Survey”. In: *IEEE Network* 31.5 (2017), pp. 21–27. ISSN: 0890-8044. DOI: 10.1109/MNET.2017.1600182.

[144] Y. Jiménez, C. Cervelló-Pastor, and A. J. García. "On the controller placement for designing a distributed SDN control layer". In: *2014 IFIP Networking Conference*. June 2014, pp. 1–9. DOI: 10.1109/IFIPNetworking.2014.6857117.

[145] Teemu Koponen, Martin Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama, and Scott Shenker. "Onix: A Distributed Control Platform for Large-scale Production Networks". In: *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. OSDI'10. Vancouver, BC, Canada: USENIX Association, 2010, pp. 351–364.

[146] A. S. Muqaddas, P. Giaccone, A. Bianco, and G. Maier. "Inter-Controller Traffic to Support Consistency in ONOS Clusters". In: *IEEE Transactions on Network and Service Management* 14.4 (Dec. 2017), pp. 1018–1031. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2723477.

[147] V. Pashkov, A. Shalimov, and R. Smeliansky. "Controller failover for SDN enterprise networks". In: *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*. Oct. 2014, pp. 1–6. DOI: 10.1109/MoNeTeC.2014.6995594.

[148] F. Németh, R. Steinert, P. Kreuger, and P. Sköldström. "Roles of DevOps tools in an automated, dynamic service creation architecture". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. May 2015, pp. 1153–1154. DOI: 10.1109/INM.2015.7140455.

[149] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Maziéres, and Nick McKeown. "Where is the Debugger for My Software-defined Network?" In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. HotSDN '12. Helsinki, Finland: ACM, 2012, pp. 55–60. ISBN: 978-1-4503-1477-0. DOI: 10.1145/2342441.2342453.

[150] Peyman Kazemian, Michael Chang, Hongyi Zeng, George Varghese, Nick McKeown, and Scott Whyte. "Real Time Network Policy Checking Using Header Space Analysis". In: *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*. nsdi'13. Lombard, IL: USENIX Association, 2013, pp. 99–112.

[151] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. "Fast failure recovery for in-band OpenFlow networks". In: *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN)*. Mar. 2013, pp. 52–59.

[152] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester. "In-band control, queuing, and failure recovery functionalities for openflow". In: *IEEE Network* 30.1 (Jan. 2016), pp. 106–112. ISSN: 0890-8044. DOI: 10.1109/MNET.2016.7389839.

[153] Purnima Murali Mohan, Tram Truong-Huu, and Mohan Gurusamy. "Fault tolerance in TCAM-limited software defined networks". In: *Computer Networks* 116 (2017), pp. 47–62. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2017.02.009.

[154] H. Li, Q. Li, Y. Jiang, T. Zhang, and L. Wang. "A declarative failure recovery system in software defined networks". In: *2016 IEEE International Conference on Communications (ICC)*. May 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7510887.

[155] Maciej Kuźniar, Peter Perešíni, Nedeljko Vasić, Marco Canini, and Dejan Kostić. "Automatic Failure Recovery for Software-defined Networks". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN '13. Hong Kong, China: ACM, 2013, pp. 159–160. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491218.

[156] Hyojoon Kim, M. Schlansker, J. R. Santos, J. Tourrilhes, Y. Turner, and N. Feamster. "CORONET: Fault tolerance for Software Defined Networks". In: *2012 20th IEEE International Conference on Network Protocols (ICNP)*. Oct. 2012, pp. 1–2. DOI: 10.1109/ICNP.2012.6459938.

[157] L. Schiff, S. Schmid, and M. Canini. "Ground Control to Major Faults: Towards a Fault Tolerant and Adaptive SDN Control Network". In: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. June 2016, pp. 90–96. DOI: 10.1109/DSN-W.2016.48.

[158] J. Chen, J. Chen, J. Ling, and W. Zhang. "Failure recovery using vlan-tag in SDN: High speed with low memory requirement". In: *2016 IEEE 35th International Performance Computing*

and Communications Conference (IPCCC). Dec. 2016, pp. 1–9. DOI: 10.1109/PCCC.2016.7820627.

[159] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. "B4: Experience with a globally-deployed software defined WAN". In: *ACM SIGCOMM Computer Communication Review*. Vol. 43. 4. ACM. 2013, pp. 3–14.

[160] Pankaj Thorat, S. M. Raza, Dung T. Nguyen, Giyeol Im, Hyunseung Choo, and Dongsoo S. Kim. "Optimized Self-healing Framework for Software Defined Networks". In: *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*. IMCOM '15. Bali, Indonesia: ACM, 2015, 7:1–7:6. ISBN: 978-1-4503-3377-1. DOI: 10.1145/2701126.2701235.

[161] Naga Katta, Haoyu Zhang, Michael Freedman, and Jennifer Rexford. "Ravana: Controller Fault-tolerance in Software-defined Networking". In: *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. SOSR '15. Santa Clara, California: ACM, 2015, 4:1–4:12. ISBN: 978-1-4503-3451-8. DOI: 10.1145/2774993.2774996.

[162] Fábio Andrade Botelho, Fernando Manuel Valente Ramos, Diego Kreutz, and Alysson Neves Bessani. "On the Feasibility of a Consistent and Fault-Tolerant Data Store for SDNs". In: *Proceedings of the 2013 Second European Workshop on Software Defined Networks*. EWSDN '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 38–43. ISBN: 978-1-4799-2433-2. DOI: 10.1109/EWSDN.2013.13.

[163] F. Botelho, A. Bessani, F. M. V. Ramos, and P. Ferreira. "On the Design of Practical Fault-Tolerant SDN Controllers". In: *2014 Third European Workshop on Software Defined Networks*. Sept. 2014, pp. 73–78. DOI: 10.1109/EWSDN.2014.25.

