UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARCOS ROCHA DE MORAES FALCÃO

RESOURCE ALLOCATION FOR URLLC IN NFV-MEC

Recife

2022

MARCOS ROCHA DE MORAES FALCÃO

RESOURCE ALLOCATION FOR URLLC IN NFV-MEC

This thesis has been submitted to the Postgraduate Program in Computer Science of the Informatics Center of the Federal University of Pernambuco as a partial requirement to obtain the degree of Doctor in Computer Science.

Main Research Field : Computer Networks and Distributed Systems

Advisor: Kelvin Lopes Dias

Co-Advisor: Andson Marreiros Balieiros

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

F178r Falcão, Marcos Rocha de Moraes
Resource allocation for URLLC in NFV-MEC / Marcos Rocha de Moraes Falcão. – 2022.
99 f.: il., fig., tab.

Orientador: Kelvin Lopes Dias.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências.

1. Rede de computadores e sistemas distribuídos. 2. Computação de borda multiacesso. 3. Virtualização de funções de rede. 4. Alocação de recursos. I. Falcão, Marcos Rocha de Moraes (orientador). II. Título

004.6 CDD (23. ed.) UFPE - CCEN 2022 – 87

Marcos Rocha de Moraes Falcão

"RESOURCE ALLOCATION FOR URLLC IN NFV-MEC"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 14/03/2022.

I dedicate this work to my father, Marcos José de Moraes Falcão.

ACKNOWLEDGEMENTS

First I must give thanks to God for being with me all the time, and for providing me the strength to overcome the challenges in this journey. I would also like to thank my mother Teresa Didier, uncles, aunts, and other family members for the constant encouragement and for being true heroes. I am specially grateful for my advisors Dr. Kelvin Dias, Dr. Andson Balieiro and future Dr. Caio Souza whose expertise and support have been of great value. Furthermore, I must express gratitude to the committee members, Dr. Edmundo Madeira, Dr. Antonio Loureiro, Dr. Renato Mariz, Dr. Divanilson Campelo and Dr. Paulo Cunha for agreeing to evaluate my thesis, and for offering helpful comments to improve it. Finally, I also thank the Science and Technology Foundation of Pernambuco (FACEPE) for its financial support, and the Center of Informatics (CIn-UFPE) for providing a suitable study environment.

ABSTRACT

Multi-access Edge Computing (MEC) and Network Function Virtualization (NFV) emerge as complementary paradigms that shall support Ultra-reliable Low Latency Communication (URLLC) by offering fine-grained on-demand distributed resources closer to the User Equipment (UE), thus mitigating physical layer issues. On the other hand, the adoption of the NFV-MEC inevitably raises deployment and operation costs. We have addressed the combination of MEC, NFV and dynamic virtual resource allocation in order to overcome the problem of resource dimensioning in a special scenario were MEC infrastructure is mounted over Unmanned Aerial Vehicles (UAVs) in the context of URLLC. First, a Continuous-time Markov Chain (CTMC)-based model was proposed to characterize dynamic virtual resource allocation in the MEC node together with four performance metrics that are both relevant for URLLC applications (e.g., reliability and response time) and for service providers (e.g., availability and power consumption). In order to yield the model more practical, the effect of virtual host resource failures, setup (repair) times and processing overheads were embedded into the formulation, since they may significantly affect the stringent requirements of URLLC applications. Moreover, a multi-objective problem related to MEC-enabled UAV node dimensioning in terms of virtual resources (VMs, containers and buffer positions) was formulated. In this context, the compromise between on-board computation resources and the URLLC requirements become a great challenge since UAVs are limited due to their size, weight and power, which imposes a burden on the conventional Network Functions (NFs). Finally, an approach based on Genetic Algorithms (GA) was formulated to solve the dimensioning problem, with the proposed scheme achieving a better tradeoff in terms of availability, reliability, power consumption and response time compared to the commonly adopted approaches based on the First-fit strategy.

Keywords: multi-access edge computing; network function virtualization; resource allocation.

RESUMO

A Computação de Borda Multiacesso (MEC) e a Virtualização de Funções de Rede (NFV) surgem como paradigmas complementares que devem suportar a Comunicação de Baixa Latência Ultraconfiável (URLLC), oferecendo recursos distribuídos sob demanda de forma granular e mais próximos do Equipamento do Usuário (UE), mitigando assim os problemas da camada física. Por outro lado, a adoção do NFV-MEC inevitavelmente eleva os custos de implantação e operação devido a distribuição dos recursos. Abordamos a combinação de MEC, NFV e alocação dinâmica de recursos virtuais para superar o problema de dimensionamento de recursos em um cenário especial onde a infraestrutura MEC é montada sobre Veículos Aéreos Não Tripulados (UAVs) no contexto de URLLC. Primeiro, um modelo baseado em Cadeias de Markov de Tempo Contínuo (CTMC) foi proposto para caracterizar a alocação dinâmica de recursos virtuais no nó MEC juntamente com quatro métricas de desempenho que são relevantes tanto para aplicações URLLC (por exemplo, confiabilidade e tempo de resposta) quanto para provedores de serviços (por exemplo, disponibilidade e consumo de energia). Para tornar o modelo mais prático, o efeito de falhas de recursos de host virtual, tempos de configuração (reparo) e sobrecargas de processamento foram incorporados à formulação, uma vez que podem afetar significativamente os requisitos rigorosos de URLLC. Além disso, foi formulado um problema multiobjetivo relacionado ao dimensionamento de nós UAV habilitados para MEC em termos de recursos virtuais (VMs, contêineres e posições de buffer). Nesse contexto, o compromisso entre os recursos computacionais de bordo e os requisitos de URLLC torna-se um grande desafio, uma vez que os UAVs são limitados devido ao seu tamanho, peso e potência, o que impõe um ônus às funções de rede (NFs) convencionais. Por fim, uma abordagem baseada em Algoritmos Genéticos (GA) foi formulada para resolver o problema de dimensionamento com os esquemas propostos alcançando um melhor compromisso em termos de disponibilidade, confiabilidade, consumo de energia e tempo de resposta em comparação com as abordagens baseadas na estratégia First-fit que é comumente usada por outros autores.

Palavras-chaves: computação de borda multiacesso; virtualização de funções de rede; alocação de recursos.

LIST OF FIGURES

Figure 1 – Reference architecture for MEC in NFV 22
Figure 2 – Causes of 100 Major Public Outages in Datacenters 24
Figure 3 – Multiple edge node deployment scopes. 35
Figure 4 – Hybrid VM-Container edge node infrastructure 42
Figure 5 – State Transition Diagram 44
Figure 6 – Simulation Example for n=1, c=2 and k=3 45
Figure 7 – Simulation Flow 51
Figure 8 – Part of the simulator logic 52
Figure 9 – Availability 54
Figure 10 – Reliability 54
Figure 11 – Power Consumption 55
Figure 12 – Response Time 55
Figure 13 – Availability 57
Figure 14 – Reliability 57
Figure 15 – Power Consumption 57
Figure 16 – Response Time 57
Figure 17 – Availability 58
Figure 18 – Reliability 58
Figure 19 – Power Consumption 59
Figure 20 – Response Time 59
Figure 21 – Space state diagram 64
Figure 22 – Availability 71
Figure 23 – Reliability 71
Figure 24 – Power Consumption 72
Figure 25 – Response Time 72
Figure 26 – Availability 73
Figure 27 – Reliability 73
Figure 28 – Power Consumption 73
Figure 29 – Response Time 73
Figure 30 – Availability 74

Figure 31 – Reliability 74
Figure 32 – Power Consumption 75
Figure 33 – Response Time 75
Figure 34 – Data Plane architecture mapping example - 3GPP 5G system and MEC-NFV 79
Figure 35 – Test Cases for Pc and Pm 83
Figure 36 – GA Scheme Execution Flow 84
Figure 37 – GA Convergence Test 85
Figure 38 – Availability 88
Figure 39 – Reliability 88
Figure 40 – Power Consumption 89
Figure 41 – Response Time 90

LIST OF TABLES

Table 1 – Chapter 2 Summary 32
Table 2 – Resource Allocation Problems and Solution Strategies 35
Table 3 – Edge Position in Related Work 36
Table 4 – Virtual Resource and Resource Assumptions in Related Work 38
Table 5 – Performance Metrics Adopted in Related Work 39
Table 6 – Summary of Existing Related Work 40
Table 7 – Chapter 3 Summary 41
Table 8 – Balance Equations Summary 47
Table 9 – Power Consumption Notation 49
Table 10 – Log description 53
Table 11 – Simulation Parameters 54
Table 12 – Reliability and Response Time Thresholds for URLLC 56
Table 13 – Chapter 4 Summary 60
Table 14 – Balance Equations Summary 67
Table 15 – Power Consumption Values 70
Table 16 – Experiment Sets 70
Table 17 – Chapter 5 Summary 76
Table 18 – Crossover rate and Mutation Probability Tests 83
Table 19 – Selected GA Parameters 83
Table 20 – Scheme Evaluation Parameters 86
Table 21 – Chapter 6 Summary 90
Table 22 – Publication List 92

LIST OF ABBREVIATIONS AND ACRONYMS

1G	First Generation
3GPP	3rd Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
ACS	Aerial Control System
AF	Application Function
AMF	Access and Mobility Function
AR	Augmented Reality
CaaS	Container-as-a-Service
CN	Core Network
CT	Container
CTMC	Continuous-time Markov Chain
DN	Data Network
DRA	Dynamic resource allocation
EMBB	Enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
FCFS	First Come, First Served
GA	Genetic Algorithms
IaaS	Infrastructure as a Service
LOS	Line-of-Sight
MEAO	MEC application orchestrator
MEC	Multi-access Edge Computing
MEPM-V	MEC platform manager
MME	Mobility Management Entity
NEF	Network Exposure Function

NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestration
OS	Operating System
PM	Physical Machine
QoS	Quality of Service
RAN	Radio Access Networks
SLA	Service Level Agreement
SM	Smart Manufacturing
SMF	Session Management Function
SNC	Stochastic Network Calculus
TI	Transport Industry
UAV	Unmanned Aerial Vehicle
UE	User Equipment
uHPC	Ultra-high precision communication
uMBB	Ultra mobile broadband
UPF	User Plane Function
URLLC	Ultra-Reliable Low-Latency Communications
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function

LIST OF SYMBOLS

α	Setup Rate
γ	Failure Rate
μ	Service Rate
μ_v	VM Service Rate
μ_c	CT Service Rate
λ	Arrival Rate
π	Steady-state Probability Vector
Ω	Feasible Space State
ω_r	Reliability constant
ω_t	Response Time constant
P_{idle}^{VM}	Idle VM Energy Consumption
P_{idle}^{VM}	Busy VM Energy Consumption
P_{idle}^{CT}	Idle CT Energy Consumption
P_{setup}^{CT}	Setup CT Energy Consumption
P_{busy}^{CT}	Busy CT Energy Consumption
δ	delta auxiliary variable
ϵ	epsilon auxiliary variable
ζ	zeta auxiliary variable
η	eta auxiliary variable
θ	theta auxiliary variable

CONTENTS

1 INTRODUCTION . . . 17
1.1 OBJECTIVES . . . 19
1.2 DOCUMENT ORGANIZATION . . . 19
2 TECHNICAL BACKGROUND . . . 20
2.1 KEY TECHNOLOGIES FOR A VIRTUALIZED EDGE/CORE . . . 20
2.1.1 Multi-access Edge Computing (MEC) . . . 20
2.1.2 Network Function Virtualization (NFV) . . . 21
2.1.3 NFV-MEC Architecture . . . 21
2.1.4 Resource Allocation in NFV-MEC . . . 23
2.1.5 Edge/Core Failures . . . 24
2.2 QUEUEING THEORY . . . 25
2.2.1 Concept and Notation . . . 25
2.2.2 Types of Queues . . . 26
2.2.3 Continuous-Time Markov Process . . . 27
2.3 MULTI-OBJECTIVE OPTIMIZATION . . . 28
2.4 GENETIC ALGORITHMS . . . 29
2.4.1 Chromosome Encoding and Fitness Function . . . 30
2.4.2 Selection, Crossover and Mutation Operators . . . 31
2.5 CHAPTER SUMMARY . . . 32
3 RELATED WORK . . . 33
3.1 RESOURCE ALLOCATION PROBLEMS . . . 33
3.2 EDGE POSITION . . . 35
3.3 VIRTUAL HOST TYPES AND ASSUMPTIONS . . . 37
3.4 PERFORMANCE METRICS . . . 38
3.5 CHAPTER SUMMARY . . . 41
4 SINGLE NODE LOCATION-AGNOSTIC NFV-MEC MODEL . . . 42
4.1 SYSTEM MODEL . . . 42
4.2 ANALYTICAL MODEL . . . 43
4.2.1 System Example . . . 44
4.2.2 Steady State Analysis . . . 45

4.3 PERFORMANCE METRICS . . . 48
4.3.1 Availability (A) . . . 48
4.3.2 Reliability (R) . . . 48
4.3.3 Power Consumption (C) . . . 49
4.3.4 Response Time (T) . . . 50
4.4 MODEL VALIDATION . . . 51
4.4.1 The Discrete-event Simulator . . . 51
4.4.2 Validation Results . . . 53
4.4.2.1 Multiple edge sizes (k) . . . 53
4.4.2.2 Multiple VM/Container Arrangements (n, c) . . . 56
4.4.2.3 Multiple Setup and Failure Rates (α, γ) . . . 58
4.5 CHAPTER SUMMARY . . . 60
5 MODEL EXTENSION . . . 61
5.1 SYSTEM MODEL . . . 61
5.2 ANALYTICAL MODEL . . . 63
5.3 PERFORMANCE METRICS . . . 67
5.3.1 Availability (A) . . . 67
5.3.2 Reliability (R) . . . 68
5.3.3 Power Consumption (C) . . . 68
5.3.4 Response Time (T) . . . 69
5.4 MODEL VALIDATION . . . 69
5.4.1 Multiple VM Amounts (n) and Overhead Degradation Factor (d) . 70
5.4.2 Multiple Container Amounts (c) and Failure Rates (γ) . . . 72
5.4.3 Multiple Queue Sizes (q) and Container Setup Rates (α) . . . 74
5.5 CHAPTER SUMMARY . . . 75
6 A SCHEME BASED ON GENETIC ALGORITHMS FOR MEC NODE DIMENSIONING . . . 77
6.1 MEC-ENABLED UAVS . . . 77
6.2 MEC NODE DIMENSIONING OPTIMIZATION PROBLEM . . . 80
6.3 GA OPERATIONAL DETAILS . . . 81
6.3.1 Chromosome Structure and Fitness Function . . . 81
6.3.2 Genetic Operators and Parameters . . . 82
6.3.3 GA Scheme Execution Flow . . . 84

6.4 SCHEME EVALUATION 84

6.4.1 GA Convergence 85

6.4.2 Evaluation Scenarios 85

6.4.3 Numerical Results 87

6.4.4 Evaluation Scenarios 87

6.5 CHAPTER SUMMARY 90

7 CONCLUSION AND FUTURE WORKS 91

7.1 FINAL CONSIDERATIONS 91

7.2 CONTRIBUTIONS 91

7.2.1 Publication List 92

7.3 FUTURE WORKS 93

7.3.1 Related Mathematical Models 93

7.3.1.1 Reducing model computational complexity 93

7.3.1.2 Edge and Central Cloud Representation 93

7.3.1.3 Multiple Service Classes 93

7.3.1.4 Precision Models 94

7.3.2 Optimization Algorithms for Multi-Objective Problems 94

7.3.2.1 Bio-Inspired Approaches 94

7.3.3 Other Approaches Towards MEC-NFV Environment 94

7.3.3.1 Probability Distributions 94

7.3.3.2 Testbed and Simulation 94

REFERENCES 95

1 INTRODUCTION

Mobile communications have experienced dramatic technological changes in short intervals since the First Generation (1G) and following up until the Fourth Generation (4G). However, from 1G to 4G the number of end users was generally bounded by the size of a given population in a region, whereas the Fifth Generation (5G) and future networks focus on both human and non-human interactions, e.g., inter-machine communications [Arshad and Kashif 2019]. Yet, this shift is not only related to the increasing amount of traffic, but with a myriad of conflicting requirements (e.g., reliability and latency) that emerge with a set of applications under the Ultra-Reliable Low-Latency Communications (URLLC) use case. URLLC is a service category introduced by the 5G, which is expected to concomitantly deliver end-to-end reliability of up to 99.999% and packet latencies less than 1ms, often being assigned to critical applications such as wireless factory automation [Feng et al. 2019].

In this context, Multi-access Edge Computing (MEC) extends the notion of cloud computing to the customer's premises, wherein part of the URLLC services would be processed at the network edge, i.e., physically located close to the user. This allows an application to be hosted in a private network, directly accessible from the core network, which simplifies the design of both network and transport layer protocols since the flow and congestion controls are handled by the application layer, lowering response times and enhancing reliability. On the other hand, the resources in MEC nodes are limited compared to that in regular datacenters. Thus, the utilization of these limited resources is usually associated with the concept of Network Function Virtualization (NFV), which allows network operators to efficiently allocate resources for Virtual Network Function (VNF) according to the MEC usage [Gundall et al. 2021].

NFV paves the way for the cloudification trend by enabling VNF, i.e., network function (e.g. Mobility Management Entity) decoupling from dedicated hardware, providing substantial cost reductions by leveraging the use of physical hardware and allowing seamless elastic resource provisioning (VNF scaling) [Bi et al. 2019]. Nonetheless, to leverage the advantages of the joint use of MEC and NFV in the URLLC domain several challenges have to be solved. For instance, the virtualization layer that supports these technologies are expected to handle VNF loading and faults without breaking the stringent URLLC requirements. Thus, besides under and over-provisioning issues that may respectively cause severe Service Level Agreement (SLA) violations and increased edge costs (e.g., power consumption), another concern resides on the URLLC sensibility towards extra delays that can be caused by unexpected faults such as hardware/software failures and even resource boot, setup or migration delays that usually depend on the available resources and image size.

Multiple works have compared VMs to containers in the web context but performance analysis for the mobile MEC-NFV is still in its infancy [Doan et al. 2019]. In general, it is said that communication software needs to be specifically redesigned for cloud/MEC environments, specially with regards to URLLC [Pocovi et al. 2018]. Although microservice-based container is put forward as a natural solution due to their enhanced provisioning, still they are currently more failure-prone than the stable Virtual Machine (VM) [Li et al. 2017]. Despite the promised benefits, this research endeavor aims to evaluate the impact of resource allocation in MEC nodes, taking into account part of the expected burden related to the virtualization layer. In particular, we have identified that virtual resource failures have been neglected by the literature, along with VM processing overheads and initialization delays [Ren et al. 2016]. Hence, this work details the interactions regarding the MEC context and concomitantly addresses dynamic resource allocation for critical applications such as those in URLLC [Ji et al. 2018].

Moreover, knowing that URLLC is typically prone to dynamic and continuous shifts in device density, position and services types [Filippou et al. 2020], we provide a node dimensioning scheme for a particular scenario where the edge servers are mounted over Unmanned Aerial Vehicle (UAV). As opposed to the fixed MEC infrastructure, MEC-enabled UAVs yield a flexible solution to meet the dynamic demands of ultra-dense networks [Islambouli and Sharafeddine 2019]. Owing to the mobility, UAVs enable edge servers/core functions to fly closer to UEs, assuring the best connectivity conditions towards propagation quality and consequently higher transmission rates and reliability [Li et al. 2019], but is challenged due to the limited resource and battery life, which leads to the importance of efficient resource dimensioning.

While there has been significant attention paid to latency and energy consumption aspects of MEC-enabled UAVs based on trajectory optimisation, computing failure resilience and resource availability has received far less attention even though these are of paramount importance for resource dimensioning, especially considering critical applications. Motivated by this gap, we propose a framework that allows a service provider to optimize a given MEC-enabled UAV node dimension aiming at maximizing the system's availability and minimizing power consumption while also satisfying reliability and latency constraints, offering a complementary solution to those proposed by previous authors.

1.1 OBJECTIVES

The main objective of this work is to address the combination of MEC, NFV and dynamic virtual resource allocation in the context of URLLC in order to overcome the problem of resource dimensioning in MEC-enabled UAVs. The following specific aims have been defined to achieve this objective:

- Identify the main mechanisms/events related to virtualization layer elements that may affect URLLC application performance.
- Model and validate virtual resource allocation in the MEC-NFV node considering a hybrid virtualization layer.
- Propose a multi-objective problem in the context of MEC-enabled UAV node dimensioning in terms of virtual resources (VMs, containers and buffer positions) for URLLC.
- Propose a scheme to solve the multi-objective problem.

1.2 DOCUMENT ORGANIZATION

This thesis is organized as follows: Chapter 2 examines the technical background, which includes the key enabling technologies envisioned for future mobile communications namely MEC and NFV. This chapter also includes an overview on the main mathematical tools used in this document, namely queueing theory, multi-objective optimization, and Genetic Algorithms (GA). In Chapter 3, a review of the current literature on the topic of resource allocation for MEC-NFV is described. It includes the main features of these works and a short classification. Moreover, in Chapter 4 a single MEC-NFV node is described together with multiple assumptions regarding the virtual environment and the proposed analytical formulation that models these assumptions. Chapter 5 describes some extensions on the model described in Chapter 4 and its validation results. The formulation of the multi-objective problem of resource dimensioning is expressed in Chapter 6, which also encompasses the proposed GA-based scheme for solving it. The simulation and results of the analysis are discussed in the same Chapter. Finally, Chapter 7 provides our concluding remarks and highlights future work directions.