[164] P. Fonseca, R. Bennesby, E. Mota, and A. Passito. "A replication component for resilient OpenFlow-based networking". In: *2012 IEEE Network Operations and Management Symposium*. Apr. 2012, pp. 933–939. DOI: 10.1109/NOMS.2012.6212011.

[165] W. H. F. Aly and Y. Kotb. "Towards SDN Fault Tolerance using Petri-Nets". In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. Nov. 2018, pp. 1–3. DOI: 10.1109/ATNAC.2018.8615188.

[166] Amin Tootoonchian and Yashar Ganjali. "HyperFlow: A Distributed Control Plane for OpenFlow". In: *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*. INM/WREN'10. San Jose, CA: USENIX Association, 2010, pp. 3–3.

[167] A. J. Gonzalez, G. Nencioni, B. E. Helvik, and A. Kamisinski. "A Fault-Tolerant and Consistent SDN Controller". In: *2016 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2016, pp. 1–6. DOI: 10.1109/GLOCOM.2016.7841496.

[168] K. ElDefrawy and T. Kaczmarek. "Byzantine Fault Tolerant Software-Defined Networking (SDN) Controllers". In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. June 2016, pp. 208–213. DOI: 10.1109/COMPSAC.2016.76.

[169] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali. "On scalability of software-defined networking". In: *IEEE Communications Magazine* 51.2 (Feb. 2013), pp. 136–141. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6461198.

[170] A. Bessani, J. Sousa, and E. E. P. Alchieri. "State Machine Replication for the Masses with BFT-SMART". In: *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. June 2014, pp. 355–362. DOI: 10.1109/DSN.2014.43.

[171] Brandon Heller, Colin Scott, Nick McKeown, Scott Shenker, Andreas Wundsam, Hongyi Zeng, Sam Whitlock, Vimalkumar Jeyakumar, Nikhil Handigol, James McCauley, Kyriakos Zarifis, and Peyman Kazemian. "Leveraging SDN Layering to Systematically Troubleshoot Networks". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN '13. Hong Kong, China: ACM, 2013, pp. 37–42. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491197.

[172] Colin Scott, Andreas Wundsam, Barath Raghavan, Aurojit Panda, Andrew Or, Jefferson Lai, Eugene Huang, Zhi Liu, Ahmed El-Hassany, Sam Whitlock, H.B. Acharya, Kyriakos Zarifis, and Scott Shenker. "Troubleshooting Blackbox SDN Control Software with Minimal Causal Sequences". In: *SIGCOMM Comput. Commun. Rev.* 44.4 (Aug. 2014), pp. 395–406. ISSN: 0146-4833. DOI: 10.1145/2740070.2626304.

[173] Marco Canini, Daniele Venzano, Peter Perešíni, Dejan Kostić, and Jennifer Rexford. "A {NICE} Way to Test OpenFlow Applications". In: *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*. 2012, pp. 127–140.

[174] Mark Reitblatt, Marco Canini, Arjun Guha, and Nate Foster. "FatTire: Declarative Fault Tolerance for Software-defined Networks". In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN '13. Hong Kong, China: ACM, 2013, pp. 109–114. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491187.

[175] Balakrishnan Chandrasekaran and Theophilus Benson. "Tolerating SDN Application Failures with LegoSDN". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. HotNets-XIII. Los Angeles, CA, USA: ACM, 2014, 22:1–22:7. ISBN: 978-1-4503-3256-9. DOI: 10.1145/2670518.2673880.

[176] Balakrishnan Chandrasekaran, Brendan Tschaen, and Theophilus Benson. "Isolating and Tolerating SDN Application Failures with LegoSDN". In: *Proceedings of the Symposium on SDN Research*. SOSR '16. Santa Clara, CA, USA: ACM, 2016, 7:1–7:12. ISBN: 978-1-4503-4211-7. DOI: 10.1145/2890955.2890965.

[177] *Open Network Operating System (ONOS)*. Accessed: Dec. 30, 2020. Open Network Foundation. URL: https://www.opennetworking.org/onos/.

[178] *The POX network software platform*. Accessed: Dec. 30, 2020. Github. URL: http://www.projectfloodlight.org/floodlight/.

[179] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. "NOX: Towards an Operating System for Networks". In: *SIGCOMM Comput. Commun. Rev.* 38.3 (July 2008), pp. 105–110. ISSN: 0146-4833. DOI: 10.1145/1384609.1384625.

[180] Nate Foster, Rob Harrison, Michael J. Freedman, Christopher Monsanto, Jennifer Rexford, Alec Story, and David Walker. "Frenetic: A Network Programming Language". In: *Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming*. ICFP '11. Tokyo, Japan: ACM, 2011, pp. 279–291. ISBN: 978-1-4503-0865-6. DOI: 10.1145/2034773.2034812.

[181] *Project Floodlight*. Accessed: Dec. 30, 2020. Floodlight. URL: http://www.projectfloodlight.org/floodlight/.

[182] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. "Traffic Engineering with Forward Fault Correction". In: *SIGCOMM Comput. Commun. Rev.* 44.4 (Aug. 2014), pp. 527–538. ISSN: 0146-4833. DOI: 10.1145/2740070.2626314.

[183] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. "Network Architecture for Joint Failure Recovery and Traffic Engineering". In: *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*. SIGMETRICS '11. San Jose, California, USA: ACM, 2011, pp. 97–108. ISBN: 978-1-4503-0814-4. DOI: 10.1145/1993744.1993756.

[184] Timothy L. Hinrichs, Natasha S. Gude, Martin Casado, John C. Mitchell, and Scott Shenker. "Practical Declarative Network Management". In: *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*. WREN '09. Barcelona, Spain: ACM, 2009, pp. 1–10. ISBN: 978-1-60558-443-0. DOI: 10.1145/1592681.1592683.

[185] Akram Hakiri and Pascal Berthou. "Leveraging SDN for the 5G Networks". In: *Software Defined Mobile Networks (SDMN)*. John Wiley & Sons, Ltd, 2015. Chap. 5, pp. 61–80. ISBN: 9781118900253. DOI: 10.1002/9781118900253.ch5.

[186] I. T. Haque and N. Abu-Ghazaleh. “Wireless Software Defined Networking: A Survey and Taxonomy”. In: *IEEE Communications Surveys Tutorials* 18.4 (Fourthquarter 2016), pp. 2713–2737. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2571118.

[187] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer. “Software Defined Optical Networks (SDONs): A Comprehensive Survey”. In: *IEEE Communications Surveys Tutorials* 18.4 (Fourthquarter 2016), pp. 2738–2786. ISSN: 1553-877X. DOI: 10.1109/COMST.2016.2586999.

[188] T Janevski, M Jankovic, and S Markus. “Quality of service regulation manual”. In: *Telecommunication development Bureau* (2017), p. 173.

[189] X. Zhou, R. Li, T. Chen, and H. Zhang. “Network slicing as a service: enabling enterprises’ own software-defined cellular networks”. In: *IEEE Communications Magazine* 54.7 (2016), pp. 146–153. DOI: 10.1109/MCOM.2016.7509393.

[190] S. Kitanov, E. Monteiro, and T. Janevski. “5G and the Fog-Survey of related technologies and research directions”. In: *2016 18th Mediterranean Electrotechnical Conference (MELECON)*. 2016, pp. 1–6. DOI: 10.1109/MELCON.2016.7495388.

[191] Gavin Patterson, Sunil Bharti Mittal, and Bruce Weinelt. “Digital Transformation Initiative Telecommunications Industry (White Paper)”. In: *World Economic Forum*. 2017.

[192] Christian Esteve Rothenberg, ALLAN Vidal, M Salvador, et al. “Hybrid networking towards a software defined era”. In: *Network Innovation through OpenFlow and SDN: Principles and Design*. CRC Press, 2014, pp. 153–198.

[193] O. N. C. Yilmaz, Y. -. E. Wang, N. A. Johansson, N. Brahmi, S. A. Ashraf, and J. Sachs. “Analysis of ultra-reliable and low-latency 5G communication for a factory automation use case”. In: *2015 IEEE International Conference on Communication Workshop (ICCW)*. 2015, pp. 1190–1195.

[194] C. -C. Teng, M. -C. Chen, M. -H. Hung, and H. -J. Chen. “End-to-end Service Assurance in 5G Crosshaul Networks”. In: *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2020, pp. 306–309. DOI: 10.23919/APNOMS50412.2020.9236977.

[195] Mao Yang, Yong Li, Bo Li, Depeng Jin, and Sheng Chen. “Service-oriented 5G network architecture: an end-to-end software defining approach”. In: *International Journal of Communication Systems* 29.10 (2016), pp. 1645–1657. DOI: https://doi.org/10.1002/dac.2941. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.2941. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.2941.

[196] Zheng Chang, Zhenyu Zhou, Sheng Zhou, Tao Chen, and Tapani Ristaniemi. “Towards Service-Oriented 5G: Virtualizing the Networks for Everything-as-a-Service”. In: *IEEE Access* 6 (2018), pp. 1480–1489. DOI: 10.1109/ACCESS.2017.2779170.

[197] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. “5G wireless network slicing for eMBB, URLLC, and mMTC: A communication-theoretic view”. In: *IEEE Access* 6 (2018), pp. 55765–55779.

[198] Madyan Alsenwi, Nguyen H Tran, Mehdi Bennis, Anupam Kumar Bairagi, and Choong Seon Hong. “eMBB-URLLC resource slicing: A risk-sensitive approach”. In: *IEEE Communications Letters* 23.4 (2019), pp. 740–743.

[199] Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. “5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges”. In: *Computer Networks* 167 (2020), p. 106984. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2019.106984. URL: https://www.sciencedirect.com/science/article/pii/S1389128619304773.

[200] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano. “5G RAN Slicing for Verticals: Enablers and Challenges”. In: *IEEE Communications Magazine* 57.1 (2019), pp. 28–34.

[201] A. Kaloxylos. “A Survey and an Analysis of Network Slicing in 5G Networks”. In: *IEEE Communications Standards Magazine* 2.1 (2018), pp. 60–65.

[202] M. Xie, Q. Zhang, A. J. Gonzalez, P. Gronsund, P. Palacharla, and T. Ikeuchi. “Service Assurance in 5G Networks: A Study of Joint Monitoring and Analytics”. In: *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2019, pp. 1–7.

[203] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. “Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions”. In: *IEEE Communications Surveys Tutorials* 20.3 (2018), pp. 2429–2453. DOI: 10.1109/COMST.2018.2815638.

[204] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung. “Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges”. In: *IEEE Communications Magazine* 55.8 (2017), pp. 138–145. DOI: 10.1109/MCOM.2017.1600940.

[205] M. Touloupou, E. Kapassa, C. Symvoulidis, P. Stavrianos, and D. Kyriazis. “An Integrated SLA Management Framework in a 5G Environment”. In: *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 2019, pp. 233–235. DOI: 10.1109/ICIN.2019.8685916.