2 TECHNICAL BACKGROUND

This Chapter outlines the background and basic concepts for better understanding this document. It sets out in section 2.1 by addressing the reasons for adopting MEC and NFV in future mobile communications, and examines some features of these technologies. This section also illustrates how resource allocation plays a pivotal role in the NFV-MEC environment and details some of its issues. Next, the basic concepts about queuing theory (section 2.2), multi-objective optimization (section 2.3) and genetic algorithms (section 2.4) are discussed.

2.1 KEY TECHNOLOGIES FOR A VIRTUALIZED EDGE/CORE

This section presents the concept, characteristics and main functionalities of the main pieces of technology related to the virtualized edge: MEC and NFV.

2.1.1 Multi-access Edge Computing (MEC)

Next-generation networks are expected to operate at rates up to Tbps and delays lower than 1ms, e.g., URLLC. In order to achieve such stringent constraints, servers are required to minimize response time and failures concomitantly and being so, the literature strengthens this fact by reinforcing the idea of avoiding excessive backhaul delays, with servers being physically placed closer to the end-user, a phenomenon named as Edge Computing. Two technologies emerged as candidates: Fog computing and MEC [Khan 2017]. The former is expected to run at the edge router, while the latter, supported by the European Telecommunications Standards Institute (ETSI), is expected to run close to base stations. MEC is typically characterized by the following attributes: (1) physical proximity between server and user, (2) location awareness and (3) context information. MEC hosts (i.e., small computing infrastructures) are expected to be deployed on the so called network edges, that is, placed only one or two network hops away from the users, guaranteeing support to time sensitive applications such as augmented reality, object recognition and critical data processing.

MEC is said to be mature compared to FoG mainly for two reasons: (1) Some tools have already been built for its deployment, for instance, Open Source Mano and OpenBaton orchestration suites [Kekki and Featherstone 2018]. (2) ETSI has recently updated the reference architecture to run MEC entities in a NFV environment (ETSI Group Report MEC 017). Hence, for now, MEC turns to be the best candidate for URLLC. MEC is not only reasonable for the end users attached to the edges, but also for the backhaul network, since core decongestion is a natural side effect.

2.1.2 Network Function Virtualization (NFV)

NFV is a carrier-driven initiative for virtualizing network functions (VNF) e.g. switches, routers, IDSs and NATs, using virtual machines and/or containers on standard servers instead of proprietary single-purpose network devices [Zhao et al. 2021]. Two advantages related to VNFs are (1) the easiness of relocating network functions on different locations, not requiring new hardware besides reducing operational and (2) task optimization, task scheduling and resource allocation given a limited amount of available computing resources, as for each service request that arrives, the required VNF service and/or processing can also differ.

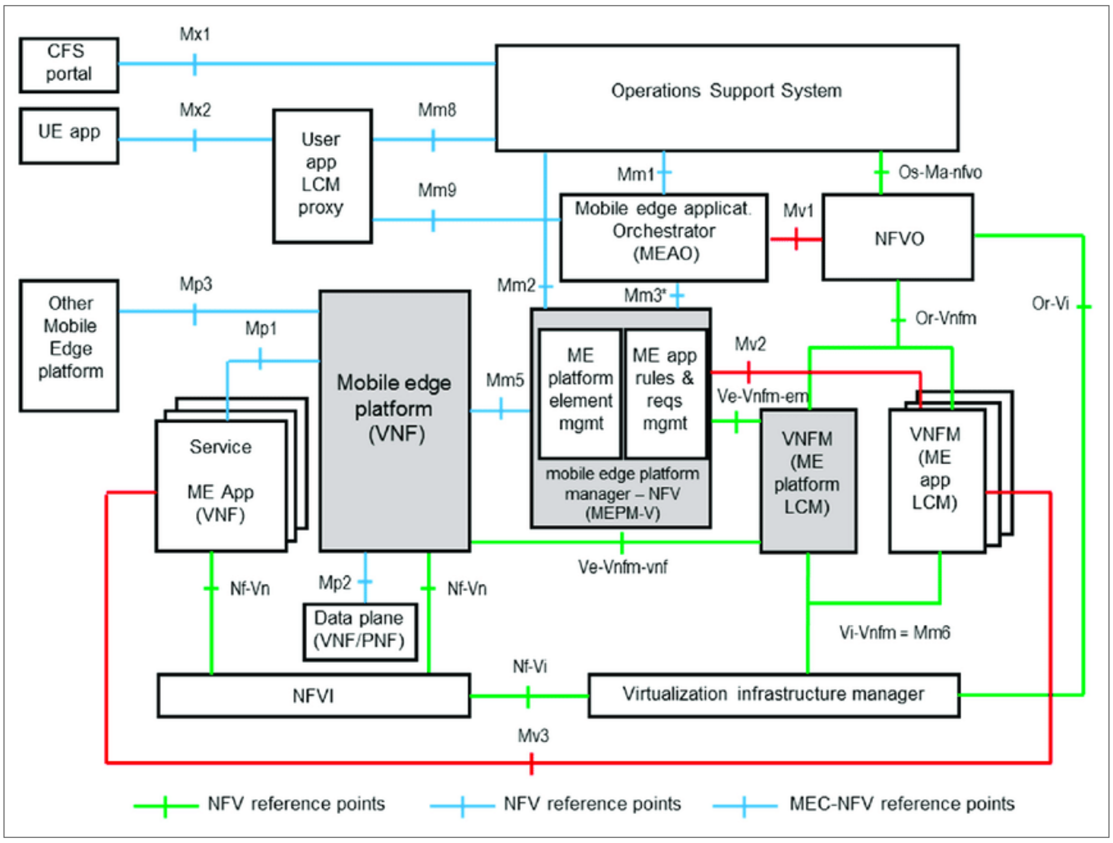
The basic idea is that only one VNF is dedicated to a single request (service on-demand model). The combined use of NFV and MEC should allow a scalability increase, as it facilitates on demand resource scaling. Network Function Virtualization should support the mobile cloudification trend by enabling mobile network functions, e.g., Mobility Management Entity (MME) to be decoupled from dedicated hardware [Mijumbi et al. 2016]. According to ETSI, MEC can use the NFV Infrastructure (NFVI) as the virtualization platform to run mobile edge applications alongside other VNFs. Therefore, MEC applications also appear as VNFs and parts of edge orchestration can be delegated to the NFV Orchestration (NFVO).

2.1.3 NFV-MEC Architecture

MEC and NFV can be seen as complementary concepts, especially because their combination should allow a scalability increase [Kekki and Featherstone 2018], facilitating on demand resource allocation. The MEC architecture has been designed in such a way that a number of different deployment options of MEC systems are possible. In this section, we explore the nuances of a recent architecture proposed by the ETSI that encompasses both MEC and NFV and furthermore we discuss resource allocation under this new paradigm.

A dedicated Group Report (ETSI GR MEC 017) has recently provided details of MEC deployment in an NFV environment, which allows MEC application and VNF instantiation on the same virtualization infrastructure. Fig. 1 details the entities for both the traditional MEC and NFV-MEC. The first consists of MEC hosts and the MEC management that are necessary to run MEC applications within an operator network. The compute, storage, and network resources are provided by a virtualization infrastructure in the MEC platform. The MEC platform is the collection of essential functionality required to run MEC applications on a particular virtualization infrastructure and enable them to provide and consume MEC services. MEC applications are instantiated on the virtualization infrastructure of the MEC host based on configuration or requests validated by the MEC management, which comprises the MEC system level management and the MEC host level management. The MEC system level management includes the Multi-access edge orchestrator as its main component, which has an overview of the complete MEC system. Lastly, both MEC platform manager and virtualization infrastructure manager are built in the MEC host level management, which handles the management of a particular MEC host specific functionality.

Figure 1 – Reference architecture for MEC in NFV



Source: ETSI (2019)

The assumptions for the NFV-MEC are as follows: (1) MEC platform is deployed as a VNF. (2) MEC applications appear as VNFs towards the ETSI NFV MANO components. (3) The virtualization infrastructure is deployed as an NFVI and is managed by a Virtual Infrastructure Manager (VIM). (4) A MEC platform manager (MEPM-V) delegates the VNF lifecycle management to one or more VNF managers (VNFM). (5) A MEC application orchestrator (MEAO) relies on the NFVO for resource orchestration and for orchestration of the set of MEC application VNFs as one or more network services [Kekki and Featherstone 2018].

2.1.4 Resource Allocation in NFV-MEC

Resource allocation is a complex problem that has been subject of a great deal of effort since the first network designs. In general, resource allocation accounts for the simple objective of allocating the minimum amount of assets to support a given performance requirement, often also being associated to operational cost minimization and scalability. Compared to the traditional mobile or cloud architecture, resource allocation in the context of future mobile networks also has to be responsive in order to accommodate workload variations [Zhao et al. 2021].

Resource allocation problems can also be analytically formulated rather than studied in real environments, allowing greater objective freedom. Due to similar analytical approaches and objectives, we can cite datacenter VM placement problems [Rochwerger 2009], which include latency [Alicherry 2012], energy [Beloglazov 2012] and network cost minimization. These objectives could be achieved in many different ways, e.g. load balancing and data and end-user locality [Agarwal et al. 2010]. To find optimality, the problem is usually formulated as NP-hard, so various heuristics to solve the problem have been proposed.

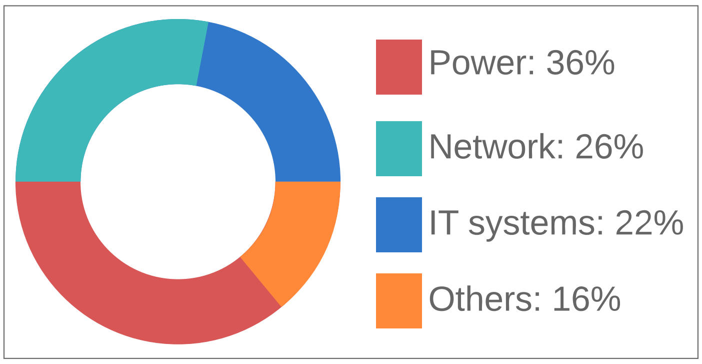
In MEC, the objectives for resource allocation problems can include (1) Round Trip Time (RTT) [Tong 2016], or (2) cost minimization [Chen 2017], (3) edge resource [Skarlat 2017] or (4) provider revenue maximization [Guo 2018]. Various constraints can be used in these different objectives, depending on the relevance for a particular case. For instance, the RTT minimization objective can take resource capacities [Tan et al. 2017], while the operational costs can include those due to energy consumption [Huang 2016], which may further depend on other factors such as edge or the cloud processing, making prices different [Al-Shuwaili 2017]. The system latency and cost may also account for migration and/or reconfiguration costs.

2.1.5 Edge/Core Failures

Previous sections emphasized the role of MEC and NFV for improved real-time, high-bandwidth and low-latency access to sensitive applications. The promises of enhanced cost-benefits have encouraged both market and researches to propose a number of resource allocation schemes for handling 5G traffic in a similar way to regular traffic in cloud computing environments. Unfortunately, many authors simply adapt previous works on that context, assuming misleading information such as the assumption of failure-proof networks. Indeed, datacenters promise reliability of up to 99.99999% [Ascierto 2019], which would be quite acceptable even for 5G standards. There is a widespread belief that virtualization software advances suddenly made IT services far more resilient, especially when coupled with highly engineered data center operations. However, an Uptime Institute survey [Ascierto 2019] shows that failures and downtimes are still common and possibly even increasing.

According to the Uptime Institute, the "extremely reliable clouds" myth is largely driven by organization reluctance to report failures, resulting in significant investments and/or assumptions without real risk assessments [Ascierto 2019]. The biggest cause of IT service outage is a data center power outage, closely followed by network problems, and then by an IT system failure (Fig. 2). Explanations for high outage rates are not clear, but the increased complexity and interdependencies of different systems and datacenters using complicated management systems may be increasing the number and impact of failures. Furthermore, the study has found that failures tend to occur either during technology change periods or non-updated sites. For the average cost that these failures may cause, there is no standard way for quantifying this aspect since interdependency and multiple locations make this estimate less reliable.

Figure 2 – Causes of 100 Major Public Outages in Datacenters



Source: Uptime Institute (2018)

Based on the current cloud scenario, where datacenter's reliability is far from giving the promised support to basic internet traffic, many risks have been suppressed by previous authors that neglected the possibility of edge/core failures on their scaling, resource allocation and even dimensioning schemes [Ren et al. 2016]. In particular, such risks might severely increase when the misleading assumptions are propagated towards MEC-like architectures; i.e., if today's central clouds with few locations are reported as not fully reliable, even though running over a mature technology (VM, hypervisors), multiple edge clouds running under newly launched NFV-MEC architecture are expected to be even less reliable.

2.2 QUEUEING THEORY

Queueing theory is derived from probability theory and its object of study is the phenomenon of waiting in queues. Although the term is often used to describe the queue's mathematical behavior, it may also be applied to models where queues are not allowed to form [Cooper 1981]. Queueing theory has been widely adopted to study communication features in the early days of telecommunication but has continually being used for modern proposals. This section addresses a review on queueing theory and Continuous-Time Markov process.

2.2.1 Concept and Notation

A general queueing system can be usually described in terms of the following:

- The arrival process: an arrival process is generally characterized by a distribution that describes the amount of costumers that arrive in queue per unit time and the distribution that describes the times between successive customer arrivals (inter-arrival times) [Gross et al. 2008]. The most common Markovian arrival process is a Poisson process where the time between each arrival is exponentially distributed.
- The service time: describes the time a customer spends being served.
- The number of servers: indicates how many servers are considered in the system to attend the customers.
- The system capacity: this parameter specifies the maximum number of customers allowed to stay in the system. This includes customers that may be waiting for service (in a buffer)

and those customers that are currently being served.

- Population size: is the total amount of customers that can enter the system. This parameter can be finite or infinite.
- The service discipline: this parameter defines the policy for service order. The most common disciplines are the First Come, First Served (FCFS), Last Come, First Served (LCFS) and the Static Priorities (SP). The latter selects customers to be served based on pre-defined priorities.
- The preemption discipline can be used in conjunction with the LCFS or Static Priorities. This discipline interrupts or preempts the customer currently being served if there is a higher priority customer in the queue [Bolch et al. 2006].

Kendall's notation has been widely used to represent queueing systems. The symbols A/S/m/N/K/SD are commonly used to describe them, where A indicates the inter-arrival times distribution, S is the service time distribution, m is the number of servers, N is the system capacity, K is the population size and SD is the service discipline [Cooper 1981]. The letter M (Markovian) is used to denote that the inter-arrival times and service times that are exponentially distributed. A queueing system can also be represented in a shorter version considering that the system capacity is infinite, the population size is infinite or the service discipline is First Come, First Served (FCFS).

2.2.2 Types of Queues

The main queueing systems are briefly described as follows:

- M/M/1 queue: Is a single-server queue, widely used to model systems where a single server provides the service to the customers. In this type of queue, the inter-arrival times and service times are exponentially distributed, there is no limitations toward either the population size or the system capacity and the adopted service discipline is FCFS [Jain 1991]. The number of customers in the system denotes its state and the two main parameters are the arrival rate of customers and service rate [Gross et al. 2008].
- M/M/m queue: The M/M/m queue is a multi-server model where the arrival rate distribution is Poisson, the service times have exponential distributions, and there are m

identical servers, where each one has the same service capacity. In this system, if there is at least one idle server, the arriving customer is serviced immediately. Otherwise, the customer may wait in a buffer to be served. The buffer is of infinite size, which implies that there is no limit on the number of customers it can handle [Jain 1991].

- M/M/m/N queue: This system is similar to the previous (M/M/m), but it has a limited amount of users denoted by N, that is, the system capacity is limited [Gross et al. 2008]. If m and N have the same value, thus becoming M/M/N/N, it is assumed that the system has no buffer to hold blocked or interrupted users. Other sources apply a similar notation to indicate this special case, for instance, the M/M/m/0 (Erlang's loss system). They are characterized by m identical servers, Poisson input, Exponential service times, no waiting positions N = 0 and unlimited number of customers. This means that after the system reaches full capacity, all new arrivals are blocked. In such cases, the authors use the effective arrival rate as the difference between the total and the blocked arrival rates.

Moreover, depending on the system to be modeled, a priority discipline can be adopted. There are prioritized queueing systems in which customers with higher priority are selected for service ahead of those with lower priorities, regardless of arrival time. Our research takes the basic M/M/m/N queue with few adaptations in order to cope with the proposed scenarios. For instance, we assume that waiting jobs are served according to FCFS, however, priority feedback is allowed when serving jobs suffer failure. We also admit that server resources may have different provisioning capacities, i.e., their setup times can differ depending on each scenario. We call this model an M/M/m/N setup/failure queue.

2.2.3 Continuous-Time Markov Process

In probability theory, Continuous-time Markov Chain (CTMC) is a collection of variables generally indexed by time, which is a continuous quantity. It follows the Markovian property that future variable distribution is independent of the historical behavior, depending solely on its current state [Kemeny 1960]. A continuous-time Markov chain $(X_t)_t \geq 0$ is defined by:

- a finite state space Ω
- a transition rate matrix Q with dimensions equal to that of Ω; and
- an initial state S such that $X_0 = S$ or a probability distribution for this first state.

For $i \neq j$, the elements q_{ij} are non-negative and describe the rate of the process transitions from state i to state j. The elements q_{ii} could be chosen to be zero, but for mathematical convenience a common convention is to choose them such that each row of Q sums to zero. Most CTMCs properties follow directly from the results about its discrete version, the Poisson process and the exponential distribution [Norris 1998].

Moreover, the stationary analysis for a CTMC gives the probability distribution to which the process converges for large values of time units. In brief, a CTMC may be a powerful tool for forecasting a system's stationary state probability (π). The stationary distribution is found by solving $\pi\ Q = 0$, subject to the sum of the elements that must equal 1 [Kemeny 1960].

2.3 MULTI-OBJECTIVE OPTIMIZATION

A multi-objective problem is a problem with two or more objectives that need to be optimized, i.e., maximized or minimized simultaneously, which are often conflicting with each other. Hence, multi-objective problems are related to alternative and incremental strategies that are usually suitable for medium-size or large scale optimization problems. In addition, there can also be constraints to be satisfied so as to solution to be considered feasible [Deb 2011].

Being Inf_i and Sup_i the inferior and superior bounds (decision space) for the decision variable x_i, and x an array of I decision variables $x = [x_1, x_2, ...x_I]^T$, a multi-objective problem can be defined as in (2.1). Please note that f_m denotes a specific objective that should be optimized, whereas the inequalities/equality represent constrained functions. Thus, a solution x is only feasible if all constraints and limits are satisfied simultaneously.

$$\begin{aligned} Maximize/Minimize\ f_m(x) \\ Subject\ to\ g_j(x) \geq 0, \\ h_k(x) = 0, \\ Inf_i \leq x_i \leq Sup_i \\ with\ m = 1, 2, ..., M\ \ j = 1, 2, ..., J\ \ \ k = 1, 2, ..., K\ and\ i = 1, 2, ..., I \end{aligned} \tag{2.1}$$

Conflicting objectives are common in this type of problem. In such case, a single solution that optimizes all objectives is usually not practical. Hence, the concept of dominance may be employed to find solutions that result in reasonable trade-offs between objectives of problem. For instance, in the Pareto's dominance, two feasible solutions are compared to know whether one dominates the other. In other words, given two feasible solutions x and y, x is said to dominate y if the following are satisfied [Deb 2011]: (1) The solution x is no worse than y in all objectives and (2) The solution x is strictly better than y in at least one objective.

In Pareto dominance-based multi-objective evolutionary algorithms, Pareto dominance can sometimes not be effective since it may fail to provide adequate selection pressure, to push convergence in a faster pace [Palakonda and Mallipeddi 2017]. To enhance convergence, multiple authors have been using the idea of a secondary criterion such as knee points. In this work, we propose to employ an adapted version of the classical Pareto Dominance (Section 6.3.1), where the solutions are ranked according to the number of individual objectives that dominate another solution. In particular, we have used equal weights, but different weights could also have been applied. Besides aiding convergence, this allows better variability, since a weaker solution may still earn few points and not be given as completely inferior to the other solution in pair comparison.

2.4 GENETIC ALGORITHMS

Optimization problems can be approached as searching problem where an algorithm is proposed to search for feasible solutions. Evolutionary algorithms have been utilized to solve such problems, including multi-objective problems. GA is a search heuristic and optimization technique inspired by the evolution theory, in which the most adapted individuals are likely survive and pass on their genes [McCall 2005].

Some characteristics of the GA make its adoption interesting for the problem to be described in Chapter 6. Differently from other methods, it handles multiple solutions simultaneously at each interaction and evolves them in order to achieve better solutions. Thus, in our case, many possible resource configurations are evaluated at each interaction.

In general, GA works well in problems with many objectives or constraints. In addition, GA has been widely adopted in the literature to solve problems in telecommunications and computer networks. In addition, although the GA and evolutionary algorithms in general have slow convergence times, we assume that our scheme is adopted for the MEC node dimensioning phase (see Chapter 6), i.e., before proper operation. In other words, the time to obtain the resulting set of possible resource configurations is not critical.

In a GA, each candidate solution is denoted as an individual or chromosome and has a 'fitness', i.e., a value that expresses its suitability to a given problem [McCall 2005]. During its execution, GA tries out a set of solutions (termed population) simultaneously, and creates new populations out of parts of previous individuals.

A randomly generated initial population is created by the GA within the problem domain. For each generation, individuals are evaluated and evolved by the use of genetic operators (e.g. selection, crossover and mutation), in order to find a most suitable solution. After these are applied, a new population is generated. The process (from evaluation to mutation) is repeated until the stop criterion has been met. When satisfied, the best individual of the last population is selected. Generally, this choice is based on the individual's fitness value. The stop criterion is a condition that can be the number of generations, variability degree of individuals, or predefined values for the fitness.

The GA can be expressed in terms of five main components: (1) chromosome encoding, (2) fitness function, (3) selection, (4) crossover and (5) mutation [McCall 2005]. These standard general components are described below while our adaptations are detailed in Chapter 6.

2.4.1 Chromosome Encoding and Fitness Function

The chromosome expresses the information contained in the problem, representing candidate solutions, i.e., in a chromosome, each position or 'locus' is termed a gene and the value placed in that position is denoted as its 'allele' [McCall 2005]. Information can be coded in multiple forms such as real (decimal base), octal, hexadecimal, or other symbols that differ from numbers (e.g. alphabet), however, the most common coding option is binary encoding, which was adopted in this work. In this type of representation, chromosomes are strings of genes where the allele values assume 0 or 1. The choice of which encoding to adopt must consider how well it matches the problem. In [Katoch 2020], a number of different encoding schemes are outlined, as well as the main kinds of problems they are designed to tackle.

The chromosome encoding contains part of the problem's information, however, much of the meaning of the problem is encoded in the 'fitness function'. The fitness function evaluates the quality of each chromosome, which is denoted as the 'fitness value'. The fitness function guides the evolutionary process in accordance with predefined goals and constraints, and thus exerts a strong influence on the GA's effectiveness [Silva et al. 2014]. Different fitness functions can lead to different solutions, although each has its own fitness landscape.

2.4.2 Selection, Crossover and Mutation Operators

The Selection operator determines which chromosomes will be used for reproduction. There are multiple types of such operator, e.g., roulette Wheel, sequential, tournament, and dominant [Kaya 2011], with the first being most widely used. This operator selects individuals based on their fitness value, i.e., the probability for selection is proportional to the individual's fitness. The proportional value is obtained by dividing the each fitness by the sum of all chromosomes' fitness in the entire population.

Crossover is the process in which chromosomes selected by the selection operator are combined to form new members. The operation is applied to each pair of chromosomes according to a given probability, denoted as crossover probability [McCall 2005]. Once two parent chromosomes have been selected, a uniformly distributed random number between [0, 1] is generated and compared to crossover probability; if the number is greater than the crossover probability, no crossover occurs and one or both parents move unchanged onto the next stage. Otherwise, proper crossover is applied. Multiple crossover operators have been proposed, e.g., one-point, two-point, and uniform [Kaya 2011]. However, some studies have argued that the uniform crossover outperforms the single and two-point [Syswerda 1989] since it may combine features regardless of their relative locations.

The uniform crossover operator adopts a crossover mask that indicates how the parents will be combined to generate the children. It is defined as a binary string and its length is equal to the chromosomes' length. For the first child chromosome, if the ith bit of the mask is 0, then the code of the first parent is used in the position i of this child. Otherwise, the code of the second parent is used in this position. For the second child, if the crossover mask has the bit 0 in the position i, then the code of the second parent is used in the ith position. Otherwise, the code for this position comes from the first parent [Syswerda 1989].

The mutation operator determines the search directions and avoids convergence to local optima by inserting diversity in the process [Kaya 2011]. Applied to each individual after the crossover operator, it alters each position of the chromosome according to a given probability, denoted as mutation probability. Thus, for each position of the chromosome, a random number between 0 and 1 is generated according to a uniform probability distribution and compared to mutation probability. If this value is less than the mutation probability, then the position suffers mutation. In binary coding, the most used mutation operator flips the chromosome position value of from 0 to 1 and vice-versa, and it is termed bit mutation.

2.5 CHAPTER SUMMARY

In this Chapter the main concepts regarding this work were reviewed. First, the concepts related to multi-access edge computing and virtualization were explained. These will be used further in multiple chapters to describe the environment in which future mobile networks will rely in order to fulfill the strict latency and reliability constrains. The concepts on queueing theory were also addressed. These will be used especially in Chapters 4 and 5 to model the proposed MEC node environment and derive performance metrics. Multi-objective optimization was also described in order to assist the formulation of the multi-objective problem of the MEC node dimensioning that is discussed in Chapter 6. The classical approach of genetic algorithms was detailed as one of the possibilities to solve the multi-objective problem. Together with the multiobjective optimization problem, its contents and notation are presented in Chapter 6.

Table 1 – Chapter 2 Summary

Section	**Goal(s)**	**Output(s)**
2.1 KEY TECHNOLOGIES FOR A VIRTUALIZED EDGE/CORE	a) Define MEC and NFV concepts; b) Depict the key components of the NFV-MEC Architecture; c) Explain the Resource Allocation problem in NFV-MEC; d) Explain why Edge/Core Failures can impact URLLC.	Definitions of MEC, NFV and description of the resource allocation in the NFV-MEC context for URLLC.
2.2 QUEUEING THEORY	a) Explain the concept, notation and types of queues regarding Queuing Theory; b) Explain the Continuous-Time Markov Process.	Queueing Theory notations and Continuous-Time Markov Process used in Chapters 4 and 5
2.3 MULTI-OBJECTIVE OPTIMIZATION	Explain the main concept, notation and examples of multi-objective problems.	Formal definition of multi-objective problems discussed in depth in Chapter 6.
2.4 GENETIC ALGORITHMS	Explain GA basic concepts: Chromosome Encoding, Fitness Function, Selection, Crossover and Mutation Operators.	Introductory definitions of GA, useful for understanding part of Chapter 6.

Source: The author (2020)

3 RELATED WORK

There is no current consensus on the size, computational power, virtualization technology nor on the best location of MEC nodes [Santoyo-Gonzalez and Cervello-Pastor 2018]. In fact, MEC has not yet been clearly defined; neither functionally nor physically, and since there is a lack of real-world data for new service categories (e.g., URLLC), multiple works fail to provide adequate considerations in order to evaluate MEC-NFV-enabled networks. Performance analysis of Infrastructure as a Service (IaaS) clouds has drawn considerable attention in the last decade although most authors considered solely VMs and hypervisors. Recently, container-based virtualization has been getting momentum due to many strengths over traditional VMs, but especially for being leveraged by the microservice architecture. This chapter provides an overview of the main analytical models and optimization schemes proposed in the literature to deal with virtualization, particularly with regards to MEC-NFV-enabled networks. It seeks to clarify what already exists in the field and to highlight the main differences from this thesis.

This chapter is structured as follows. In Section 3.1, we discuss the main subcategories of resource allocation problems in the MEC-NFV context. Section 3.2 describes how each author interprets and represents the position of edge servers relative to the User Equipment (UE). Following this, Section 3.3 outlines main assumptions regarding the virtual environment that were taken into consideration in each work. Finally, Section 3.4 details which performance metrics were adopted by the related works and, a general classification of the works is presented.

3.1 RESOURCE ALLOCATION PROBLEMS

A body of existing works on MEC-related computational resource issues encompasses multiple classes of problems. The most common can be categorized in resource placement, scheduling, node dimensioning, and Dynamic resource allocation (DRA) [Li et al. 2021].

Placement and scheduling consider applications with two or more components, one typically running on a cloud (MEC or central cloud) and another on the UE. Regarding this class of problems, [Yala and Frangoudis 2018] proposed a placement optimization problem and a GA-based scheme to solve it, considering two conflicting objectives: minimizing access latency and maximizing service availability. In [Farhadi et al. 2019], the authors consider a dynamic standpoint, i.e., adaptable data placement to serve time-varying demands, while considering system stability and operation costs under communication, computation, and storage constraints.

Edge node dimensioning problems are usually related to the decision on the computational resource characteristics based on a given traffic load, e.g., the total number of servers, processing capacity, and storage. In [Lee 2019], the authors designed a greedy-based search algorithm to find the minimum number of MEC servers considering both delay and workload budget. In [Emara et al. 2021], an analytical model based on queuing theory was proposed together with an optimization problem to identify the optimal number of virtual resources to maximize the task execution capacity by means of first fit strategy to solve it.

Lastly, DRA relates to resource provisioning optimization given both the maximum edge node dimensions and the expected traffic load range, which allows a Service Provider to adjust the existing computational resources dynamically, usually with regards to the load. In [Samanta and Tang 2020], a delay and pricing model to supply equitable resources to UEs and minimize network delay and price was suggested. [Tong et al. 2020] proposed a DRA algorithm that minimizes the end-to-end delay while ensuring the minimum service rate and maximum reliability. Another example can be found in [Sarrigiannis et al. 2020], which investigated a DRA approach accounting to minimize edge SLA violations and maximize the serving users.

In some cases, a single formulation can be applied to multiple problem classes. For instance, in [Kherraf et al. 2019] the author jointly solves 1) a MEC dimensioning sub-problem, 2) an application placement sub-problem, and 3) a workload assignment sub-problem. Besides, other works propose performance analysis in order to provide guidelines for the design of a specific service class as in [Ma et al. 2021], where the authors provide a general formulation based on Stochastic Network Calculus (SNC) but takes the 3GPP standard for URLLC into account when selecting simulation parameters. The main purpose of these types of work is to verify how the parameters interact (e.g., Arrival and Service Rate), however, [Ma et al. 2021] goes further and investigates delay and MEC node dimension optimizations.

Similar to [Ma et al. 2021], in this work the performance analysis of the proposed CTMC-based model is the largest contribution. Besides, we take the 3GPP standard for URLLC into account when selecting simulation parameters to explore their relationship. However, we also formulate a multiobjective problem related to MEC node dimensioning which is similar to [Emara et al. 2021], but instead of the first fit, a GA-based scheme is adopted. Table 2 summarizes the related work contributions in terms of resource allocation problem and adopted mathematical branch and optimization strategies used to solve it.

Table 2 – Resource Allocation Problems and Solution Strategies

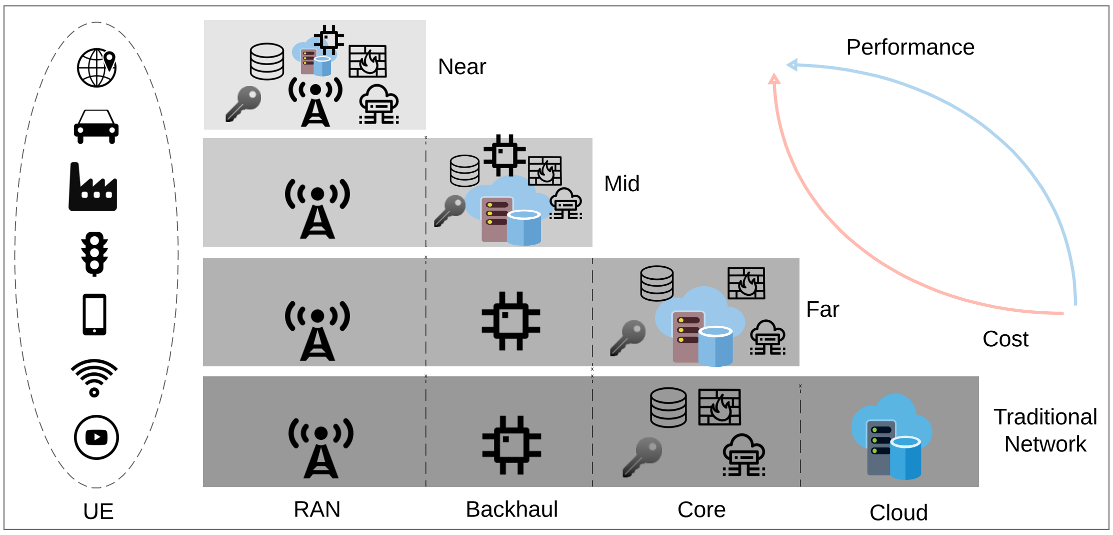
Work	Problem	Mathematical Branch	Optimization Strategy
[Yala and Frangoudis 2018]	Placement	n/a	GA
[Farhadi et al. 2019]	Placement/Scheduling	n/a	Shadow-Algorithm
[Samanta and Tang 2020]	Scheduling/DRA	n/a	Lagrange Multiplier
[Lee 2019]	Dimensioning	n/a	Greedy Search
[Emara et al. 2021]	Dimensioning	Queuing Theory	First Fit
[Tong et al. 2020]	DRA	Grapgh Theory	Subgraph Isomorphism
[Sarrigiannis et al. 2020]	Scheduling/DRA	n/a	Breadth-First
[Kherraf et al. 2019]	Dimensioning	Grapgh Theory	Decomposition approach
[Ma et al. 2021]	Dimensioning/Delay	SNC	n/a
This Work	DRA/Dimensioning	Queuing Theory	GA

Source: The author (2020)

3.2 EDGE POSITION

Under the MEC paradigm, edge nodes can significantly differ in their deployment location. Fig. 3 describes the traditional communication path from UE to a service hosted in a central cloud, which includes: the backhaul, Core Network (CN) and cloud host. The three alternatives exemplify the edge placement variants that could emerge to handle future mobile applications. While the first commercial deployments follow the Far variant, the increasing uncertainty brought by each additional intermediate hop may deeply affect the performance (blue line) for critical applications. In contrast, fine-grained server distribution is known to increase the overall infrastructure cost (red line) and management complexity [Santoyo-Gonzalez and Cervello-Pastor 2018].

Figure 3 – Multiple edge node deployment scopes.



Source: The author (2020)

While some authors place computation capacities within Radio Access Networks (RAN) sites, others prefer a farther away location, similarly to centralized data centers but introducing new components and inter-working procedures to ensure better performance. In [Lee 2019], [Tong et al. 2020] and [Ma et al. 2021], the analysis covers the full path from the RAN cluster to the MEC node, including some core functions and application layer besides considering the edge servers placed exclusively on the Near Scope. Similarly, [Sarrigiannis et al. 2020] and [Samanta and Tang 2020] also consider the Near Scope solely but excludes the RAN analysis focusing only on their optimization schemes for MEC resources. On the contrary, [Yala and Frangoudis 2018] assumes a more flexible approach, allowing an operator to either install the MEC servers close or far away from the end-users. Finally, the analytical model in [Emara et al. 2021] allows a high degree of flexibility since there are separate models for the RAN and MEC, i.e., the RAN model's output flow is one of the inputs to the MEC model. Table 3 summarizes the related work in terms of edge position.

Table 3 – Edge Position in Related Work

Work	**Edge Position**
[Yala and Frangoudis 2018]	Flexible
[Farhadi et al. 2019]	Near
[Samanta and Tang 2020]	Near
[Lee 2019]	Near
[Emara et al. 2021]	Flexible
[Tong et al. 2020]	Near
[Sarrigiannis et al. 2020]	Near
[Kherraf et al. 2019]	Near
[Ma et al. 2021]	Flexible
This Work	Near

Source: The author (2020)

Concerning edge node positioning, the closest works to ours are [Emara et al. 2021] and [Ma et al. 2021]. In summary, both models offer no restrictions towards the total Edge node size nor any underlying considerations that compel the edge node position to a certain location from the UE (Near, Mid, or Far). Hence, it is possible to shift the edge node position towards the UE (Near Scope) or central cloud (Far Scope), alter the total number of resources and other parameters (e.g., processing capacity). However, by doing so, very few variables become overloaded with multiple sub-parts, i.e., the overall service rate becomes a representation of the multiple service rates in each network sub-part. Thus, we indicate that our model is most adequate for the near scope, isolated from the RAN and CN.

3.3 VIRTUAL HOST TYPES AND ASSUMPTIONS

MEC is frequently put forward considering virtualization instead of legacy physical equipment. However, the literature consistently brought regular data center architectural approaches as if they could also be applied seamlessly and without greater modifications to the mobile environment, e.g., [Ren et al. 2016]. For this reason, some works such as [Farhadi et al. 2019], [Lee 2019], [Sarrigiannis et al. 2020], [Tong et al. 2020] and [Ma et al. 2021] are agnostic towards a given virtualization technology, which denotes a certain lack of commitment to the feasibility of their propositions.

Moreover, multiple works have proposed a resource infrastructure built only by physical machine (PM) [Lee 2019], VM as in [Emara et al. 2021] and [Kherraf et al. 2019] or a mix of both as in [Yala and Frangoudis 2018]. However, although NFV has traditionally been implemented over VMs, the concept of Container-as-a-Service (CaaS) has gained momentum. In contrast to VM-based VNFs, CaaS allows VNF instances to be loaded using containers, which are known to consume less computational resources, besides having less instantiation overhead and thus being much more cost-effective [Morabito 2015]. Therefore, some authors consider MEC infrastructure using only containers [Samanta and Tang 2020].

The main issue of solely using containers for future mobile communications is that they are still not mature compared to VMs. There are multiple security risks involved in containerization since all containers in an Operating System (OS) share a single kernel. Hence, any breach on kernel OS can break all containers dependent on it. Besides, isolating faults within containers is not trivial and as a fault can be replicated to subsequent instances. On the other hand, containers can be used along with VMs (hybrid) in NFV environments. In our model, this property is explored as we consider a hybrid VM-containerized infrastructure that leverages the best of both: the VM's strong isolation and the flexibility of containers.

The appropriate selection of the underlying virtual environment directly impacts the feasibility of the NFV-MEC environment. In particular, multiple works have ignored the possibility of faults related to the virtual host technology, which can be key for many of the aforementioned categories, e.g., if a resource dimensioning strategy does not account to the resource failure possibility, the resulting node size will likely be underestimated. In [Ma et al. 2021], for instance, although a reliability-related metric is proposed, the authors recognize that SNC is still not advisable for evaluating reliability. Moreover, [Yala and Frangoudis 2018], [Samanta and Tang 2020] and [Tong et al. 2020] do not account for the repair delay, which impacts resource availability and power consumption. In this context, we believe that an accurate model should account for virtual host failures, repair delays and other events that may cause burdens to the communication process, some of which can be found in [Emara et al. 2021] and in this work. Table 4 adds the information on the virtual resource and their assumptions in each evaluated work (two final columns).

Table 4 – Virtual Resource and Resource Assumptions in Related Work

Work	**Virtual Resource**	**Resource Assumptions**
[Yala and Frangoudis 2018]	VMs	Failure
[Farhadi et al. 2019]	n/a	n/a
[Samanta and Tang 2020]	Container	Failure
[Lee 2019]	n/a	n/a
[Emara et al. 2021]	VM	Failure/Repair
[Tong et al. 2020]	n/a	Failure
[Sarrigiannis et al. 2020]	n/a	n/a
[Kherraf et al. 2019]	VM	n/a
[Ma et al. 2021]	n/a	n/a
This Work	VMs and Containers	Failure/Repair

Source: The author (2020)

3.4 PERFORMANCE METRICS

From the 3rd Generation Partnership Project (3GPP) Release 16 [3GPP 2020] onwards, potential architecture enhancements for supporting URLLC services focused on MEC-NFV have been issued. With regards to the performance metrics, in addition to the most representative metrics for future critical applications (e.g., URLLC), i.e., reliability and latency, the document also includes the MEC resource availability. In the following lines, we have mapped which indicators have been used in the literature.

From the above-mentioned list, the only adopted metric in [Sarrigiannis et al. 2020] was network delay. The works [Ma et al. 2021] and [Tong et al. 2020] only evaluate reliability and latency. In [Kherraf et al. 2019] and [Yala and Frangoudis 2018], the authors only accounted for availability and latency (network delay). Differently, in [Samanta and Tang 2020], besides reliability and a latency-related indicator (Network Delay), energy consumption was also analyzed. Similarly, [Emara et al. 2021] covers availability and reliability with an energy constraint per device. Lastly, in [Lee 2019], both delay and energy constraints are considered, while availability and reliability are left out of scope.

Besides the performance indicators suggested by the 3GPP, some authors also adopt the energy-related indicators. From the user perspective, the suggested indicators should be reasonable, however, from the service provider perspective, infrastructure cost metrics are equally relevant. The problem is that the list of location-dependent costs for building and operating edge nodes can be quite extensive, going from land acquisition to installation expenses. Besides, it is not feasible to numerically map many of these variables since they are not universal. One of the few exceptions is the computational power consumption, which does reflect part of the operational costs. Thus, together with 3GPP metrics, we have included power consumption in the evaluation so as to make our results strongly-coupled with the Service Provider reality. The related work classification in terms of performance metrics is given in Table 5.

Table 5 – Performance Metrics Adopted in Related Work

Work	**Availability**	**Reliability**	**Energy**	**Latency**
[Yala and Frangoudis 2018]	✓	✗	✗	✓
[Farhadi et al. 2019]	✓	✗	✓	✗
[Samanta and Tang 2020]	✗	✓	✓	✓
[Lee 2019]	✗	✗	✓	✓
[Emara et al. 2021]	✓	✓	✓	✗
[Tong et al. 2020]	✗	✓	✗	✓
[Sarrigiannis et al. 2020]	✗	✗	✗	✓
[Kherraf et al. 2019]	✓	✗	✗	✓
[Ma et al. 2021]	✗	✓	✗	✓
This Work	✓	✓	✓	✓

Source: The author (2020)

Contribution We propose a multipurpose analytical model to analyze MEC-NFV environment from the virtualization layer standpoint, which enables service providers to tune multiple network and infrastructure parameters. The model allows the evaluation of traffic behavior in isolation from the RAN and Core networks, not being restricted by the edge node position nor a particular choice of virtual technology. Finally, taking into account the literature review, the proposed model includes a formulation for the main performance metrics related to critical applications: Availability, Reliability, Power Consumption and Latency. The resulting classification, is summarized in Table 6.