[206] Albert Banchs, David M Gutierrez-Estevez, Manuel Fuentes, Mauro Boldi, and Silvia Provvedi. “A 5G mobile network architecture to support vertical industries”. In: *IEEE Communications Magazine* 57.12 (2019), pp. 38–44.

[207] M. Xie, C. Banino-Rokkones, P. Gronsund, and A. J. Gonzalez. “Service assurance architecture in NFV”. In: *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2017, pp. 229–235. DOI: 10.1109/NFV-SDN.2017.8169881.

[208] J. Pedreno-Manresa, P. S. Khodashenas, M. S. Siddiqui, and P. Pavon-Marino. “Dynamic QoS/QoE assurance in realistic NFV-enabled 5G Access Networks”. In: *2017 19th International Conference on Transparent Optical Networks (ICTON)*. 2017, pp. 1–4. DOI: 10.1109/ICTON.2017.8025149.

[209] K.S. Sendhil Kumar and N. Jaisankar. “An automated resource management framework for minimizing SLA violations and negotiation in collaborative cloud”. In: *International Journal of Cognitive Computing in Engineering* 1 (2020), pp. 27–35. ISSN: 2666-3074. DOI: https://doi.org/10.1016/j.ijcce.2020.09.001. URL: https://www.sciencedirect.com/science/article/pii/S2666307420300036.

[210] M. Xie, W. Y. Poe, Y. Wang, A. J. Gonzalez, A. M. Elmokashfi, J. A. Pereira Rodrigues, and F. Michelinakis. “Towards Closed Loop 5G Service Assurance Architecture for Network Slices as a Service”. In: *2019 European Conference on Networks and Communications (EuCNC)*. June 2019, pp. 139–143. DOI: 10.1109/EuCNC.2019.8802040.

[211] A. Sgambelluri, F. Tusa, M. Gharbaoui, E. Maini, L. Toka, J. M. Perez, F. Paolucci, B. Martini, W. Y. Poe, J. M. Hernandes, A. Muhammed, A. Ramos, O. G. de Dios, B. Sonkoly, P. Monti, I. Vaishnavi, C. J. Bernardos, and R. Szabo. “Orchestration of Network Services across multiple operators: The 5G Exchange prototype”. In: *2017 European Conference on Networks and Communications (EuCNC)*. June 2017, pp. 1–5. DOI: 10.1109/EuCNC.2017.7980666.

[212] Alexander Stanik, Marc Koerner, and Leonidas Lymberopoulos. “SLA-driven Federated Cloud Networking: Quality of Service for Cloud-based Software Defined Networks”. In: *Procedia Computer Science* 34 (2014). The 9th International Conference on Future Networks and Communications (FNC’14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC’14)/Affiliated Workshops, pp. 655–660. ISSN: 1877-0509. DOI: 10.1016/j.procs.2014.07.093.

[213] P. Martinez-Julia, V. P. Kafle, and H. Asaeda. “Using the Total Cost of Ownership to Decide Resource Adjustment in Virtual Networks”. In: *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 2019, pp. 329–335.

[214] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. “A Survey of DevOps Concepts and Challenges”. In: *ACM Comput. Surv.* 52.6 (Nov. 2019). ISSN: 0360-0300. DOI: 10.1145/3359981.

[215] F. M. A. Erich, C. Amrit, and M. Daneva. “A Qualitative Study of DevOps Usage in Practice”. In: *J. Softw. Evol. Process* 29.6 (June 2017), n/a. ISSN: 2047-7473. DOI: 10.1002/smr.1885.

[216] A. U. Rehman, R. L. Aguiar, and J. P. Barraca. "Network Functions Virtualization: The Long Road to Commercial Deployments". In: *IEEE Access* 7 (2019), pp. 60439–60464. DOI: 10.1109/ACCESS.2019.2915195.

[217] J. A. Wickboldt, W. P. De Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville. "Software-defined networking: management requirements and challenges". In: *IEEE Communications Magazine* 53.1 (2015), pp. 278–285. DOI: 10.1109/MCOM.2015.7010546.

[218] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. "A Survey on Mobile Edge Computing: The Communication Perspective". In: *IEEE Communications Surveys Tutorials* 19.4 (2017), pp. 2322–2358. DOI: 10.1109/COMST.2017.2745201.

[219] Edelman Jason, Lowe Scott, and Oswalt Matt. "Network Programmability and Automation: Skills for the Next-Generation Network Engineer". In: 1st ed. USA, O'Reilly Media, 2018, pp. 17–33.

[220] Edelman Jason, Lowe Scott, and Oswalt Matt. "Network Programmability and Automation: Skills for the Next-Generation Network Engineer". In: 1st ed. USA, O'Reilly Media, 2018, pp.357–455.

[221] Danish Rafique and Luis Velasco. "Machine Learning for Network Automation: Overview, Architecture, and Applications (Invited Tutorial)". In: *J. Opt. Commun. Netw.* 10.10 (2018), pp. D126–D143. DOI: 10.1364/JOCN.10.00D126.

[222] Michael Behringer, Max Pritikin, Steinthor Bjarnason, Alexander Clemm, Brian Carpenter, Sheng Jiang, and Laurent Ciavaglia. "Autonomic networking: Definitions and design goals". In: *Internet, RFC7575* (2015).

[223] Mark O. Riedl. "Human-centered artificial intelligence and machine learning". In: *Human Behavior and Emerging Technologies* 1.1 (2019), pp. 33–36. DOI: 10.1002/hbe2.117.

[224] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle. "A Survey of Autonomic Network Architectures and Evaluation Criteria". In: *IEEE Communications Surveys Tutorials* 14.2 (2012), pp. 464–490. DOI: 10.1109/SURV.2011.042711.00078.