Table 6 – Summary of Existing Related Work

Work	Problem	Math. Branch	Optim. Strategy	Edge Position	Virtual Resource	Resource Assumpt.	Availability	Reliability	Energy	Latency
[Yala and Frangoudis 2018]	Placement	n/a	GA	Flexible	VMs	Failure	✓	✗	✗	✓
[Farhadi et al. 2019]	Placement/Scheduling	n/a	Shadow-Algorithm	Near	n/a	n/a	✓	✗	✓	✗
[Samanta and Tang 2020]	Scheduling/DRA	n/a	Lagrange Multiplier	Near	Container	Failure	✗	✓	✓	✓
[Lee 2019]	Dimensioning	n/a	Greedy Search	Near	n/a	n/a	✗	✗	✓	✓
[Emara et al. 2021]	Dimensioning	Queuing Theory	First Fit	Flexible	VM	Failure/Repair	✓	✓	✓	✗
[Tong et al. 2020]	DRA	Grapgh Theory	Subgraph Isomorphism	Near	n/a	Failure	✗	✓	✗	✓
[Sarrigiannis et al. 2020]	Scheduling/DRA	n/a	Breadth-First	Near	n/a	n/a	✗	✗	✗	✓
[Kherraf et al. 2019]	Dimensioning	Grapgh Theory	Decomposition approach	Near	VM	n/a	✓	✗	✗	✓
[Ma et al. 2021]	Dimensioning/Delay	SNC	n/a	Flexible	n/a	n/a	✗	✓	✗	✓
This Work	DRA/Dimensioning	Queuing Theory	GA	Near	Hybrid	Failure/Repair	✓	✓	✓	✓

Source: The author (2020)

3.5 CHAPTER SUMMARY

In this chapter, we highlighted the characteristics, strengths and shortcomings, and performed a qualitative comparison towards the related work, clarifying our thesis proposal contributions. In addition, a classification of the main related works was performed. We took into account aspects such as scenarios in which the proposed models could be used, the types of underlying virtualization technologies and its assumptions towards failure and repair events, besides the adopted performance metrics.

Table 7 – Chapter 3 Summary

Section	**Goal(s)**	**Output(s)**
3.1 RESOURCE ALLOCATION PROBLEMS	Describe and compare the most updated works regarding MEC infrastructure and URLLC.	Placement, Dimensioning, Scheduling and Dynamic Resource Allocation are the most popular topics regarding MEC infrastructure.
3.2 EDGE POSITION	Compare how each author's formulation describe the edge position.	Edge position considerations are usually for the near scope.
3.3 VIRTUAL HOST TYPES AND ASSUMPTIONS	Describe which virtual technology assumptions are taken into consideration by other works.	a) Most authors ignore the underlying virtual technology assumptions; b) Few works consider virtual host problems in their formulations.
3.4 PERFORMANCE METRICS	Discover which performance metrics are the most adequate for the URLLC scenario considering MEC.	Latency, Reliability and Energy Consumption are the most commonly adopted metrics.

Source: The author (2020)

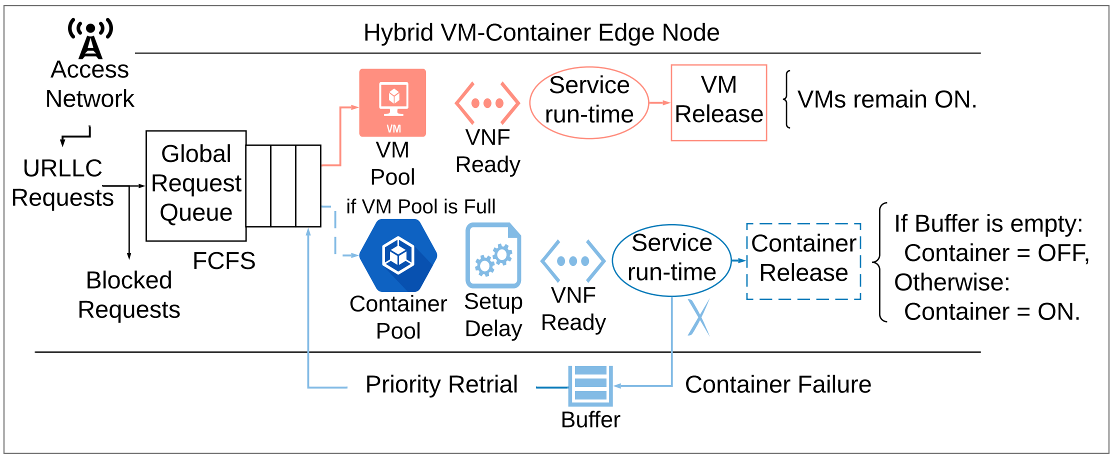
4 SINGLE NODE LOCATION-AGNOSTIC NFV-MEC MODEL

In light of the characteristics described in Chapter 3, this chapter describes a CTMC-based analytical representation for a single node NFV-MEC, assuming a hybrid virtual environment which is prone to failures and repair times. This Chapter is structured as follows. First, Section 4.1 outlines the main events related to the computing model and failure/repair characteristics. Section 4.2 details the formulation/modeling itself, providing the equations and its main conditions. Section 4.3 describes the formulation for the adopted performance metrics and Section 4.4 describes the validation scenarios based on simulation of the proposed model.

4.1 SYSTEM MODEL

Analytical models can be useful tools for rapidly evaluating largely distributed MEC-related infrastructure projects since simulation and testbeds for this kind of experiment, where thousands of Edge Nodes are expected, may not be feasible. In this work, we evaluate a single isolated MEC node depicted in Fig. 4, where requests originated from UEs are processed by the RAN, passed to the MEC node and handled by a VM-hosted (red flow) or containerized VNF (blue flow). This model was designed in isolation from RAN, Core, and Central Cloud, i.e., rather than accounting for multiple network path subparts, the only uncertainty is related to the virtual components, thus having two advantages: (1) the flexibility for adapting to multiple node sizes and (2) the precision due to the limited variables that can affect Quality of Service (QoS).

Figure 4 – Hybrid VM-Container edge node infrastructure



Source: The author (2020)

A dynamic VNF scaling strategy was embedded into our formulation to help cope with the sudden load increase caused by the intensive requests. Each VNF runs equally and independently on a single VM or container, with VMs executing uninterruptedly while containers are scaled upon demand. A centralized control unit determines if requests are admitted or blocked, only activating containers when all VMs are busy.

The containerized VNF activation comprises two phases: initializing the kernel image and launching the specified function, which is interpreted as a single transition interval (setup time), during which power and resources are consumed but no request is processed. Furthermore, active containerized VNFs may suffer failures during attendance, which implies either a service migration to an available VM/container or a reset, triggering a new setup period, with progress being lost only in the latter case.

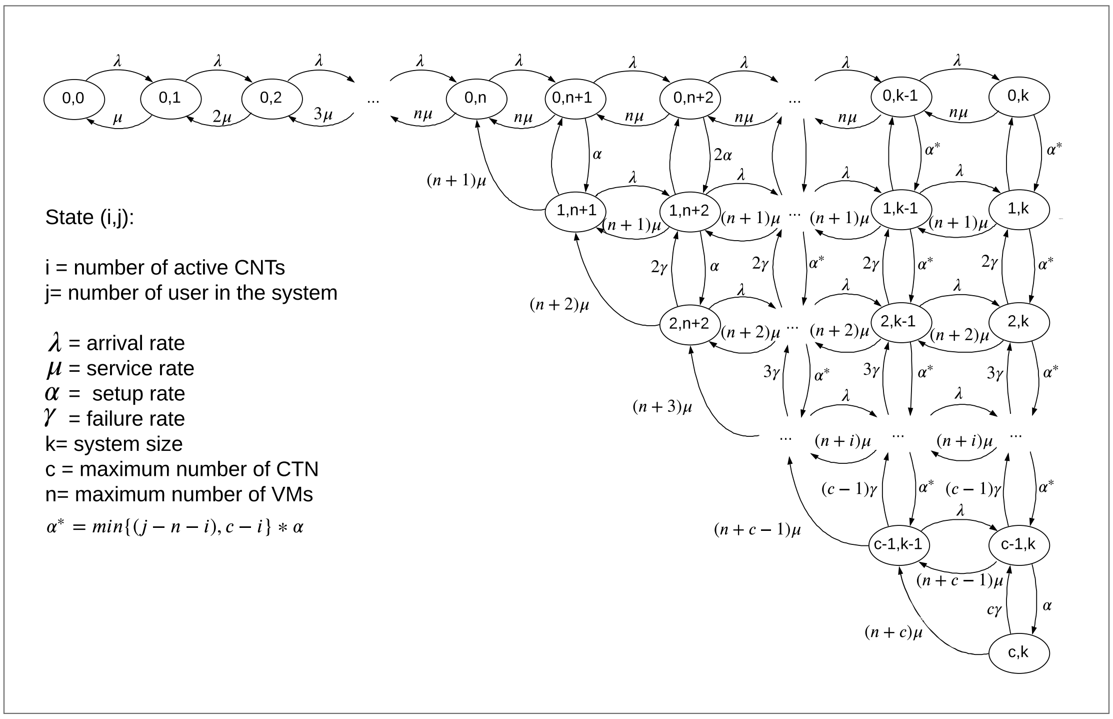
In general, recovery time depends on the failure type; for instance, a software component crash can be rapidly fixed by the host in few microseconds, while others may take milliseconds to reboot device and VNF. Since future mobile networks encompass highly critical applications (e.g., URLLC), only the worst-case scenario is considered. Lastly, as soon as an operative VNF concludes processing and there are no remaining requests, the VNF instance can either be powered down instantaneously together with the host container or remain active if hosted by a VM. The shutdown delay is ignored for being significantly smaller than the setup/recovery magnitudes [Kaur et al. 2017].

4.2 ANALYTICAL MODEL

The system comprises n VMs and c containers, with a maximum limit of k simultaneous services. The service request follows a Poisson process with rate λ (requests/ms) and server capacities of one service with an exponentially distributed service rate μ for both VM-based and containerized VNFs. A Poissonian arrival process was selected for its simplicity and tractability. Control applications are likely to fit a regularly spaced packet trace (isochronous), i.e., a superposition of deterministically spaced and sporadic packet streams, where each contributes to a portion of the overall traffic, which might be well modeled as a Poisson [Anand and Veciana 2018]. Container setup/recovery times and failures are also exponentially distributed with rates α and γ, respectively. A regular FCFS was assumed for new requests with prioritization for retrial due failures.

We assume that the network is a standalone deployment and the system is modeled as an M/M/n+c/k queue (Fig.5) with setup time and failure, $n, c, k > 0$ and $n + c \leq k$ [Kleinrock 1975]. The two-dimension CTMC diagram in Fig.5 includes horizontal flows to the right (left) representing requests (service conclusion) whereas the vertical flows to the top (down) denote the container startup (failures). The diagonal transitions represent both a service conclusion and a container shutdown following our scaling rule. The feasible state space (Ω) is formed by a set of states (i,j), which denotes the number of active containers (i) and online services (j). $\Omega = (i, j) \mid 0 \leq i \leq c$, $0 \leq j \leq k$, provided that $i \leq j - n$. Since VMs are always active regardless of being busy or idle, the states $(0, j)$ with $0 \leq j \leq n$ indicates that the existing load is being processed in VMs, whereas states with $j > n$ imply that in addition to all available VMs, there are requests being processed in containers.

Figure 5 – State Transition Diagram

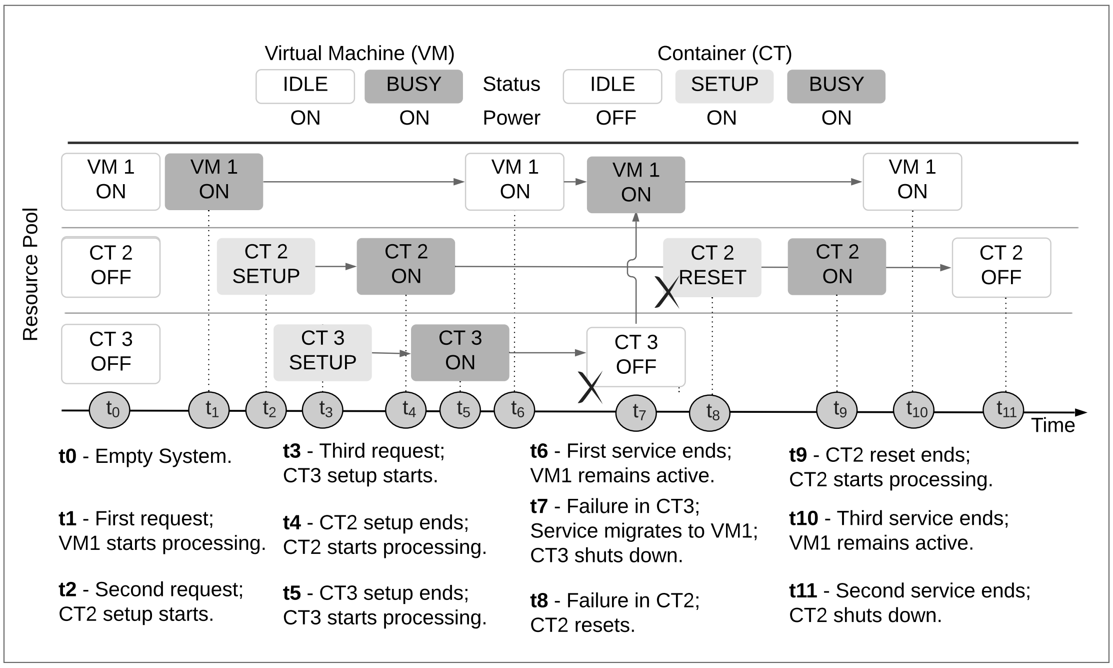


Source: The author (2020)

4.2.1 System Example

A short example for a system formed by one VM, two Containers (CTs) and capacity for three simultaneous services (n=1, c=2 and k=3) is depicted in Fig. 6. The first three events are regular service requests that are allocated to each available resource in t1,t2 and t3, respectively. Each container (CT2 and CT3) requires a setup, hence a waiting period is set until the resource is ready; the service is only processed if the setup is successful, such as in t4 and t5. Conversely, VMs are always available for immediately attending; not requiring waiting periods and ergo begins serving since the task arrival in t1. In t6, VM1 completes processing the first service, becoming available. In t7, a CT2 failure forces the system either to reset or to move the current service to another available resource, but since there no free resources, CT2 is reset. In t8, another failure forces the system to the same situation, except that a resource is available at VM1. Hence, since the delay for moving a service is negligible compared to a Container (CT) reset, the former becomes the best decision, which causes CT3 to shut down as there are no new requests. In t9, CT2 starts processing the same service again after the setup period. Furthermore, the events in t10 and t11 are regular service completions, differing only with regards to the post completion resource status; since there are no new service requests, CT2 is shut down whereas VM1 remains active.

Figure 6 – Simulation Example for n=1, c=2 and k=3


Virtual Machine (VM)
Container (CT)
IDLE
BUSY
Status
IDLE
SETUP
BUSY
ON
ON
Power
OFF
ON
ON
Resource Pool
VM 1 ON
CT 2 OFF
CT 3 OFF
CT 2 SETUP
CT 3 SETUP
CT 2 ON
CT 3 ON
CT 2 RESET
CT 2 ON
CT 2 OFF
t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11
Time
t0 - Empty System.
t1 - First request; VM1 starts processing.
t2 - Second request; CT2 setup starts.
t3 - Third request; CT3 setup starts.
t4 - CT2 setup ends; CT2 starts processing.
t5 - CT3 setup ends; CT3 starts processing.
t6 - First service ends; VM1 remains active.
t7 - Failure in CT3; Service migrates to VM1; CT3 shuts down.
t8 - Failure in CT2; CT2 resets.
t9 - CT2 reset ends; CT2 starts processing.
t10 - Third service ends; VM1 remains active.
t11 - Second service ends; CT2 shuts down.

Source: The author (2020)

4.2.2 Steady State Analysis

The steady-state probabilities $\pi(i,j)$ are extracted from the solution of a linear system formed by the normalization condition (4.1) and balance equations (4.2-4.12), which are summarized in Table 8. Please consider $(i,j) \in \Omega$ in all equations to follow.

$$\sum_{i=0}^{c}\sum_{j=0}^{k}\pi(i,j) = 1 \tag{4.1}$$

The balance-equation for the empty system $(0,0)$ can be expressed in (4.2). For states where no container is active $(i = 0)$, there is at least one service $(j > 0)$ and there are enough VMs to process the admitted requests, the following will apply: (4.3) if $j < n$ and $n > 1$ or (4.4) if $j = n$. However, if the ongoing services overcome the existing VM limit and the system allows at least two containers and two services $(j > n, c > 1$ and $k > 1)$, then only (4.5) applies. Finally, other border states are represented by (4.6), i.e., no container is on $(i = 0, j > 0)$ and request limit k is reached $(j = k$ and $c > 0)$.

$$\lambda\pi_{0,0} = \mu\pi_{0,1} \tag{4.2}$$

$$(\lambda + j\mu)\pi_{0,j} = \lambda\pi_{0,j-1} + (j+1)\mu\pi_{0,j+1} \tag{4.3}$$

$$(\lambda + n\mu)\pi_{0,n} = (\lambda\pi_{0,n-1}) + (n\mu\pi_{0,n+1}) + ((n+1)\mu\pi_{1,n+1}) \tag{4.4}$$

$$(\lambda + (min(c, j-n)\alpha) + n\mu)\pi_{0,j} = (\lambda\pi_{0,j-1}) + (n\mu\pi_{0,j+1}) + (\gamma\pi_{1,j}) \tag{4.5}$$

$$((min(c, k-n)\alpha) + n\mu)\pi_{0,k} = (\lambda\pi_{0,k-1}) + (\gamma\pi_{1,k}) \tag{4.6}$$

For the states where the number of active containers varies between $0 < i < c$, (4.7) only applies if the amount of active containers and VMs is equal to the number of active services $(j = i+n, j < k$ and $c > 1)$, otherwise, if the active service number is bounded by that value and the limit k $(i+n < j < k$ and $k > n+2)$, then (4.8) is applicable. The remaining states covered by (4.9) are limited by k $(j = k$ and $c > 1)$.

$$(\lambda + (n+i)\mu + i\gamma)\pi_{i,j} = ((n+i)\mu\pi_{i,j+1}) + (\alpha\pi_{i-1,j}) + (n+i+1)\mu\pi_{i+1,j+1}) \tag{4.7}$$

$$\begin{aligned}&(\lambda + (n+i)\mu + ((min(c, j-n) - i)\alpha + i\gamma))\pi_{i,j} \\ &= \lambda\pi_{i,j-1} + ((min(c, j-n) - (i-1))\alpha\pi_{i-1,j}) + ((n+i)\mu\pi_{i,j+1}) + ((i+1)\gamma\pi_{i+1,j})\end{aligned} \tag{4.8}$$

$$
\begin{aligned}
&((n+i)\mu + ((min(c,k-n)-i)\alpha + i\gamma)\pi_{i,k} \\
&\qquad = ((min(c,k-n)-(i-n))\alpha\pi_{i-1,k}) + (\lambda\pi_{i,k-1}) + ((i+1)\gamma\pi_{i+1,k}) \quad (4.9)
\end{aligned}
$$

When all containers are active $(i = c)$, (4.10) relates only if the active service limit is reached $(j = c+n)$. However, if the active service number varies between $c+n < j < k$ and k is larger than the previous case $(c+n < k-1)$ then (4.11) applies instead. Lastly, (4.12) refers to the set of states where all containers are active $(i = c)$ and the service limit $(j = k)$ is reached.

$$(\lambda + (n+c)\mu + i\gamma)\pi_{c,c+n} = ((n+c)\mu\pi_{c,c+n+1}) + (\alpha\pi_{c-1,c+n}) \quad (4.10)$$

$$(\lambda + (n+c)\mu + i\gamma)\pi_{c,j} = \lambda\pi_{c,j-1} + ((n+c)\mu\pi_{c,j+1}) + (\alpha\pi_{c-1,j}) \quad (4.11)$$

$$((n+c) + i\gamma)\mu\pi_{c,k} = (\alpha\pi_{c-1,k}) \quad (4.12)$$

Table 8 – Balance Equations Summary

Eq.	State(s)	Condition(s)	Description
(4.2)	$(0,0)$	n/a	Empty system.
(4.3)	$(0,j)$	$(0 < j < n)$ and $(n > 1)$	All services running on VMs
(4.4)	$(0,n)$	$j > 0$	Existing VMs match service load
(4.5)	$(0,j)$	$(j > n)$, $(c > 1)$ and $(k > 1)$	Service load surpasses VM capacity; Containers are setting up
(4.6)	$(0,k)$	$(j > n)$, $(c > 1)$ and $(k > 1)$	Similar to (4.4) but with full system;
(4.7)	$(i,i+n)$	$(0 < i < c)$, $(0 < j < k)$ and $(c > 1)$	All VMs are busy and some Containers are serving;
(4.8)	(i,j)	$(0 < i < c)$, $(i+n < j < k)$ and $(k > n+2)$	Similar to (4.6) but with containers setting up
(4.9)	(i,k)	$(0 < i < c)$, $(i+n < j < k)$,$(k > n+2)$ and $(c > 1)$	Similar to (4.7) but with full system
(4.10)	$(c,c+n)$	$c > 1$	All online services are being served by all resources
(4.11)	(c,j)	$(c+n < j < k)$ and $(c+n < k-1)$	All resources are serving but there are waiting services
(4.12)	(c,k)	n/a	All resources are processing and the system is full

Source: The author (2020)

4.3 PERFORMANCE METRICS

In this section, we consider the steady-state analysis of the CTMC under study, followed by the derivation of four performance metrics: Availability (A), Reliability (R), Mean Power Consumption (C) and Mean Response Time (T).

4.3.1 Availability (A)

It is widely accepted that the adoption of the MEC-NFV environment for Core Network and Application functions closer to the UE can reduce Latency and increase Reliability. However, the likely resource limitation of edge nodes restricts their service capacity and consequently its availability, i.e., if the maximum capacity is reached, the natural options are to forward the flow to a neighbor MEC node or central cloud [Sarrigiannis et al. 2020], both of which incur on a new route built of multiple intermediate hops, introducing a high degree of uncertainty towards latency and reliability. In this respect, it becomes imperative to analyze the MEC-NFV node availability. In our model, Availability is the system's ability to offer the minimum amount of functional and accessible VNFs. In particular, a VNF instance is considered available if at least one of its constituents (VM-hosted or containerized) remains accessible. In brief, the MEC node Availability (A) (4.12) is obtained by the probability sum of all states except those representing full capacity, i.e., the system with k users.

$$A = 1 - \sum_{i=0}^{c} \pi_{i,k} \tag{4.12}$$

4.3.2 Reliability (R)

The reliability analysis of future mobile networks is of paramount importance for network operators, especially considering critical applications since it directly impacts the QoS and user experience (e.g., service response time). The designed model also evaluates the Reliability (R) being given by (4.13), which combines the admitted flow λ*A with the effective failure rate in the node, i.e., denotes the probability that a service is processed without experiencing failure.

$$R = 1 - \frac{\gamma}{\lambda * A} \sum_{i=1}^{c} \sum_{j=1}^{k} i\pi_{i,j}. \tag{4.13}$$

4.3.3 Power Consumption (C)

The computational power consumption is an important component of the operational costs and must be considered by the service provider for resource planning to address cost-performance trade-off. In this model, power consumption (C) is formed from the combination of the mean number of virtual resources and energy consumption constants (P) for each virtualization technology (VM and Container) and operating states (Idle, Setup and Busy). The power consumption (in Watts) of a single VM in idle state is denoted as: P_{idle}^{VM}, with the remaining variables being represented in the same notation (Table 9).

The mean number of VMs and containers (CT) in each state is described in (4.14)-(4.18), which are detailed in the following lines. (4.14) captures the mean amount of VMs in idle state by iterating over each system state in which no container is active $(i = 0)$ and until the total number of online services reaches the maximum amount of VMs $(j = n)$. (4.15) has three terms: the first is similar to the one in (4.14), but captures only the mean amount of busy VMs within the range of states until $(j = n - 1)$. The second iterates over the states where the load is equal or greater than the VM maximum amount and there are no containers ready yet. Lastly, the third term contains the mean amount of busy VMs on states where at least one container is processing. The same idea is applied for (4.16) and (4.18), whereas (4.17) calculates the mean number of containerized VNFs in setup by iterating over states where the number of online services is greater than the total number of active resources (VMs and Containers). Finally, power consumption (C) is given by (4.19).

Table 9 – Power Consumption Notation

Virtualization Technology	State	Status	Notation
VM-hosted	Idle	ON	P_{idle}^{VM}
VM-hosted	Busy	ON	P_{busy}^{VM}
Containerized	Idle	SLEEP	P_{idle}^{CT}
Containerized	Setup	ON	P_{setup}^{CT}
Containerized	Busy	ON	P_{busy}^{CT}

Source: The author (2020)

$$\overline{VM}_{idle} = \sum_{j=0}^{n}(n-j)\pi_{0,j} \tag{4.14}$$

$$\overline{VM}_{busy} = \sum_{j=0}^{n-1} j\pi_{0,j} + \sum_{j=n}^{k} j\pi_{0,j} + \sum_{i=1}^{c} \sum_{j=n+i}^{k} j\pi_{i,j} \tag{4.15}$$

$$\begin{aligned}\overline{CT}_{idle} = \sum_{j=0}^{n} c\pi_{0,j} + \sum_{j=n+1}^{n+c} (n+c-j)\pi_{0,j} \\ + \sum_{i=1}^{c} \sum_{j=n+i}^{n+c} (n+c-j)\pi_{i,j}\end{aligned} \tag{4.16}$$

$$\overline{CT}_{setup} = \sum_{i=0}^{c} \sum_{j=n+i}^{k} min(j-n, c)\pi_{i,j} \tag{4.17}$$

$$\overline{CT}_{busy} = \sum_{i=1}^{c} \sum_{j=n+i}^{k} i\pi_{i,j} \tag{4.18}$$

$$\begin{aligned}C = P_{idle}^{VM}\overline{VM}_{idle} + P_{busy}^{VM}\ \overline{VM}_{busy} \\ + P_{idle}^{CT}\ \overline{CT}_{idle} + P_{setup}^{CT}\overline{CT}_{setup} + P_{busy}^{CT}\ \overline{CT}_{busy}\end{aligned} \tag{4.19}$$

4.3.4 Response Time (T)

Since part of the future mobile applications have strict communication latency requirements, analyzing the response time also becomes crucial for MEC node resource-related issues such as dimensioning. We define the Response Time of a VNF that processes the service T as the interval between the service arrival (on the edge node) and its departure (service conclusion), which includes the containerized VNF setup restart times if these events are triggered. The Response Time (4.21) is obtained by calculating the mean number of online services (4.20) and dividing by the mean number of accepted services.

$$\overline{U} = \sum_{j=0}^{k} j\pi_{0,j} + \sum_{i=1}^{c} \sum_{j=n+i}^{k} j\pi_{i,j} \tag{4.20}$$

$$T = \frac{\overline{U}}{\lambda A} \tag{4.21}$$

4.4 MODEL VALIDATION

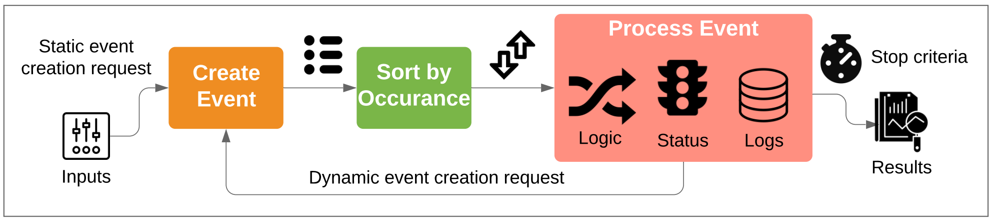
A simulation model was adopted to validate the analytical model. In this simulator, the interactions between requests and attendance are implemented so that the same performance metrics can be obtained and compared to those from the analytical model.

4.4.1 The Discrete-event Simulator

A discrete-event simulator (Fig. 7) emulates the system's operation such as the behavior of a multipurpose network based on a discrete sequence of events in time. Each event occurs at a particular instant and may cause a state change [Tako and Robinson 2009]. As opposed to real-time simulations, no state transitions are assumed to occur between consecutive events and the simulation jumps to the next event. The simulation must also keep track of the current time; in whatever measurement units are suitable for the system being modeled.

In a simulation model, the performance metrics are not analytically derived from probability distributions, but rather determined as averages from different runs. For this reason, it is important to dimension the total number of runs that is necessary to guarantee reasonable statistical significance. In addition, the simulation designer must decide the ending condition for a "run". Common examples are: at a specific time instant or after processing a pre-defined number of events. We have opted for discrete-event simulators for validating our models mainly due to simplicity and flexibility. In one hand, these are quite straightforward and accurate but on another, it takes some effort to code when compared to off-the-shelf solutions (e.g., ns2, ns3). We have built our simulators using MATLAB v2016 but other researchers also distribute their code within two or more languages (e.g., C) for performance gains.

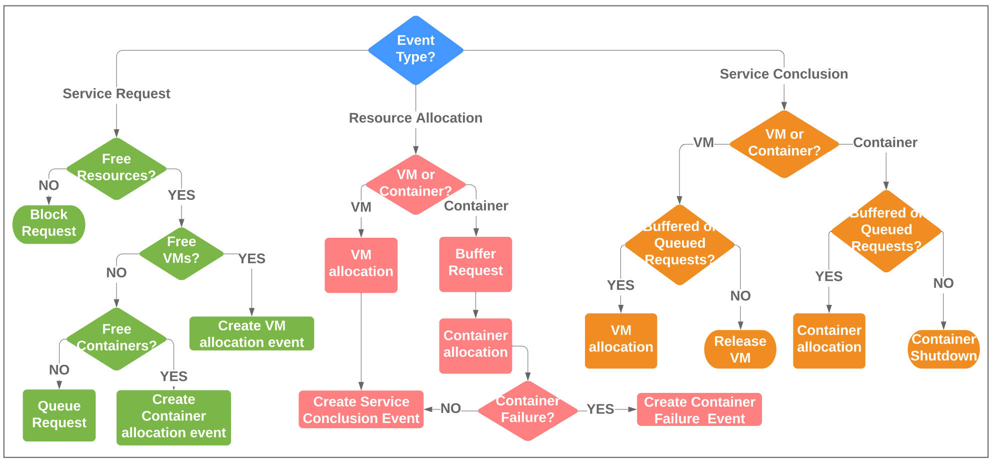
Figure 7 – Simulation Flow



Source: The author (2020)

The simulator is formed by three blocks: (1) event creation, (2) event sorting and (3) event processing, and two additional steps: input setting (e.g., request and service rates) and stop criterion (total simulation time). The first block creates events that can be either static, i.e., created before the simulation starts (e.g. request arrivals) or dynamic, i.e., created during simulation (e.g. service conclusions). A list of events is then sent to the next block, with each event being associated to an unique time instant, i.e., the time that the event will be processed during the simulation. The second block essentially sorts the event list in ascending order (time occurrence), providing an ordered list to the next phase. Lastly, the third block processes each event individually, deciding which actions to take by consulting the system status, hence creating dynamic events if necessary, for instance. Moreover, it also updates the state variables and registers all relevant data in an event log, which is the source for further producing the performance metrics.

Figure 8 – Part of the simulator logic



Source: The author (2020)

The simulation model allows greater control since we are able to track which resource is being used by each request, which is not possible in an analytical model. Hence, we keep a resource table that maps the Resource ID to the Active request ID. The logic subdivision is quite long for being fully represented in this document, but part of the simulator was depicted in Fig. 8, which encompasses three main event types "Service Request", "Resource Allocation" and "Service Conclusion". Please note that each of these events may trigger multiple actions depending on the system state. For instance, a "service conclusion" that was running on a Container may trigger a Container Allocation event if there are buffered services. Finally, the consequences of each event are recorded using the log component, after each round (see Table 10); Once the stop criteria is met, the simulation is considered to be finished and therefore the performance metrics can be computed.

Table 10 – Log description

Status	Description
№ of Accepted Request	№ of accepted requests for VMs or Containers
№ of Blocked Requests	№ of blocked requests due to resource occupation
№ of VM allocations	№ of requests allocated to a VM
№ of Container allocations	№ of requests allocated to a Container
№ Containers in Setup	№ of Containers in setup process
№ of Container Failures	№ of Containers that failed during service
№ of Buffered Requests	№ of buffered requests due to resource occupation

Source: The author (2020)

4.4.2 Validation Results

The analytical results were validated against extensive discrete-event simulations (Figs. 9-20), where the lines denote the analytical and the markers represent simulation results. With regards to the main parameters, we have followed a subset of the 3GPP Release 16 (TR 38.824) [3GPP 2020], in which the service time is 1ms (1 service/ms) while service arrivals range from 1 up to 100 requests/ms. In addition, unless otherwise stated, the baseline values for failure (γ) and setup rates (α) were 0.001 and 1 unit/ms, respectively, which is in accordance with [Kaur et al. 2017]. In terms of energy power consumption for VMs and containers in different operation states, we adopted the individual consumption values described in [Morabito 2015], and summarized in Table 11 with other general parameters. The subsections 4.4.2.1-4.4.2.3 encompass three evaluation scenarios were we show the flexibility of the proposed model in terms of multiple edge sizes (4.4.2.1), the impact of various VM-hosted/containerized VNF arrangements (4.4.2.2) and improved setup/failure rates (4.4.2.3). In all experiments, it is assumed that all calculations related to the steady-state analysis and performance metrics are performed in a central cloud or neighbor MEC node.

4.4.2.1 Multiple edge sizes (k)

This scenario shows the impacts of λ on the different MEC-enabled node sizes ($k = 25, 50, 100$) but running the same VNF scaling process with equal VM or Containerized (CT) VNF ratios ($VMs = 40\%, CTs = 60\%$). Generally, one can see that in Figs. 9-10 the best values for Availability and Reliability are found for low workloads, respectively. However, as λ increases, the Availability tends to always decrease and it is not bounded by a particular value.

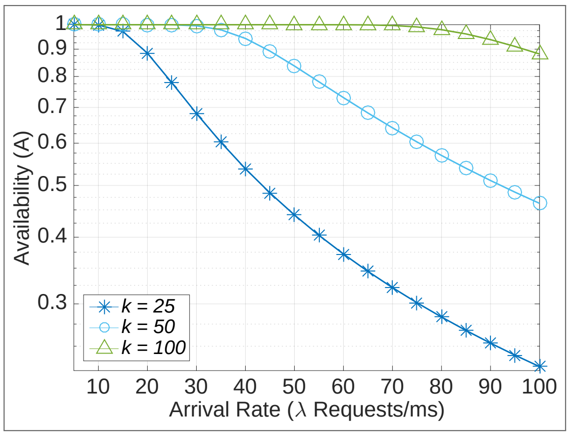
Table 11 – Simulation Parameters

Parameter	Value
Arrival rate (λ)	$[1100]$ requests/ms
Service rate (μ)	1 service/ms
Failure rate (γ)	0.001 unit/ms
Setup rate (α)	1 unit/ms
Idle VM Energy Consumption (P^{VM}_{idle})	20 W
Busy VM Energy Consumption (P^{VM}_{busy})	25 W
Idle CT Energy Consumption (P^{CT}_{idle})	4 W
Setup CT Energy Consumption (P^{CT}_{setup})	8 W
Busy CT Energy Consumption (P^{CT}_{busy})	23 W

Source: The author (2020)

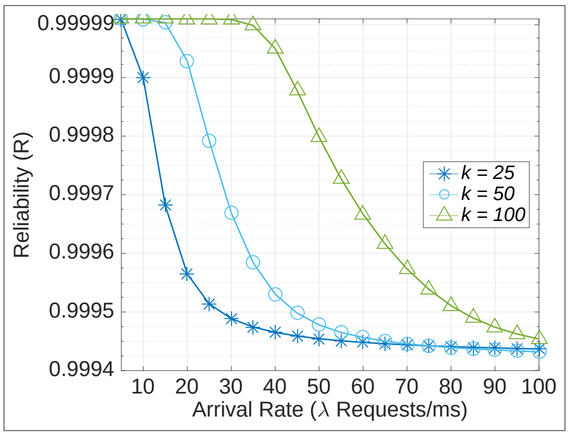
In addition, although the same VM/Containerized VNF ratios were used, the number of VM-hosted VNFs in each curve are different ($n = 15$, 20 and 40), which explains the discrepancy towards the results for this metric.

Figure 9 – Availability



Source: The author (2020)

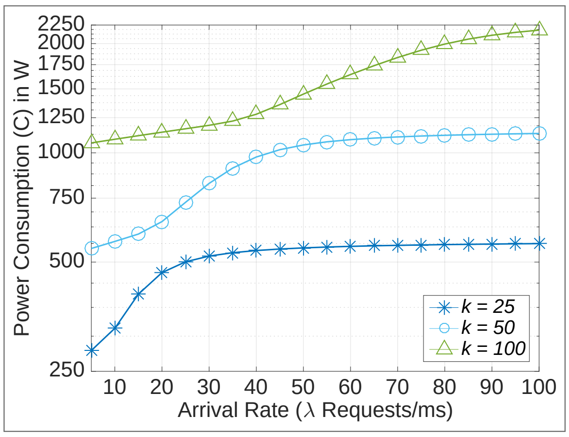
Figure 10 – Reliability



Source: The author (2020)

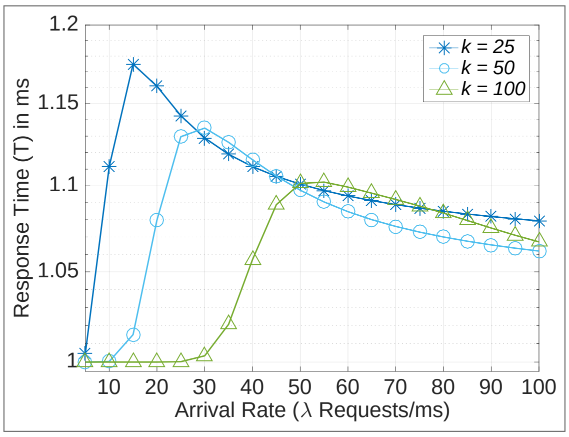
Contrarily, the Reliability is bounded by the number of accepted services, and since the same VM/Containerized VNF ratios were used, the curves are expected to overlap for higher λ values. The reasons are as follows. When $\lambda << n\mu$, the incoming services are usually handled by VMs, i.e., the containerized VNFs are hardly required. Moreover, they are turned on as λ approaches $n\mu$, negatively impacting the Reliability due to the increasing failures brought by the containerized VNFs. Finally, when λ approaches $(n + c)\mu$, most containerized VNFs are turned on, indicating a saturated infrastructure. The same reasoning applies for the Power Consumption in Fig. 11. However, the curves do not overlap for higher λ since each configuration has a different absolute number of resources.

Figure 11 – Power Consumption



Source: The author (2020)

Figure 12 – Response Time



Source: The author (2020)

The impact of the increasing load on the mean response time (Fig. 12) illustrates the only metric in which the trend of the curves experiences both an ascent and descent phase. In the first phase, the response time grows sharply due to the containerized VNF setup delay. Specifically, smaller response delays will happen invariably when $\lambda << n\mu$ because most services will be handled by VM-hosted VNFs. As λ approaches $n\mu$ and becomes larger, containerized VNFs are turned on. In this phase, however, the response time still increases due to the setup delay. The third phase encompasses the moment when the mean response time begins to decrease in $\lambda = 20,\ 32.5$, and 60 respectively for $k = 25, 50, 100$. Such behavior occurs since the containerized VNFs tend to become readily available for new service arrivals, not needing to wait for the setup delay. Lastly, a saturation phase takes place when $\lambda >> (n + c)\mu$, i.e., the system is unable to handle the load, which is only reflected in the Availability (Fig. 9).

In order to keep the figures within the same range of λ, Fig. 12 is limited to $\lambda = 100$ which might be confusing since the best apparent result for this metric comes from the intermediate configuration ($k = 50$). Indeed, this configuration shows the best result for large part of the experiment, but for $\lambda > 100$ the configuration $k = 100$ is likely to retake the advantage, similar to the comparison between $k = 25$ and $k = 50$. The fact is that at some point, the impact of the setup phase of the containerized VNFs is mitigated. This result reinforces the importance of the holistic view promoted by the evaluation of at least these four metrics concomitantly, since for a given scenario, the intermediate ($k = 50$) or even the configuration with least resources ($k = 25$) could eventually be the most cost-beneficial to handle the load.

The designed model allows the evaluation of multiple MEC node sizes by tuning the appropriate parameters accordingly, however, the next evaluations (subsections 4.4.2.2 and 4.4.2.3) focus on a single-sized MEC node ($k = 25$) positioned together within RAN equipment, i.e., similar to the Near Scope in Fig. 3. In addition, the following results display dashed lines for the Reliability and Response Time, which represent three URLLC industry vertical thresholds: Augmented Reality (AR), Smart Manufacturing (SM) and, Transport Industry (TI) that are summarized in Table 12. For the sake of simplicity, we have adopted specific values rather than the ranges described in the original document.