[225] H. Kim and N. Feamster. "Improving network management with software defined networking". In: *IEEE Communications Magazine* 51.2 (Feb. 2013), pp. 114–119. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6461195.

[226] Mi-Jung Choi and James Won-Ki Hong. "Towards management of next generation networks". In: *IEICE transactions on communications* 90.11 (2007), pp. 3004–3014.

[227] Girma M. Yilma, Zarrar F. Yousaf, Vincenzo Sciancalepore, and Xavier Costa-Perez. "Benchmarking open source NFV MANO systems: OSM and ONAP". In: *Computer Communications* 161 (2020), pp. 86–98. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2020.07.013.

[228] R. K. Chawda and G. Thakur. "Big data and advanced analytics tools". In: *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*. 2016, pp. 1–8. DOI: 10.1109/CDAN.2016.7570890.

[229] A. Terra, R. Inam, S. Baskaran, P. Batista, I. Burdick, and E. Fersman. "Explainability Methods for Identifying Root-Cause of SLA Violation Prediction in 5G Network". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–7. DOI: 10.1109/GLOBECOM42002.2020.9322496.

[230] Rob Van der Mei, Hans Van den Berg, Ivan Ganchev, Kurt Tutschku, Philipp Leitner, Pasi Lassila, Wojciech Burakowski, Fidel Liberal, Åke Arvidsson, Tobias Hoβfeld, et al. "State of the art and research challenges in the area of autonomous control for a reliable internet of services". In: *Autonomous Control for a Reliable Internet of Services*. Springer, Cham, 2018, pp. 1–22.

[231] Robert C. Streijl, Stefan Winkler, and David S. Hands. "Mean opinion score (MOS) revisited: methods and applications, limitations and alternatives". In: *Multimedia Systems* 22.2 (Mar. 2016), pp. 213–227. ISSN: 1432-1882. DOI: 10.1007/s00530-014-0446-1.

[232] A. Balalaie, A. Heydarnoori, and P. Jamshidi. "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture". In: *IEEE Software* 33.3 (May 2016), pp. 42–52. ISSN: 0740-7459. DOI: 10.1109/MS.2016.64.

[233] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. "DevOps adoption benefits and challenges in practice: a case study". In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2016, pp. 590–597.

[234] *Service assurance in the 5G era*. TM Forum, 2021. URL: https://inform.tmforum.org/research-reports/service-assurance-in-the-5g-era/.

[235] Guosheng Zhu, Jun Zan, Yang Yang, and Xiaoyun Qi. "A Supervised Learning Based QoS Assurance Architecture for 5G Networks". In: *IEEE Access* 7 (2019), pp. 43598–43606. DOI: 10.1109/ACCESS.2019.2907142.

[236] *How to lead in the Open API economy*. TM Forum, 2021. URL: https://inform.tmforum.org/research-reports/how-to-lead-in-the-open-api-economy/.

[237] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "Fast failure recovery for in-band OpenFlow networks". In: *9th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE. 2013, pp. 52–59.

[238] L. Schiff, S. Schmid, and M. Canini. "Ground Control to Major Faults: Towards a Fault Tolerant and Adaptive SDN Control Network". In: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*. 2016, pp. 90–96. DOI: 10.1109/DSN-W.2016.48.

[239] Diego Kreutz, Fernando Ramos, and Paulo Verissimo. "Towards secure and dependable software-defined networks". In: *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. doi:10.1145/2491185.2491199. ACM. 2013, pp. 55–60. URL: https://doi.org/10.1145/2491185.2491199.

[240] Dimitri Staessens, Sachin Sharma, Didier Colle, Mario Pickavet, and Piet Demeester. "Software defined networking: Meeting carrier grade requirements". In: *Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*. doi:10.1109/LANMAN.2011.6076935. IEEE. 2011, pp. 1–6.

[241] Sachin Sharma, Dimitri Staessens, Didier Colle, Mario Pickavet, and Piet Demeester. "OpenFlow: Meeting carrier-grade recovery requirements". In: *Computer Communications* 36.6 (2013), pp. 656–665. DOI: 10.1016/j.comcom.2012.09.011.

[242] Pankaj Thorat, Rajesh Challa, Syed M Raza, Dongsoo S Kim, and Hyunseung Choo. "Proactive failure recovery scheme for data traffic in software defined networks". In: *NetSoft Conference and Workshops (NetSoft), 2016 IEEE*. doi:10.1109/NETSOFT.2016.7502416. IEEE. 2016, pp. 219–225.

[243] James PG Sterbenz, David Hutchison, Egemen K Çetinkaya, Abdul Jabbar, Justin P Rohrer, Marcus Schöller, and Paul Smith. "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines". In: *Computer Networks* 54.8 (2010). doi:10.1016/j.comnet.2010.03.005, pp. 1245–1265.

[244] Mei-Chen Hsueh, Timothy K Tsai, and Ravishankar K Iyer. "Fault injection techniques and tools". In: *Computer* 30.4 (1997), pp. 75–82. DOI: 10.1109/2.585157.

[245] NGMN Alliance. "5G white paper". In: *Next generation mobile networks, white paper* (2015), pp. 1–125.

[246] 5G PPP Architecture Working Group et al. *View on 5G architecture*. 2016.

[247] Faqir Zarrar Yousaf, Michael Bredel, Sibylle Schaller, and Fabian Schneider. "NFV and SDN-Key Technology Enablers for 5G Networks". In: *IEEE Journal on Selected Areas in Communications* 35.11 (2017), pp. 2468–2478.

[248] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog computing and its role in the internet of things". In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM. 2012, pp. 13–16.