Table 12 – Reliability and Response Time Thresholds for URLLC

Industry Vertical	Reliability	Response Time
Augmented Reality (AR)	99.9%	1.1 ms
Smart Manufacturing (SM)	99.99%	1.15 ms
Transport Industry (TI)	99.999%	1.2 ms

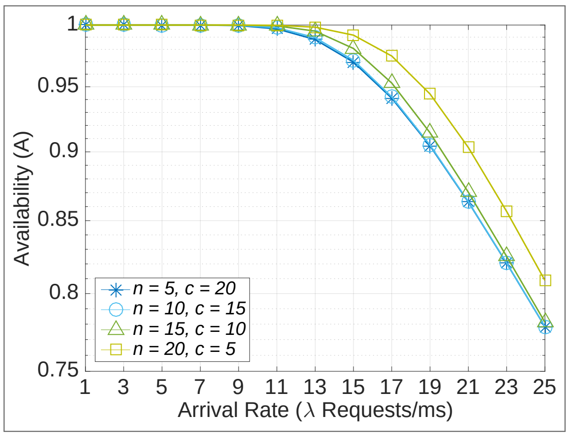
Source: [Kherraf et al. 2019]

4.4.2.2 Multiple VM/Container Arrangements (n, c)

Figs. 13-16 illustrate the impact of varying VM/Container ratios that are limited to the same amount of resources (k). The results evince that configurations with a smaller number of VMs compared to Containers tend to have lower Availability, faster Response Times and higher Reliability, which is expected since the VM-hosted VNFs are stable compared to containerized VNFs, i.e., not prone to failures. For these metrics, we expected similar results between curves at least until $\lambda = 5$, since this is the load in which all analyzed configurations have enough VMs $n = (5, 10, 15, 20)$. However, for the Availability (Fig. 13) the curves start to differentiate only at $\lambda = 11$ whereas the Reliability and Response Times responded from $\lambda = 3$.

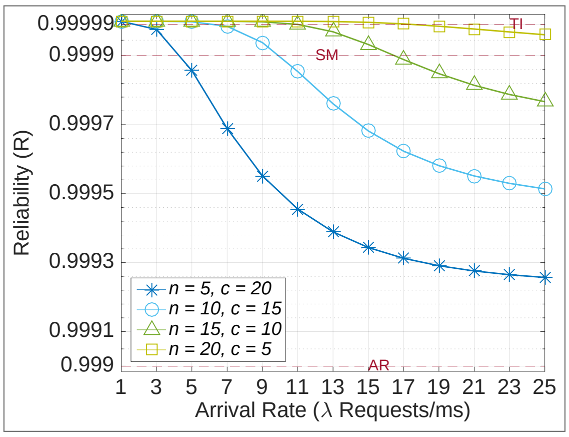
With regards to Reliability in Fig. 14, not even the arrangement with most VMs ($n = 20, c = 5$) was able to keep itself above all dashed lines, breaking the TI threshold at $\lambda = 15$, i.e., with a significant distance between its theoretical VM capacity at $\lambda = 20$. The fact is that $\lambda \leq c$ does not guarantee that all requests will be processed by VMs. Considering URLLC applications, if even relatively few requests are experiencing setup delays, one or multiple failures can cause serious capacity issues, leading to lower Reliability as more containerized VNFs are needed. On the other hand, we expected that a platform formed only by VMs would become costly in terms of power consumption since VMs are not scaled fast enough for URLLC applications, and therefore must be continuously active, regardless of being idle or busy.

Figure 13 – Availability



Source: The author (2020)

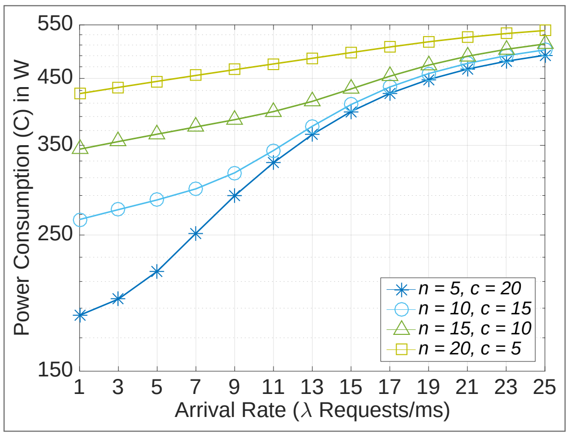
Figure 14 – Reliability



Source: The author (2020)

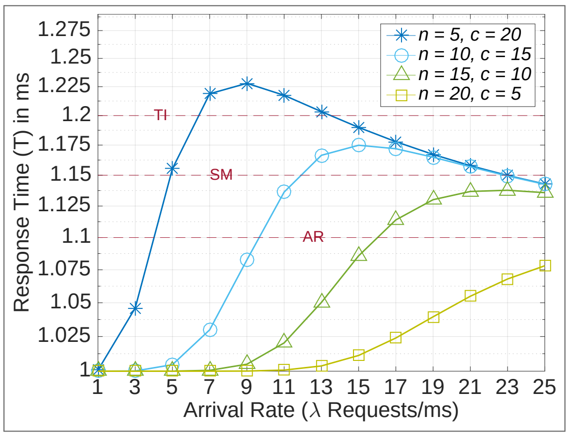
In Fig. 15 we observed that although there is a large difference in terms of power consumption until $\lambda = 7$, from this point on, the curves rapidly converge despite the scaling feature and lower consumption of containerized VNFs, possibly because the absolute difference between the curves is largely correlated with the difference between the adopted constants for the consumption of VMs and Containers in Busy state, which is only of 2W. Thus the scaling strategy is not as effective for higher loads as it is in lower loads.

Figure 15 – Power Consumption



Source: The author (2020)

Figure 16 – Response Time



Source: The author (2020)

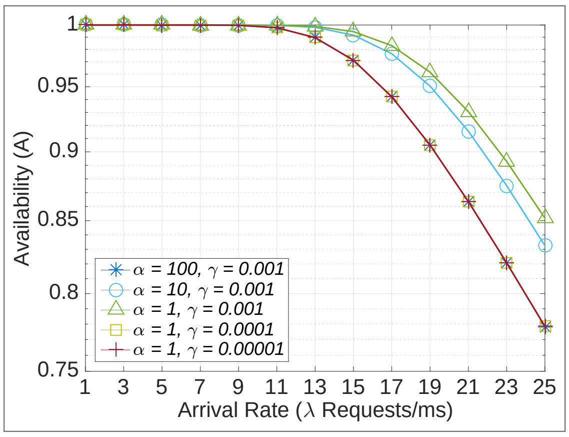
A possible solution is to adjust the VM/container ratio according to the demand, i.e., an operator can enable arrangements with more containers for low demands or with more VMs for higher loads, similarly to the approach used in [Emara et al. 2021]. For instance, considering only the Reliability and the Smart Manufacturing applications (Fig. 14), the configuration with 5 VMs and 20 containers could be used if $\lambda < 3$, whereas if $3 < \lambda < 15$ a more balanced set with 15 VMs and 10 containers could take place and finally, the arrangement with 20

VMs and 5 containers would only become available if $\lambda > 15$. Please note that this example would not be applicable considering the Mean Response Time (Fig. 16), i.e., the intervals for swapping between arrangements would necessarily differ. In [Emara et al. 2021], although multiple indicators were suggested, the authors proposed a resource optimization solution based only in one argument. On the other hand, besides considering the virtual host pitfalls, our model allows using multiple performance metrics and URLLC application thresholds to formulate specific and practical solutions.

4.4.2.3 Multiple Setup and Failure Rates (α, γ)

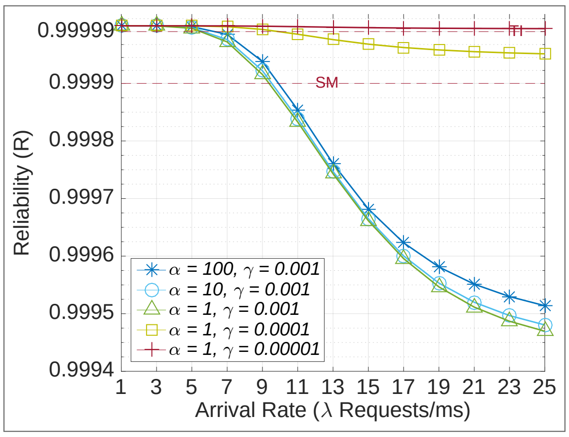
In this evaluation, a single configuration with $k = 25$ was adopted, but with a fixed arrangement of 10 VMs and 15 containers and varying setup (α) and failure (γ) rates. This configuration was analyzed in section 4.4.2.1 with fixed $\alpha = 1$ and $\gamma = 0.001$ and becomes interesting for the current experiment since it is balanced in terms of both resource types and yet it is prone also to be impacted by α and γ.

Figure 17 – Availability



Source: The author (2020)

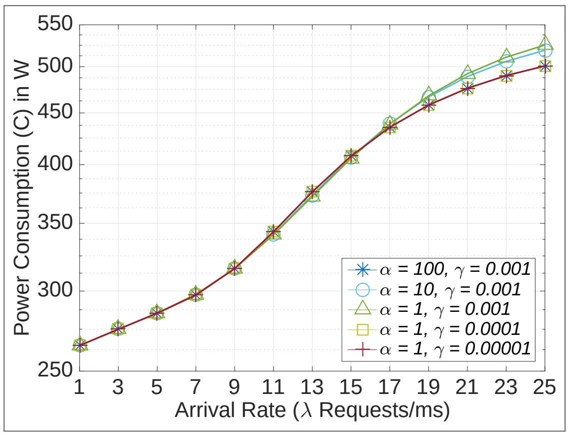
Figure 18 – Reliability



Source: The author (2020)

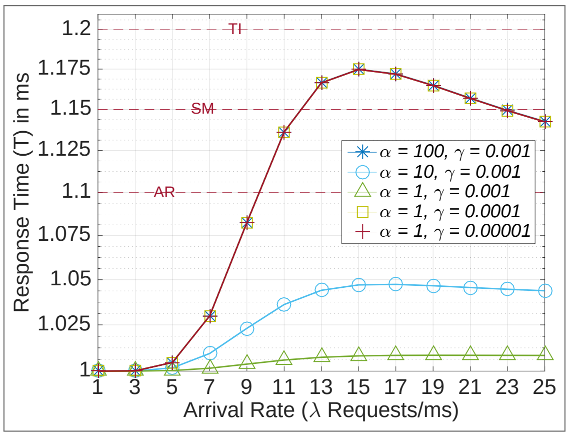
A larger α means smaller container setup delays, i.e., more VNF instances become available per unit time. As expected, higher α rates resulted in a blocking probability reduction (Fig. 17), but interestingly, Fig. 18 reveals the opposite: higher α rates actually increased the system's failure probability. This unexpected behavior indicates that isolated improvements in the setup rate may help the admission process but become a burden to the admitted flows. In other words, a higher setup rate increases the flow served by the failure-prone component (containerized VNFs) per unit time, therefore also increasing the chances of failures, which suggests that future enhancements in this parameter might be insufficient.

Figure 19 – Power Consumption



Source: The author (2020)

Figure 20 – Response Time



Source: The author (2020)

In Fig. 19, the Power Consumption is expected to differ since α variations necessarily impact the amount of powered on container per unit time. However, this experiment has shown that the baseline configuration ($\alpha = 1$) presented a similar power consumption pattern compared to those with higher α values throughout the entire evaluation. A performance difference is also observed in Fig. 20, where the Mean Response Times from the baseline curve are significantly greater than the curves with enhanced α rates. In $\lambda = 15$ the difference between baseline and the curve with $\alpha = 100$ reaches a maximum of 0.150 ms.

A larger γ means smaller intervals between successive containerized VNF failures. In contrast to α, this parameter improves as it gets smaller. Thus, we have enhanced γ by dividing its default value by 10 e and 100 times. However, if, on the one hand, our expectations were met regarding the overall Reliability (Fig. 18), i.e., lower failure rates allowed the two curves to surpass the Smart Manufacturing Reliability threshold, to our surprise, the Availability (Fig. 17), Power Consumption (Fig. 19) and Response Time (Fig. 20) remained almost unchanged, despite the large difference between the adopted γ values. These findings evince what level of improvement containerized VNFs must achieve to meet specific requirements, recalling that both software and hardware share relevance towards this aspect, as investigated in [Lal et al. 2017]. In brief, it becomes clear that containerized VNF setup delays critically impact the admission (Availability), whereas the Reliability reacts severely to failures.

4.5 CHAPTER SUMMARY

In this Chapter we described the elements that compose a single MEC node environment, giving special attention to the virtualized components and their assumptions. Once the environment was defined, a model was designed to denote the interactions between VM-hosted and containerized VNFs. For this, a queuing system with priority and repair times was considered and four performance metrics were derived, based on the related works. To validate the analytical model, an approach based on simulation was used.

Table 13 – Chapter 4 Summary

Section	**Goal(s)**	**Output(s)**
4.1 SYSTEM MODEL	Describe the top-level elements of the system to be modelled.	Virtual host assumptions: setup/recovery delays, failures and dynamic containerized VNF scaling.
4.2 ANALYTICAL MODEL	Describe the CTMC-based model of a single MEC node for URLLC with VMs and Containers scaled upon demand.	a) Steady State Analysis (transitions and state space) illustrated in the two-dimension CTMC diagram in Fig.5; b) Normalization condition (4.1) and balance equations (4.2-4.12) summarized in Table 8.
4.3 PERFORMANCE METRICS	a) Explain the importance of each adopted metric: Availability (A), Reliability (R), Power Consumption (C) and Response Time (T); b) Detail each metric formulation.	Expressions: MEC node Availability (A) (4.12), Reliability (R) (4.13), Power Consumption (C) (4.19). and Response Time (T) (4.21).
4.4 MODEL VALIDATION	a) Describe the adopted discrete-event simulator; b) Validate results in the following scenarios: 1) Multiple edge sizes (k), 2) Multiple VM/Container Arrangements (n,c) and 3) Multiple Setup and Failure Rates (α, γ).	a) First scenario: The trend of the curves for the Response Time (Fig. 12) has both an ascent and descent phase. b) Second Scenario: The scaling strategy is not as effective for higher loads as it is in lower loads. c) Third scenario: the Availability (Fig. 17), Power Consumption (Fig. 19) and Response Time (Fig. 20) remained almost unchanged, despite the large difference between the adopted γ values.

Source: The author (2020)

5 MODEL EXTENSION

This chapter describes an extension of the model designed in Chapter 4. Compared to the previous model, this part differs in terms of two main characteristics: (a) separate service rates for VMs and containers and (b) service migration.

In Chapter 4, the MEC node is equipped with a single Physical Machine (PM) that encompasses multiple VMs and containers. In that case, the VMs were considered not to be prone to failures, contrarily to the containers. However, it is widely known that VMs interfere with one another due to the limited physical resources [Mavridis and Pantelis 2019]. To account for resource sharing among the VMs (e.g., CPU, I/O, and memory), which leads to a degraded performance, input/output (I/O) interference is observed in the extended model. Hence, while the containers remained failure-prone, which indirectly impacts the overall service rate, a higher number of VMs can also negatively impact the system's service rate. Moreover, service migration between containers and VMs occurred in two ways: (1) due to container failure and (2) due to a VM becoming idle. In the extended model, the second case is not considered anymore since this move could incur extra overhead without necessarily having occurred a failure event.

This chapter is structured similarly to Chapter 4, with Section 5.1 describing the most relevant events, together with the model assumptions. Section 5.2 details the analytical formulation/modeling for the MEC node. Following this, Section 5.3 outlines the formulation for each adopted metric and finally, the validation based on simulation of the proposed model is outlined in Section 5.4.

5.1 SYSTEM MODEL

This section describes the proposed analytical scaling model, considering a single isolated MEC node, where requests originated from UEs are processed by VNFs, which can be scaled up (down) to cope with intensive (mild) request periods. Similar to Chapter 4, each VNF runs equally and independently on a single microkernel–based VM [Fautrel et al. 2019] or container sharing a common Physical Machine (PM), with VMs executing uninterruptedly while containers can be scaled upon demand. A centralized control unit determines request admissions, only activating containerized VNFs when all VM-hosted are busy. The containerized

VNF activation comprises the same two phases: initializing the kernel image and launching the specified function, which is interpreted as a single transition interval (setup time).

In this model, however, a VM processing overhead is taken into account, i.e., how parallel-operating VMs influence on each other, leading to degraded computational performance [Mavridis and Pantelis 2019]. Various authors have considered that multiple VMs sharing the same PM can incur in a performance reduction, mainly due to I/O interference between VMs [Bruneo 2014], [Emara et al. 2021]. We define the degradation factor d (≥ 0) as the percentage increase in the expected service time experienced by a VM when multiplexed with another VM. We also assume that, in order to obtain a fair distribution of VMs, the system is able to optimally balance the load among the PMs with respect to the resources required by VMs, thus reaching a homogeneous degradation factor.

For the case of a single VM deployment, the task's execution rate is μ services/unit time; but, to account for the VM overhead, the task's execution rate (5.1) takes the total number of VMs, where d is the computation degradation factor [Emara et al. 2021]. Hence, μ_v is a monotone decreasing function of d. On the other hand, the container's average performance is generally better than the VM's and is even comparable to that of the PM with regards to multiple features [Mavridis and Pantelis 2019]. Thus, the container performance overhead was considered to be negligible.

$$\mu_v = \frac{\mu}{(1+d)^{(n-1)}} \tag{5.1}$$

Another key difference in this model compared to the one in Chapter 4 is the fact that services that are processed by a container shall remain in that specific instance until finishes being processed or until a failure event, regardless of any VM being idle. As in Chapter 4, only active containerized VNFs may suffer failures during attendance, which implies either a service migration to an available VM/container or a repair (triggering a new setup period), with progress being lost only in the latter case. In addition, as soon as an operative VNF finishes processing and there are no remaining requests, the VNF instance can either be powered down together with the host container or remain active if hosted by a VM. The shutdown delay is ignored for being significantly smaller than the setup (repair) magnitudes [Kaur et al. 2017].

5.2 ANALYTICAL MODEL

The current system also comprises a single MEC node, with a maximum capacity of K requests that are served by up to n VMs and c containers, with $K \geq n + c$, which implies a queue (q) that is limited to $K - (n + c)$ requests. Service requests also follow a Poisson process with rate λ (requests/ms) and server capacities of one service with an exponentially distributed service rate of μ_c for containerized VNFs, whereas for VMs, μ_v is given by (5.1). In other words, the current model allows different service rates for each resource type.

Container setup/repair times and failures are also exponentially distributed with rates α and γ, respectively. A regular first come first served queue was assumed for new requests with prioritization for retrial. We assume a standalone deployment and the system is modeled as an M/M/n+c/K queue (Fig. 21) with setup time and failure. The feasible state space is given by $\Omega = (i, j, k) \mid 0 \leq i \leq n,\ 0 \leq j \leq c$, and $0 \leq k \leq K$, with $i + j \leq k$ and $i, j, n, c, k, K \in \mathbb{Z}^+$. Each state ($i$,$j$,$k$) denotes the number of services allocated to VMs (i), containers (j) and the total number of services in the system (k), respectively. Furthermore, the steady-state probabilities $\pi(i, j, k)$ are extracted from the solution of a linear system formed by the normalization condition (5.2) and balance equations (5.3)-(5.16). Please consider $(i, j, k) \in \Omega$ in all equations to follow.

$$\sum_{i=0}^{n}\sum_{j=0}^{c}\sum_{k=i+j}^{K} \pi(i, j, k) = 1 \tag{5.2}$$

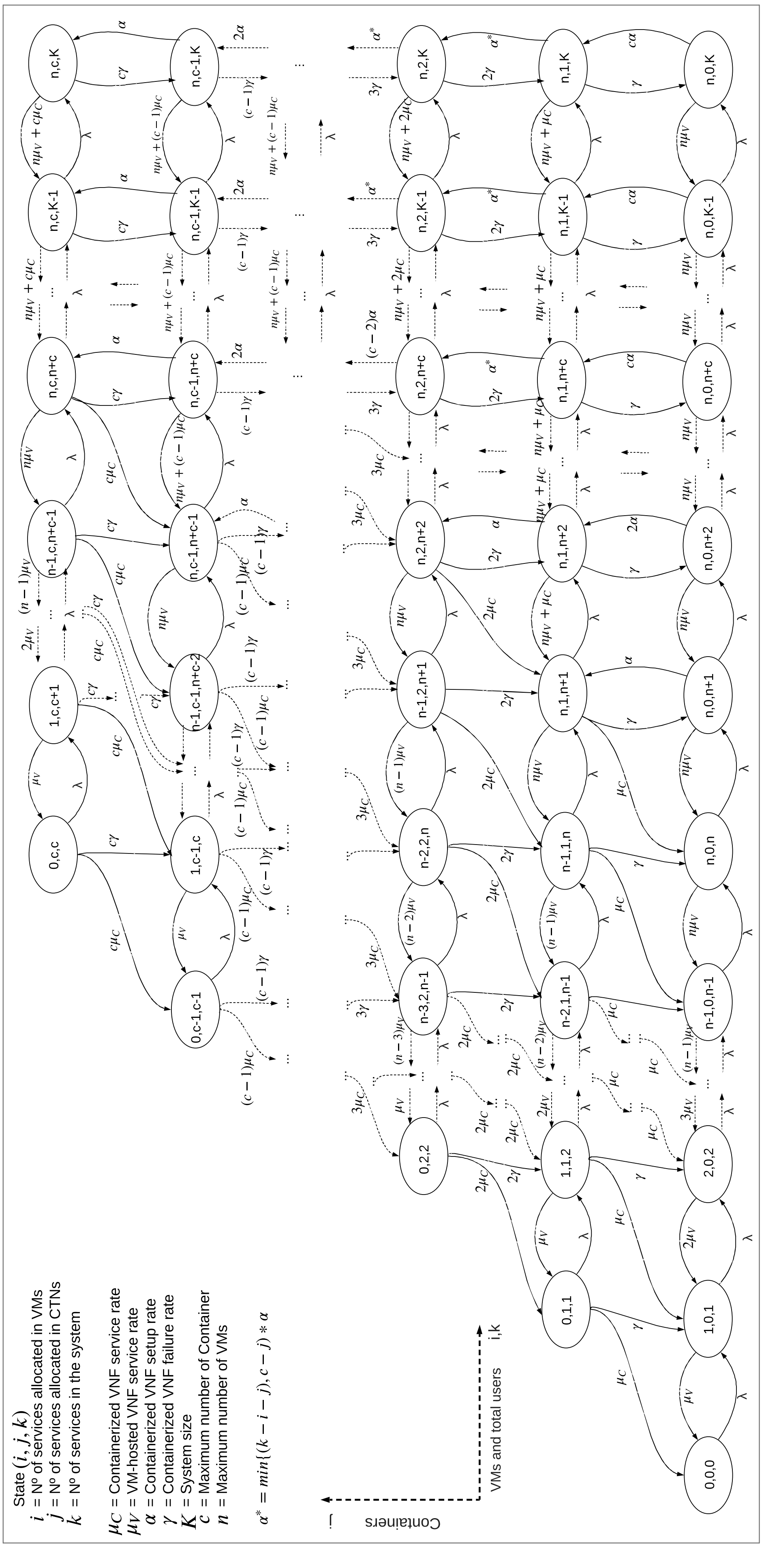
The single state where there are no services is represented by (5.3). (5.4) covers the states where all services are running on VM-hosted VNFs, i.e., no containers are required. Next, (5.5) and (5.6) are similar to (5.4), however, in the first there are services in the buffer for containers to setup while in the latter there are no containers in the system to be turned on.