[249] Qi Zhang and Frank H. P. Fitzek. “Mission Critical IoT Communication in 5G”. In: *Future Access Enablers for Ubiquitous and Intelligent Infrastructures*. Ed. by Vladimir Atanasovski and Alberto Leon-Garcia. Cham: Springer International Publishing, 2015, pp. 35–41. ISBN: 978-3-319-27072-2.

[250] H. Shariatmadari, R. Ratasuk, S. Iraji, A. Laya, T. Taleb, R. Jäntti, and A. Ghosh. “Machine-type communications: current status and future perspectives toward 5G systems”. In: *IEEE Communications Magazine* 53.9 (Sept. 2015), pp. 10–17. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7263367.

[251] J. Kang, O. Simeone, and J. Kang. “On the Trade-Off Between Computational Load and Reliability for Network Function Virtualization”. In: *IEEE Communications Letters* 21.8 (Aug. 2017), pp. 1767–1770. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2017.2698040.

[252] Siwar Ben Hadj Said, Bernard Cousin, and Samer Lahoud. “Software defined networking (SDN) for reliable user connectivity in 5G networks”. In: *Network Softwarization (NetSoft), 2017 IEEE Conference on*. IEEE. 2017, pp. 1–5.

[253] A. Aydeger, K. Akkaya, and A. S. Uluagac. “SDN-based resilience for smart grid communications”. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. Nov. 2015, pp. 31–33. DOI: 10.1109/NFV-SDN.2015.7387401.

[254] Luis Tello-Oquendo, Ian F Akyildiz, Shih-Chun Lin, and Vicent Pla. “SDN-based architecture for providing reliable Internet of Things connectivity in 5G systems”. In: *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. IEEE. 2018, pp. 1–8.

[255] J. Sánchez, I. G. Ben Yahia, N. Crespi, T. Rasheed, and D. Siracusa. “Softwarized 5G networks resiliency with self-healing”. In: *1st International Conference on 5G for Ubiquitous Connectivity*. Nov. 2014, pp. 229–233. DOI: 10.4108/icst.5gu.2014.258123.

[256] Jue Chen, Jinbang Chen, Junchen Ling, Junlong Zhou, and Wei Zhang. “Link Failure Recovery in SDN: High Efficiency, Strong Scalability and Wide Applicability”. In: *Journal of Circuits, Systems and Computers* 27 (Sept. 2017), pp. 1–30. DOI: 10.1142/S0218126618500871.

[257] B. Raeisi and A. Giorgetti. “Software-based fast failure recovery in load balanced SDN-based datacenter networks”. In: *2016 6th International Conference on Information Communication and Management (ICICM)*. Oct. 2016, pp. 95–99. DOI: 10.1109/INFOCOMAN.2016.7784222.

[258] Jingyuan Fan, Zilong Ye, Chaowen Guan, Xiujiao Gao, Kui Ren, and Chunming 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*. HotMiddlebox ’15. London, United Kingdom: ACM, 2015, pp. 13–18. ISBN: 978-1-4503-3540-9. DOI: 10.1145/2785989.2786000.

[259] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. “Understanding network failures in data centers: measurement, analysis, and implications”. In: *ACM SIGCOMM Computer Communication Review*. Vol. 41. 4. ACM. 2011, pp. 350–361.

[260] A. Bratterud, A. Walla, H. Haugerud, P. E. Engelstad, and K. Begnum. “IncludeOS: A Minimal, Resource Efficient Unikernel for Cloud Services”. In: *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. Nov. 2015, pp. 250–257. DOI: 10.1109/CloudCom.2015.89.

[261] S. Lange, N. Van Tu, S. -Y. Jeong, D. -Y. Lee, H. -G. Kim, J. Hong, J. -H. Yoo, and J. W. -K. Hong. “A Network Intelligence Architecture for Efficient VNF Lifecycle Management”. In: *IEEE Transactions on Network and Service Management* (2020), pp. 1–1. DOI: 10.1109/TNSM.2020.3015244.

[262] O. Adamuz-Hinojosa, J. Ordonez-Lucena, P. Ameigeiras, J. J. Ramos-Munoz, D. Lopez, and J. Folgueira. “Automated Network Service Scaling in NFV: Concepts, Mechanisms and Scaling Workflow”. In: *IEEE Communications Magazine* 56.7 (2018), pp. 162–169. DOI: 10.1109/MCOM.2018.1701336.

[263] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P. Mekikis, A. Antonopoulos, and C. Verikoukis. “Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture”. In: *IEEE Internet of Things Journal* 7.5 (2020), pp. 4183–4194. DOI: 10.1109/JIOT.2019.2944695.

[264] Panagiotis Trakadas, Panagiotis Karkazis, Helen C. Leligou, Theodore Zahariadis, Felipe Vicens, Arturo Zurita, Pol Alemany, Thomas Soenen, Carlos Parada, Jose Bonnet, and et al. “Comparison of Management and Orchestration Solutions for the 5G Era”. In: *Journal of Sensor and Actuator Networks* 9.1 (Jan. 2020), p. 4. ISSN: 2224-2708. DOI: 10.3390/jsan9010004.

[265] L. Mamushiane, A. A. Lysko, T. Mukute, J. Mwangama, and Z. D. Toit. “Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey”. In: *2019 IEEE 2nd Wireless Africa Conference (WAC)*. 2019, pp. 1–7. DOI: 10.1109/AFRICA.2019.8843421.

[266] *Open Source MANO*. Accessed: Dec. 30, 2020. ETSI. URL: https://osm.etsi.org/.

[267] *ONAP Open Network Automation Platform*. Accessed: Dec. 30, 2020. Linux-Foundation. URL: https://www.onap.org/.