$$\lambda\pi_{(0,0,0)} = \mu_c\pi_{(0,1,1)} + \mu_v\pi_{(1,0,1)} \tag{5.3}$$

$$\begin{aligned}(\lambda + i\mu_v)\pi_{(i,0,i)} \\ = \lambda\pi_{(i-1,0,i-1)} + min(i+1, n)\mu_v\pi_{(min(i+1,n),0,i+1)} + \gamma\pi_{(i-1,1,i)} + \mu_c\pi_{(i,1,i+1)}\end{aligned} \tag{5.4}$$

$$[(\lambda + n\mu_v) + min(k-n, c)\alpha]\pi_{(n,0,k)} = \lambda\pi_{(n,0,k-1)} + n\mu_v\pi_{(n,0,k+1)} + \gamma\pi_{(n,1,k)} \tag{5.5}$$

Figure 21 – Space state diagram


State (i, j, k)
i = Nº of services allocated in VMs
j = Nº of services allocated in CTNs
k = Nº of services in the system
μ_C = Containerized VNF service rate
μ_V = VM-hosted VNF service rate
α = Containerized VNF setup rate
γ = Containerized VNF failure rate
K = System size
c = Maximum number of Container
n = Maximum number of VMs
$\alpha^* = min\{(k - i - j), c - j\} * \alpha$
Containers j
VMs and total users i,k

Source: The author (2021)

$$[n\mu_v + min(K-n,c)\alpha]\pi_{(n,0,K)} = \lambda\pi_{(n,0,K-1)} + \gamma\pi_{(n,1,K)} \tag{5.6}$$

The leftmost diagonal states of Fig. 21 are described by (5.7), with no busy VM and some containers running services, while (5.8) refers to the system states with all VM-hosted VNFs idle and all the containers busy. The top-left states in Fig. 21 are represented by 5.9, where all containers are occupied but some VMs are idle. In (5.10), all VMs and CTs are busy and there are no services in the buffer. Given by (5.11), the top-right states (Fig. 21) are those where all VMs and CTs are Busy and some services waiting in the buffer. Similar to the previous set, 5.12 refers to the top-rightmost part of Fig. 21, where all VM-hosted and containerized VNFs are busy and the system is full.

$$[(\mu_c+\gamma)j+\lambda]\pi_{(0,j,j)} = \mu_v\pi_{(1,j,j+1)} + (j+1)\mu_c\pi_{(0,j+1,j+1)} \tag{5.7}$$

$$[(\mu_c+\gamma)c+\lambda]\pi_{(0,c,c)} = \mu_v\pi_{(1,c,c+1)} \tag{5.8}$$

$$[(\mu_c+\gamma)c+i\mu_v+\lambda]\pi_{(i,c,c+i)} = \lambda\pi_{(i-1,c,c+i-1)} + (i+1)\mu_v\pi_{(i+1,c,c+i+1)} \tag{5.9}$$

$$[(\mu_c+\gamma)c+n\mu_v+\lambda]\pi_{(n,c,n+c)} = \lambda\pi_{(n-1,c,n+c-1)} + (n\mu_v+c\mu_c)\pi_{(n,c,n+c+1)} + \alpha\pi_{(n,c-1,n+c)} \tag{5.10}$$

$$[(\mu_c+\gamma)c+n\mu_v+\lambda]\pi_{(n,c,k)} = \lambda\pi_{(n,c,k-1)} + (n\mu_v+c\mu_c)\pi_{(n,c,k+1)} + \alpha\pi_{(n,c-1,k)} \tag{5.11}$$

$$[(\mu_c+\gamma)c+n\mu_v]\pi_{(n,c,K)} = \lambda\pi_{(n,c,K-1)} + \alpha\pi_{(n,c-1,K)} \tag{5.12}$$

The left-most states in Fig. 21 are covered by (5.13), where all VMs are running services and the system is full, but some containers are still scaling up. The set of states where the number of services is larger than that of both active VMs and Containers, i.e., the buffer is holding some services is described by (5.14). Furthermore, at the frontier, (5.15) refers to the states where all VMs and part of the containers are serving, but no services are buffered. Lastly, the states where both VMs and containers are partially occupied and the buffer is empty is located at the left intermediate part of Fig. 21, and its respective equation is (5.16). The complete equation set is summarized in Table 14.

$$\begin{aligned}&[(\mu_c+\gamma)j+n\mu_v+min(K-n-j,c-j)\alpha]\pi_{(n,j,K)}\\&\quad=\lambda\pi_{(n,j,K-1)}+(j+1)\gamma\pi_{(n,j+1,K)}+min(K-n-j-1,c-j-1)\alpha\pi_{(n,j-1,K)}\end{aligned}\tag{5.13}$$

$$\begin{aligned}&[(\mu_c+\gamma)j+n\mu_v+\lambda+min(k-n-j,c-j)\alpha]\pi_{(n,j,k)}\\&\quad=\lambda\pi_{(n,j,k-1)}+(j+1)\gamma\pi_{(n,j+1,k)}+min(k-n-j-1,c-j-1)\alpha\pi_{(n,j-1,k)}\end{aligned}\tag{5.14}$$

$$\begin{aligned}&[(\mu_c+\gamma)j+n\mu_v+\lambda]\pi_{(n,j,k)}\\&\quad=\lambda\pi_{(n-1,j,k-1)}+(j+1)\gamma\pi_{(n-1,j+1,k)}+(j+1)\mu_c\pi_{(n,j+1,k+1)}\\&\qquad+(n\mu_v+j\mu_c)\pi_{(n,j,k+1)}+\alpha\pi_{(n,j-1,k)}\end{aligned}\tag{5.15}$$

$$\begin{aligned}&[(\mu_c+\gamma)j+i\mu_v+\lambda]\pi_{(i,j,k)}\\&\quad=\lambda\pi_{(i-1,j,k-1)}+(i+1)\mu_v\pi_{(i+1,j,k+1)}+(j+1)\gamma\pi_{(i-1,j+1,k)}+(j+1)\mu_c\pi_{(i,j+1,k+1)}\end{aligned}\tag{5.16}$$

Table 14 – Balance Equations Summary

Eq.	State(s)	Condition(s)	Description
(5.3)	$(0,0,0)$	n/a	Empty system.
(5.4)	$(i,0,i)$	$(0 \leq i \leq n)$	All services are running on VMs.
(5.5)	$(n,0,k)$	$(n+1 \leq k \leq K-1)$	All VMs are busy and no container is yet active.
(5.6)	$(n,0,K)$	(n/a)	Similar to (5.4), but the system is full.
(5.7)	$(0,j,j)$	$(1 \leq j \leq c-1)$	No busy VM and some containers running services.
(5.8)	$(0,c,c)$	(n/a)	Containers running all services.
(5.9)	$(i,c,c+i)$	$(1 \leq i \leq n-1)$	VMs partially busy and all containers running services.
(5.10)	$(n,c,n+c)$	(n/a)	All containers and VMs are serving; queue is empty.
(5.11)	(n,c,k)	$(n+c+1 \leq k \leq K-1)$	Similar to (5.9), but with waiting services.
(5.12)	(n,c,K)	(n/a)	All VMs and containers are serving and the system is full.
(5.13)	(n,j,K)	$(1 \leq j \leq c-1)$	Full system with busy VMs and containers either serving or scaling up.
(5.14)	(n,j,k)	$(1 \leq j \leq c)$ and $(n+j+1 \leq k \leq K-1)$	All VMs and containers are serving and the queue is not empty.
(5.15)	$(n,j,n+j)$	$(1 \leq j \leq c)$	All VMs and part of the containers are serving.
(5.16)	$(i,j,i+j)$	$(1 \leq i \leq n-1)$ and $(1 \leq j \leq c-1)$	VMs and containers are partially occupied and the queue is empty.

Source: The author (2021)

5.3 PERFORMANCE METRICS

In Chapter 4, four key performance indicators were studied and modeled for that specific framework. In this section, a similar steady-state analysis of the CTMC is considered, followed by the derivation of the same performance metrics: Availability (A), Reliability (R), Power Consumption (C) and Response Time (T). However, since the CTMC differs from that of Chapter 4, new equations were derived.

5.3.1 Availability (A)

The Availability is the system's ability to offer the minimum amount of functional and accessible VNFs. In particular, a VNF instance is considered available if at least one of its constituents (VM-hosted or containerized) remains accessible. With regards to the queuing system, the MEC node Availability (A) (5.17) is obtained by the probability sum of all states except those representing full capacity.

$$A = 1 - \sum_{i=0}^{n} \sum_{j=0}^{c} \pi_{i,j,K} \tag{5.17}$$

5.3.2 Reliability (R)

The Reliability (R) is given by (5.18), which combines the admitted flow λ*A with the effective failure rate in the entire node, i.e., denotes the probability that a service is served without experiencing failures while being processed by MEC VNFs.

$$R = 1 - \frac{\gamma}{\lambda * A} \sum_{j=1}^{c} j \left[\sum_{i=0}^{n} \sum_{k=1}^{K} \pi_{i,j,k} \right] \tag{5.18}$$

5.3.3 Power Consumption (C)

The power consumption (C) Eq. 5.24 is formed from the combination of the mean number of virtual resources and power consumption values (P) for each virtualization technology (VM and Container) and operating states (Idle, Setup and Busy). Thus, the power consumption (in Watts) of a single VM in idle state is denoted P_{idle}^{VM}, and the remaining variables are represented with the same notation. However, in order to obtain the aforementioned metric, the mean number of VMs and containers (CT) in each state are described in Eqs. 5.19-5.23. Again, since VMs are constantly powered, no equation for its Setup state is described.

$$\overline{VM}_{idle} = \sum_{i=0}^{n} (n-i) \left[\sum_{j=0}^{c} \sum_{k=0}^{K} \pi_{i,j,k} \right] \tag{5.19}$$

$$\overline{VM}_{busy} = \sum_{i=0}^{n} i \left[\sum_{j=0}^{c} \sum_{k=0}^{K} \pi_{i,j,k} \right] \tag{5.20}$$

$$\overline{CT}_{idle} = \sum_{j=0}^{c}(c-j)\sum_{i=0}^{n}\pi_{i,j,k}$$
$$+\sum_{j=0}^{c}\sum_{i=0}^{n}\sum_{k=i+j+1}^{K}(c-j-min(k-i-j,c-j))\pi_{i,j,k} \tag{5.21}$$

$$\overline{CT}_{setup} = \sum_{j=0}^{c}\sum_{k=n+j+1}^{K} min(k-n-j,c-j)\pi_{n,j,k}. \tag{5.22}$$

$$\overline{CT}_{busy} = \sum_{j=1}^{c} j\left[\sum_{i=0}^{n}\sum_{k=j+i}^{K}\pi_{i,j,k}\right]. \tag{5.23}$$

$$C = P_{idle}^{VM}\overline{VM}_{idle} + P_{busy}^{VM}\ \overline{VM}_{busy}$$
$$+P_{idle}^{CT}\ \overline{CT}_{idle} + P_{setup}^{CT}\overline{CT}_{setup} + P_{busy}^{CT}\ \overline{CT}_{busy} \tag{5.24}$$

5.3.4 Response Time (T)

The Response Time (T) is defined exactly as in 4: the interval between the service arrival (on the edge node) and its departure (service conclusion) and is obtained by calculating the mean number of online services and the mean number of accepted services (5.25).

$$T = \frac{1}{\lambda * A}\sum_{k=0}^{K} k\left[\sum_{i=0}^{n}\sum_{j=0}^{c}\pi_{i,j,k}\right] \tag{5.25}$$

5.4 MODEL VALIDATION

The simulation structure is quite similar to that of the base analytical model presented in Section 4.4.1, differing only from the programmed logic to meet the specific design. Again, the analytical results were validated against discrete-event simulations (Figs. 22-33), where the lines denote the analytical and the markers represent simulation results. With regards to the main parameters, we have followed a subset of the 3GPP Release 16 (TR 38.824) [3GPP 2020], in which the service rates μ and μ_c are 1 (1 service/ms), whereas μ_v is calculated in (5.1).

Table 15 – Power Consumption Values

Parameter	Value
Idle VM Energy Consumption (P_{idle}^{VM})	20 W
Busy VM Energy Consumption (P_{busy}^{VM})	25 W
Idle CT Energy Consumption (P_{idle}^{CT})	4 W
Setup CT Energy Consumption (P_{setup}^{CT})	8 W
Busy CT Energy Consumption (P_{busy}^{CT})	23 W

Source: Morabito (2015)

Each scenario simultaneously evaluates the impact of a pair of the following parameters: Multiple VM Amounts (n) and Overhead Degradation Factor (d),multiple container amounts (c) and failure rates (γ), and multiple buffer sizes (q) and container setup rates (α), with the service arrivals ranging from 1 up to 100 requests/ms. In addition, unless otherwise stated the baseline values for failure (γ) and setup rates (α) were 0.001 and 1 unit/ms, respectively, which is in accordance with [Kaur et al. 2017]. In terms of power consumption for VMs and containers for different operation states, we adopted the values from the network intensive experiment in [Morabito 2015], which is summarized in Table 15. The remaining parameters can be found in Table 16.

Table 16 – Experiment Sets

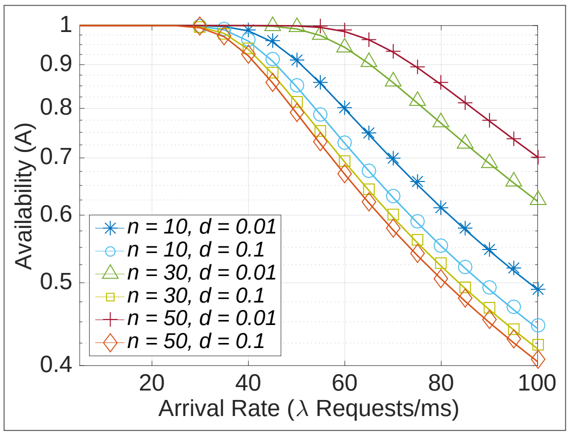
Varying Parameters	n	c	q	d	γ	α
n, d	10,30,50	40	10	10^{-2},10^{-1}	10^{-3}	1
c, γ	10	40,60,80	10	10^{-2}	10^{-3},10^{-2}	1
q, α	10	40	10,30,50	10^{-2}	10^{-3}	10^{-1},1

Source: The author (2021)

5.4.1 Multiple VM Amounts (n) and Overhead Degradation Factor (d)

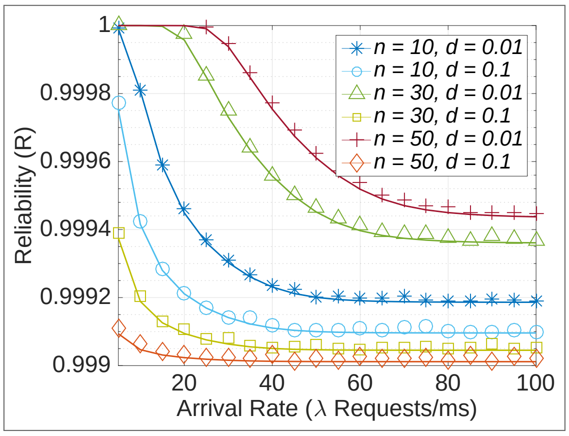
In Figs. 22-23 both Availability and Reliability have similar configuration disposal, however with large magnitude differences between them. In Fig. 22, the combined increase of n and d implies a less available system due to a lower VM service rate. This also means that containers are likely to be more required, meaning that failures happen more often, leading to a less reliable system (Fig. 23). Comparing the Availability for configurations (($n = 10$), ($n = 30$)) and (($n = 30$), ($n = 50$)) that share $d = 0.01$, the absolute difference is 11.5% and 9.1%, respectively, whereas in Fig. 23 the difference among the Reliability from the same curves is much smaller in absolute terms, which would be relevant URLLC.

Figure 22 – Availability



Source: The author (2021)

Figure 23 – Reliability

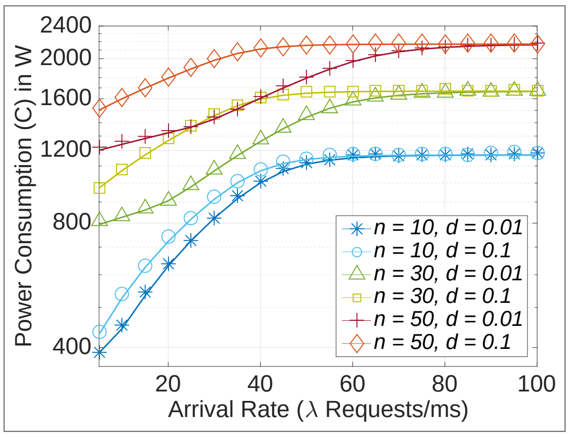


Source: The author (2021)

In Fig. 24, both the maximum number of VMs (n) and overhead degradation factor (d) greatly impacted power consumption, but in different ways. Since VMs are not dynamically scalable, it was expected that the increase in n would also increase overall power consumption, regardless of λ. On the other hand, the overhead degradation factor (d) clearly impacts the non-saturated system, and particularly in this scenario has only impacted configurations with higher n values (30 and 50), respectively. For values of λ tending to 100, the curves with same n overlap, which evinces a saturated system, i.e., all resources are processing services although each pair of curves has different throughput due to the different values for d. With respect to the differences in power consumption, the two curves with equal n values: $(n = 30, d = 0.01)$ and $(n = 30, d = 0.1)$ present a 500W maximum gap in $\lambda = 35$, whereas the difference in $(n = 50, d = 0.01)$ and $(n = 50, d = 0.1)$ is approximately $600W$ in the same reference point, which are quite significant since each of them has the same VM amount. Another particularity is observed when comparing the curves $(n = 30, d = 0.1)$ and $(n = 50, d = 0.01)$ for λ values ranging from $= 20$ to 50, which performed similarly regardless of the difference in the total number of resources. In brief, in addition to account that the impact of d is potentialized by n, higher d values also makes the overall service rate lower, leaving VMs in Busy mode for longer periods besides forcing the system to activate containers more often.

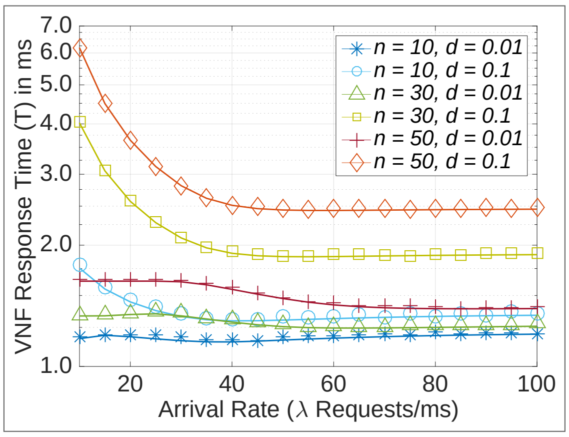
Unlike the power consumption, the changes in n have very little impact on the Response time (Fig. 25) if compared to the results of the curves with different d. This indicates that the positive effects of increasing n can be mitigated depending on d. In general, there are multiple nuances involving each parameter, service load and performance metrics that might

Figure 24 – Power Consumption



Source: The author (2021)

Figure 25 – Response Time



Source: The author (2021)

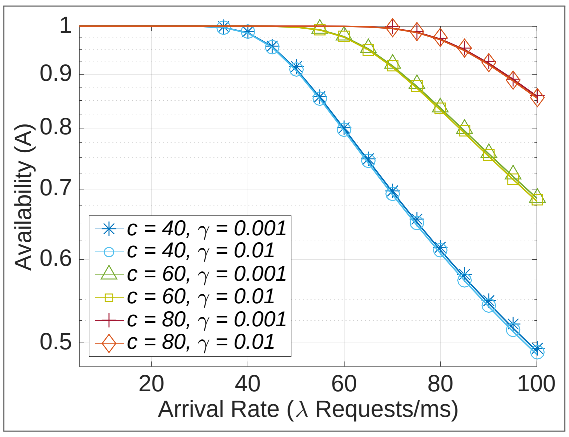
be conflicting, indicating the relevance of an adequate dimensioning process. For instance, comparing the two curves with $n = 50$ the Response Time almost doubles. Moreover, considering Availability and Reliability the curve with $(n = 50,\ d = 0.01)$ had the best performance, whereas for Power Consumption it was one of the worse.