[268] *Open orchestrator*. Accessed: Dec. 30, 2020. Linux-Foundation. URL: https://www.open-o.org/.

[269] *ECOMP (Enhanced Control, Orchestration, Management and Policy) Architecture White Paper*. AT&T, Inc, 2016. URL: https://about.att.com/content/dam/snrdocs/ecomp.pdf.

[270] Giuseppe Antonio Carella and Thomas Magedanz. “Open baton: a framework for virtual network function management and orchestration for emerging software-based 5g networks”. In: *Newsletter* 2016 (2015).

[271] *Tacker-OpenStack*. Accessed: Dec. 30, 2020. O. Foundation. URL: https://wiki.openstack.org/wiki/Tacker.

[272] Antonio Francescon, Giovanni Baggio, Riccardo Fedrizzi, Ramon Ferrusy, Imen Grida Ben Yahiaz, and Roberto Riggio. “X–MANO: Cross–domain management and orchestration of network services”. In: *2017 IEEE Conference on Network Softwarization (NetSoft)*. IEEE. 2017, pp. 1–5.

[273] J. B. Filipe, F. Meneses, A. U. Rehman, D. Corujo, and R. L. Aguiar. “A Performance Comparison of Containers and Unikernels for Reliable 5G Environments”. In: *2019 15th International Conference on the Design of Reliable Communication Networks (DRCN)*. 2019, pp. 99–106. DOI: 10.1109/DRCN.2019.8713677.

[274] *Open Source MANO*. Accessed: Dec. 30, 2020. ETSI. URL: http://osm-download.etsi.org/ftp/osm-doc/vnfd.html.

[275] A. U. Rehman, Rui. L. Aguiar, and João Paulo Barraca. “Testing Virtual Network Functions Auto-Scaling using Open-Source Management and Orchestration”. In: *2021 Telecoms Conference (ConfTELE)*. 2021, pp. 1–6. DOI: 10.1109/ConfTELE50222.2021.9435471.

[276] Petar Kochovski, Vlado Stankovski, Sandi Gec, Francescomaria Faticanti, Marco Savi, Domenico Siracusa, and Seungwoo Kum. “Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing”. In: *Journal of Grid Computing* (2020), pp. 1–18.

[277] Antonio Manzalini. “Softwarization of telecommunications”. In: *it-Information Technology* 57.5 (2015), pp. 321–329.

[278] Bego Blanco, Jose Oscar Fajardo, Ioannis Giannoulakis, Emmanouil Kafetzakis, Shuping Peng, Jordi Pérez-Romero, Irena Trajkovska, Pouria S. Khodashenas, Leonardo Goratti, Michele Paolino, Evangelos Sfakianakis, Fidel Liberal, and George Xilouris. “Technology pillars in the architecture of future 5G mobile networks: NFV, MEC and SDN”. In: *Computer Standards and Interfaces* 54 (2017). SI: Standardization SDN and NFV, pp. 216–228. ISSN: 0920-5489. DOI: 10.1016/j.csi.2016.12.007.

[279] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. “Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges”. In: *Future Generation Computer Systems* 78 (2018), pp. 680–698. ISSN: 0167-739X. DOI: 10.1016/j.future.2016.11.009.

[280] Ejaz Ahmed and Mubashir Husain Rehmani. “Mobile Edge Computing: Opportunities, solutions, and challenges”. In: *Future Generation Computer Systems* 70 (2017), pp. 59–63. ISSN: 0167-739X. DOI: 10.1016/j.future.2016.09.015.

[281] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos. “A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges”. In: *IEEE Communications Surveys Tutorials* 20.1 (2018), pp. 416–464. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2771153.

[282] G. C. Valastro, D. Panno, and S. Riolo. “A SDN/NFV based C-RAN architecture for 5G Mobile Networks”. In: *2018 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*. June 2018, pp. 1–8. DOI: 10.1109/MoWNet.2018.8428882.

[283] J. Costa-Requena, J. L. Santos, V. F. Guasch, K. Ahokas, G. Premsankar, S. Luukkainen, O. L. Pérez, M. U. Itzazelaia, I. Ahmad, M. Liyanage, M. Ylianttila, and E. M. de Oca. “SDN and NFV integration in generalized mobile network architecture”. In: *2015 European Conference on Networks and Communications (EuCNC)*. June 2015, pp. 154–158. DOI: 10.1109/EuCNC.2015.7194059.

[284] Michel S. Bonfim, Kelvin L. Dias, and Stenio F. L. Fernandes. “Integrated NFV/SDN Architectures: A Systematic Literature Review”. In: *ACM Comput. Surv.* 51.6 (Feb. 2019). ISSN: 0360-0300. DOI: 10.1145/3172866.

[285] V. Nguyen, A. Brunstrom, K. Grinnemo, and J. Taheri. “SDN/NFV-Based Mobile Packet Core Network Architectures: A Survey”. In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1567–1602. ISSN: 1553-877X. DOI: 10.1109/COMST.2017.2690823.

[286] *Standards and Technology*. Accessed: Dec. 30, 2020. Open Networking Foundation. URL: https://www.opennetworking.org/software-defined-standards/overview/.

[287] *An Innovative Combination of Standards and Open Source Software*. Accessed: Dec. 30, 2020. DMTF. URL: https://www.dmtf.org/standards.

[288] D. F. Macedo, D. Guedes, L. F. M. Vieira, M. A. M. Vieira, and M. Nogueira. “Programmable Networks—From Software-Defined Radio to Software-Defined Networking”. In: *IEEE Communications Surveys Tutorials* 17.2 (Secondquarter 2015), pp. 1102–1125. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2402617.

[289] Xenofon Foukas, Mahesh K. Marina, and Kimon Kontovasilis. “Software Defined Networking Concepts”. In: *Software Defined Mobile Networks (SDMN)*. John Wiley & Sons, Ltd, 2015. Chap. 3, pp. 21–44. ISBN: 9781118900253. DOI: 10.1002/9781118900253.ch3.

[290] A. U. Rehman, R. L. Aguiar, and J. P. Barraca. “Fault-Tolerance in the Scope of Software-Defined Networking (SDN)”. In: *IEEE Access* 7 (2019), pp. 124474–124490. DOI: 10.1109/ACCESS.2019.2939115.

[291] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. “P4: Programming Protocol-independent Packet Processors”. In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 87–95. ISSN: 0146-4833. DOI: 10.1145/2656877.2656890.

[292] Haoyu Song. “Protocol-oblivious Forwarding: Unleash the Power of SDN Through a Future-proof Forwarding Plane”. In: *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*. HotSDN ’13. Hong Kong, China: ACM, 2013, pp. 127–132. ISBN: 978-1-4503-2178-5. DOI: 10.1145/2491185.2491190.

[293] A. Galis, S. Clayman, L. Mamatas, J. Rubio Loyola, A. Manzalini, S. Kuklinski, J. Serrat, and T. Zahariadis. “Softwarization of Future Networks and Services -Programmable Enabled Networks as Next Generation Software Defined Networks”. In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. Nov. 2013, pp. 1–7. DOI: 10.1109/SDN4FNS.2013.6702557.

[294] Peter Perešíni, Maciej Kuźniar, and Dejan Kostić. “Monocle: Dynamic, Fine-grained Data Plane Monitoring”. In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. CoNEXT ’15. Heidelberg, Germany: ACM, 2015, 32:1–32:13. ISBN: 978-1-4503-3412-9. DOI: 10.1145/2716281.2836117.

[295] K. Bu, X. Wen, B. Yang, Y. Chen, L. E. Li, and X. Chen. “Is every flow on the right track?: Inspect SDN forwarding with RuleScope”. In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. Apr. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524333.

[296] Peng Zhang, Hao Li, Chengchen Hu, Liujia Hu, Lei Xiong, Ruilong Wang, and Yuemei Zhang. “Mind the Gap: Monitoring the Control-Data Plane Consistency in Software Defined Networks”. In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT ’16. Irvine, California, USA: ACM, 2016, pp. 19–33. ISBN: 978-1-4503-4292-6. DOI: 10.1145/2999572.2999605.

[297] H. Farhad, H. Lee, and A. Nakao. “Data Plane Programmability in SDN”. In: *2014 IEEE 22nd International Conference on Network Protocols*. Oct. 2014, pp. 583–588. DOI: 10.1109/ICNP.2014.93.

[298] Weverton Luis da Costa Cordeiro, Jonatas Adilson Marques, and Luciano Paschoal Gaspary. “Data Plane Programmability Beyond OpenFlow: Opportunities and Challenges for Network and Service Operations and Management”. In: *Journal of Network and Systems Management* 25.4 (Oct. 2017), pp. 784–818. ISSN: 1573-7705. DOI: 10.1007/s10922-017-9423-2.

[299] M. Bouet, K. Phemius, and J. Leguay. “Distributed SDN for Mission-Critical Networks”. In: *2014 IEEE Military Communications Conference*. Oct. 2014, pp. 942–948. DOI: 10.1109/MILCOM.2014.162.

[300] Vasileios Gkioulos, Håkon Gunleifsen, and Goitom Kahsay Weldehawaryat. “A Systematic Literature Review on Military Software Defined Networks”. In: *Future Internet* 10.9 (2018). ISSN: 1999-5903. DOI: 10.3390/fi10090088.

[301] Ramon Carreras Ramirez, Quoc-Tuan Vien, Ramona Trestian, Leonardo Mostarda, and Purav Shah. “Multi-path Routing for Mission Critical Applications in Software-Defined Networks”. In: *Industrial Networks and Intelligent Systems*. Ed. by Trung Q Duong and Nguyen-Son Vo. Cham: Springer International Publishing, 2019, pp. 38–48. ISBN: 978-3-030-05873-9.

[302] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali. “On scalability of software-defined networking”. In: *IEEE Communications Magazine* 51.2 (Feb. 2013), pp. 136–141. ISSN: 0163-6804. DOI: 10.1109/MCOM.2013.6461198.

[303] Murat Karakus and Arjan Durresi. “A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN)”. In: *Computer Networks* 112 (2017), pp. 279–293. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2016.11.017.

[304] Yinbo Yu, Xing Li, Xue Leng, Libin Song, Kai Bu, Yan Chen, Jianfeng Yang, Liang Zhang, Kang Cheng, and Xin Xiao. “Fault management in software-defined networking: A survey”. In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 349–392.

[305] *Next-Generation Assurance in NFV Networks*. Huawei Tech, 2017. Shenzhen, China. URL: https://carrier.huawei.com/~/media/CNBG/Downloads/Services/nfv/Next-Generation%20Assurance%20in%20NFV%20Networks.pdf.

[306] C. Benzaid and T. Taleb. “AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions”. In: *IEEE Network* 34.2 (2020), pp. 186–194. DOI: 10.1109/MNET.001.1900252.

[307] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities”. In: *Journal of Internet Services and Applications* 9.1 (2018), p. 16.

[308] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. “Machine Learning for Networking: Workflow, Advances and Opportunities”. In: *IEEE Network* 32.2 (2018), pp. 92–99.

[309] M. Shahin, M. A. Babar, M. Zahedi, and L. Zhu. “Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges”. In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2017, pp. 111–120.