5.4.2 Multiple Container Amounts (c) and Failure Rates (γ)

A larger γ means smaller intervals between successive containerized VNF failures. In contrast to α, this parameter improves the system as it gets smaller. Thus, γ was enhanced by a factor of 10 and 100. In addition, containers are not prone to significant overhead issues unlike VMs, and besides, there is no correlation between the container amount and its individual service rate. For this resource type, failures are isolated events. In this scenario, we have reduced and fixed the total amount of VMs to ten units $(n = 10)$ to highlight the impact of c and γ. Thus, for small $\lambda < 10$ most services are processed in VMs, not having a great influence on the results.

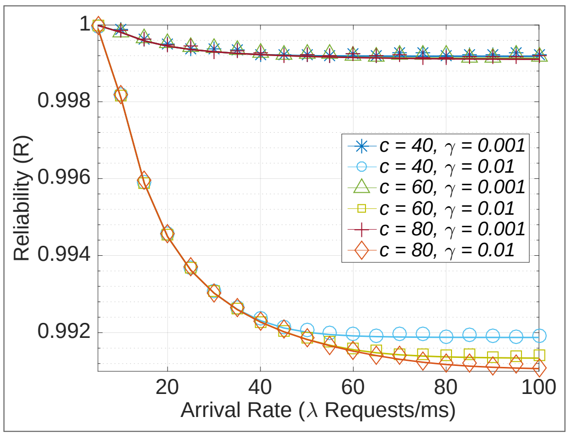
It was verified that γ values have little influence on the Availability (Fig. 26), Power Consumption (Fig. 28), and even in the Response Time (Fig. 29) for the assumed ranges. This does not means that this parameter has no impact of those metrics; but in order for this to happen, γ must be at least of the same order as α, making containers failure more frequently, thus rapidly deteriorating the system capacity. Otherwise, the container amount c has highly impacted those metrics, especially for values of λ tending to 100, where the maximum differences in each figure were registered. As for the Reliability (Fig. 27), the failure

Figure 26 – Availability



Source: The author (2021)

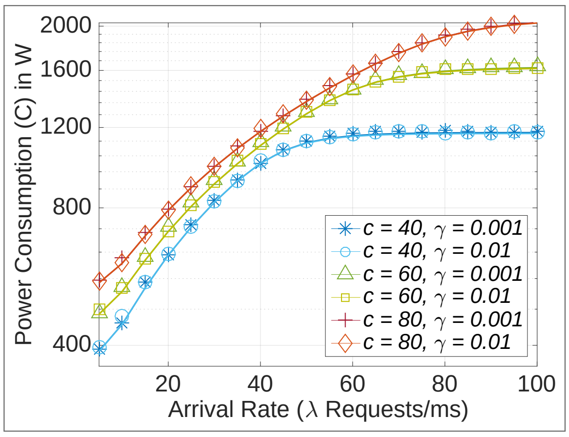
Figure 27 – Reliability



Source: The author (2021)

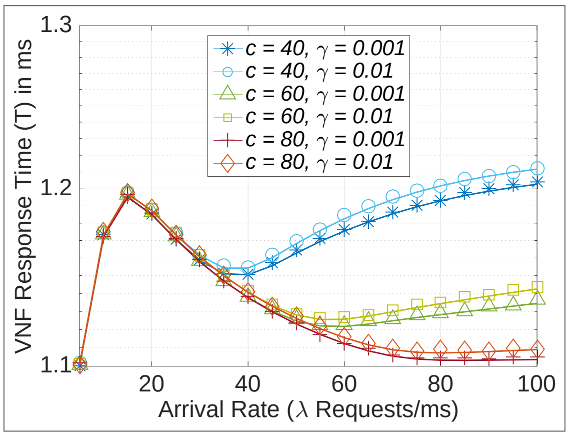
rate γ remains as one of its key components and hence significantly impacts the curves while c has little influence. In addition, compared to the Reliability in the previous experiment Fig. 23, the maximum absolute difference between curves is much higher (0.007).

Figure 28 – Power Consumption



Source: The author (2020)

Figure 29 – Response Time



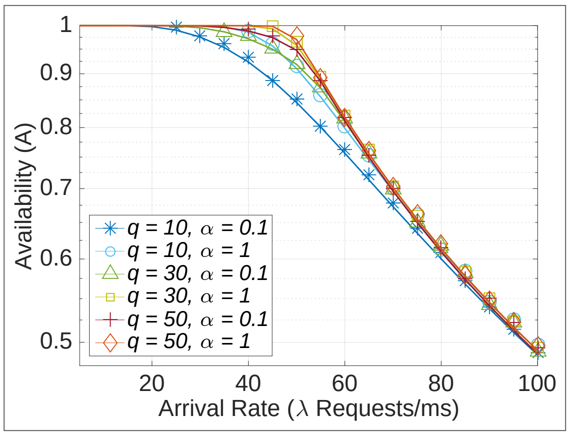
Source: The author (2020)

In Fig. 29, the Response Time spikes for $\lambda = 10$, which corresponds to the fixed VM maximum amount $(n = 10)$. From this point on, the containers are turned on, which explains the sudden spike due to the setup time and moreover the sudden drop, when part of the these resources are available and processing incoming services. Lastly, in each pair of curves there is a slight increase on the Response Time in different λ which is explained by the resource limit and further queue activation, which is not a processing unit. Compared to the Response Time in the previous experiment Fig. 25, it is noticeable that the maximum difference between curves is much less significant (less than 0.1 ms).

5.4.3 Multiple Queue Sizes (q) and Container Setup Rates (α)

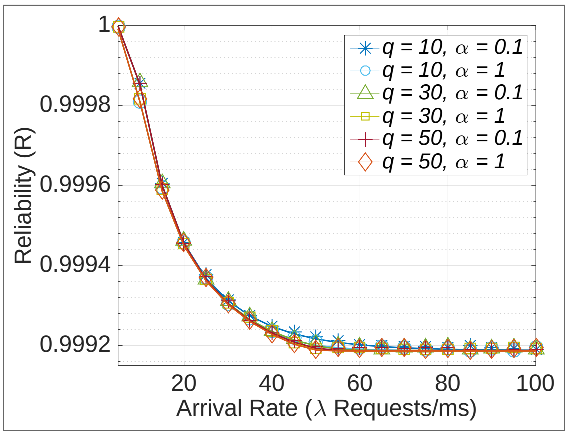
Larger α and q values mean smaller container setup delays, i.e., more VNF instances become available per unit time and greater buffer capacity, respectively. The results to follow could have been amplified for larger q values or for a higher ratio between α and γ, however, in order to keep the same total amount of resources as in previous experiments, we have opted for $q = 10, 30$ and 50, followed [3GPP 2020] to select appropriate values for α and γ. In this way, most configurations performed similarly for the Availability (Fig. 30), except the curves with $q = 10$. In fact, Availability, Power Consumption (Fig. 32) and Response Time (Fig. 33) are likely to increase when $q \rightarrow +\infty$ but in much different rates. On the other hand, the Reliability (Fig. 31) is barely affected by the increase on q.

Figure 30 – Availability



Source: The author (2021)

Figure 31 – Reliability

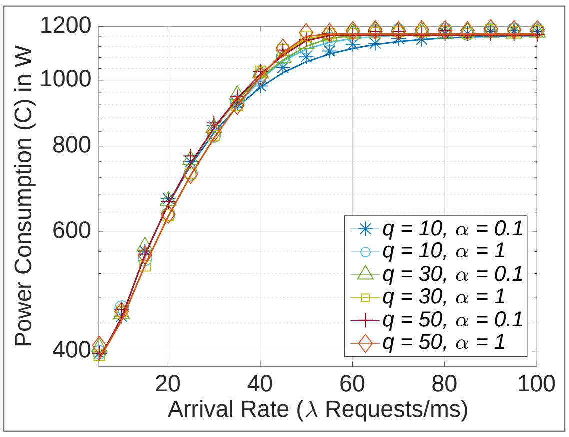


Source: The author (2021)

With regards to α variations, the Availability (Fig. 30) difference from the curves in which "$q = 10$" were significant, similarly to the results for different q. This was somehow expected since higher α mean more available containers per unit time. On the other hand, the reliability results (Fig. 31) were not significantly affected by either parameters. We believe that the ratio between α and γ has contributed for this result; if γ becomes higher, different α may result in great differences towards the Reliability being the curves with higher α the most likely to present lower reliability. Although the difference is quite small, one may note that the configuration $q = 10, \alpha = 0.1$ was the most reliable since less containers are serving per unit time, and thus there are less failures.

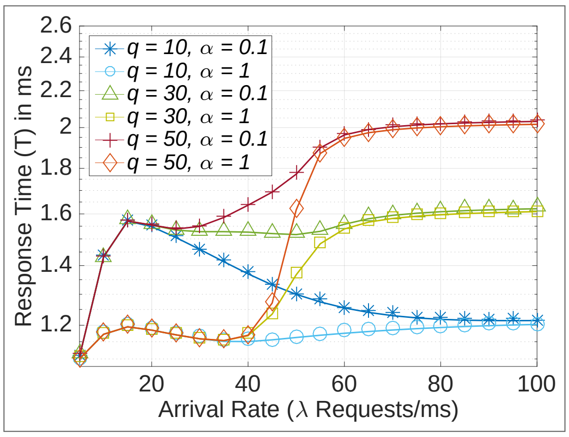
With respect to the Response time in Fig. 33, two configuration groups overlap until $\lambda = 20$ and $\lambda = 40$. Then, the groups are split end up forming 3 pairs of curves each of which with the

Figure 32 – Power Consumption



Source: The author (2021)

Figure 33 – Response Time



Source: The author (2021)

same q, crossing each other for multiple λ. The reason for such a different behavior is related to the resource availability empowered by higher α and the use of queue to store service surplus. For this reason, configurations with higher α and lower q are likely to respond much faster. For $\lambda \to +\infty$, α becomes irrelevant since the containers are continuously processing services only resetting in case of failure. On the other hand, higher q values allows the system to store more services without actually processing them, which incurs in higher response times.

5.5 CHAPTER SUMMARY

In this chapter we described additional elements that compose a single MEC node virtual environment with extra assumptions, namely (a) separate service rates for VMs and containers and (b) service migration. The new assumptions yielded a more realistic model, especially considering that in the previous model VMs were considered perfect computational resources, i.e., no failures and no further characteristics that could cause negative impacts. In addition, the CTMC was also altered so as to add some flexibility to the model and to facilitate comprehension by using three variables instead of only two as described in the previous chapter. For the next chapter, the multi-objective problem of MEC node dimensioning is formulated and a scheme based on genetic algorithms to solve it is proposed and evaluated.

Table 17 – Chapter 5 Summary

Section	Goal(s)	Output(s)
5.1 SYSTEM MODEL	Describe the top-level elements of the system to be modelled and the key differences from the previous model from 4.	Virtual host assumptions: setup/recovery delays, failures, dynamic VNF scaling and processing overhead (5.1).
5.2 ANALYTICAL MODEL	Describe the CTMC-based model of a single MEC node for URLLC with VMs and Containers scaled upon demand.	a) Steady State Analysis (transitions and state space) illustrated in Fig.21; b) normalization condition (5.2) and balance equations (5.3)-(5.16), summarized in Table 14.
5.3 PERFORMANCE METRICS	a) Explain the importance of each adopted metric: Availability (A), Reliability (R), Power Consumption (C) and Response Time (T); b) Detail each metric formulation.	Expressions: MEC node Availability (A) (5.17), Reliability (R) (5.18), Power Consumption (C) (5.24). and Response Time (T) (5.25).
5.4 MODEL VALIDATION	a) Describe the adopted discrete-event simulator; b) Validate results in the following scenarios: 1) Multiple VM Amounts (n) and Overhead Degradation Factor (d), 2) Multiple Container Amounts (c) and Failure Rates (γ) and 3) Multiple Queue Sizes (q) and Container Setup Rates (α).	a) First scenario: Multiple nuances involving each parameter, service load and performance metrics that might be conflicting, indicating the relevance of an adequate dimensioning process. b) Second Scenario: the container amount c has highly impacted those metrics, especially for values of λ tending to 100, where the maximum differences in each figure were registered. c) Third scenario: higher q allows the system to store more services without processing them, which incurs in higher response times.

Source: The author (2021)

6 A SCHEME BASED ON GENETIC ALGORITHMS FOR MEC NODE DIMENSIONING

Based on the designed models described in Chapters 4-5, we have analyzed the influence of various system parameters on different performance metrics. Those experiments required previous knowledge of all input parameters, however, it is also possible to select part of those parameters to be optimized. Although multiple subsets could have been selected, this chapter aims at the problem of optimizing the MEC node dimensions (size) in terms of its resource components, i.e., maximum number of VMs, containers and buffer positions. In addition, the scheme takes into account the model and metrics formulated in Chapter 5, i.e., the model extension, which best represents the MEC node operation.

This chapter starts with a motivation scenario again emphasizing the role of MEC on future mobile networks (Section 6.1), but embraces a new paradigm shift known as MEC-enabled UAV that is part of the space-air-ground integrated network effort [Zhao et al. 2021]. Then, in Section 6.2 the dimensioning optimization problem is formulated and characterized as a multi-objective problem, which considers multiple aspects discussed in Chapter 2. In Section 6.3, a scheme based on genetic algorithms is proposed to solve the multi-objective problem. It sets out by describing the basis GA elements: chromosome structure and fitness function, as well as its remaining operators: Selection, Crossover and Mutation. Following this, the genetic operators and test results that are needed to select the mutation and crossover probability values are presented together with the execution flow of the proposed scheme in Section 6.3. Finally, Section 6.4 contains two parts: the first is dedicated to the GA convergence analysis and the second to the results and comments on the comparison between the proposed scheme and other strategies to obtain the most suitable MEC node dimensions.

6.1 MEC-ENABLED UAVS

Cloud resource management has been a key factor for since the early datacenter development. Improper resource management results in under or over-utilized resources, which may lead to poor service quality. The scenario of MEC-enabled UAVs draws particular attention since it has limited resources and battery life. On the other hand, contrarily to fixed infrastructure, MEC-enabled UAVs yield a flexible solution to meet the stringent demands from critical applications such as in URLLC [Islambouli and Sharafeddine 2019]. Owing to the mobility, UAVs enable edge servers/core functions to fly closer to UEs, establishing Line-of-Sight (LOS), which assures the best connectivity conditions towards channel quality and consequently higher transmission rates and reliability [Li et al. 2019]. Thus, the main challenges rely on the limited resource and battery life, which leads to the importance of efficient resource dimensioning.

The role of UAVs on enhancing future networks can be widely diverse, with UAVs acting as radio, core NFs, edge cloud servers or backhaul equipment [Yang et al. 2019]. The benefits of co-locating these capabilities on a single UAV include improvements on service latency and signaling overhead, while the limitations are mainly related to energy consumption.

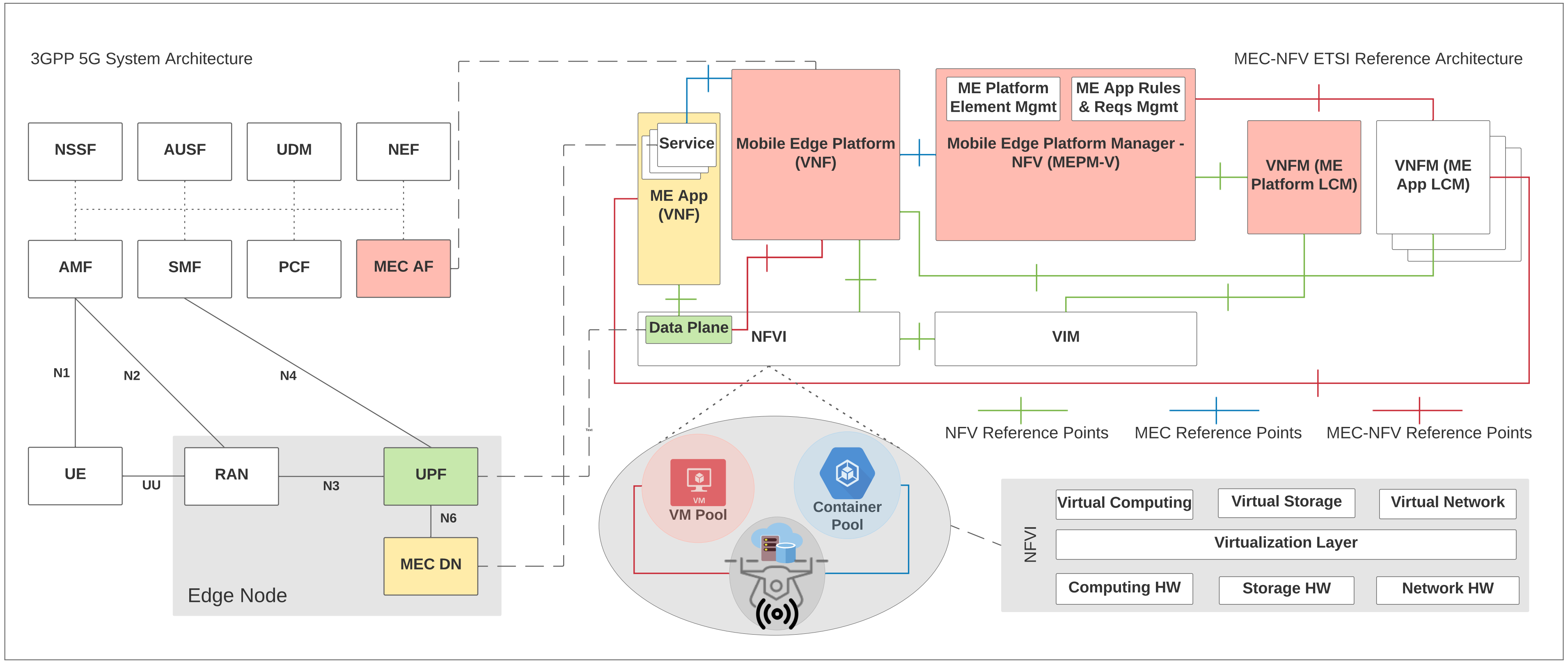
UAV-enabled NFs complement ground core network, facilitating cooperation. Core NFs on UAVs can reduce latency while ensuring stable connectivity, also providing a means of collecting local environment data for enriching network analytics, e.g. for QoS assurance. Given the natural energy limitations on UAVs, core NFs that hold user registration data, policy and authentication are not suitable to be hosted on UAVs to avoid losing data.

Typical core NFs that can be hosted on UAVs include Control plane functions e.g., Access and Mobility Function (AMF), Session Management Function (SMF) and Network Exposure Function (NEF), which are depicted in Fig. 34. This figure describes an example of a MEC-NFV deployment based on the ETSI [Kekki and Featherstone 2018] in a 3GPP 5G [3GPP 2020] system, where the MEC platform is implemented as an Application Function (AF) and the MEC data plane is as a particular implementation of a 5G User Plane Function (UPF) that forwards the traffic to MEC applications running in the Data Network (DN).

Previous studies proposed different solutions for optimal use of UAV as network infrastructure. In [Zhang et al. 2019], the authors used stochastic optimization tools for energy consumption minimization on both UE and UAV while optimizing the horizontal trajectory of the vehicle. In [Costanzo and Lorenzo 2019], a dynamic communication and computation allocation strategy is proposed for selecting the ideal altitude and minimizing the UAV energy consumption while concomitantly satisfying latency constraints. In [Yang et al. 2019], the authors propose MEC-enabled networks over multiple UAVs aiming at power consumption minimization and focusing on UAVs as backhaul and core network equipment. More recently, the authors in [Bekkouche et al. 2020] explored the multiple roles of UAVs on enhancing 5G networks, with UAVs acting as radio, core network, and edge clouds but focusing the proposed testbed on the latter case, i.e., the performance of UAVs as edge clouds hosting Aerial Control System (ACS) functions instances responsible for the control and orchestration of a UAV fleet.

While there has been significant attention paid to latency and energy consumption as-

Figure 34 – Data Plane architecture mapping example - 3GPP 5G system and MEC-NFV



Source: The author (2021)

pects of MEC-enabled UAVs based on trajectory optimization, computing failure resilience and resource availability has received far less attention even though these are of paramount importance for resource dimensioning. Motivated by this gap, we propose a resource optimization in terms of maximum number of VMs, containers and buffer size, representing a single MEC-enabled UAV node described in Chapter 5. In particular, we aim at maximizing the system's availability and minimizing power consumption while also satisfying reliability and latency constraints (for the case of URLLC), offering a complementary solution to those proposed by previous authors.

6.2 MEC NODE DIMENSIONING OPTIMIZATION PROBLEM

Different objectives must be considered for proper MEC-enabled UAV resource dimensioning, which may be formulated as an optimization problem as follows (6.1). Given a service demand characterized by arrival (λ) and service rates when running on virtual machines and containers (μ_n and μ_c, respectively) and the containerized VNF setup and failure rates (α and γ, respectively), it determines the most appropriate node dimension in terms of number of virtual machines, containers and buffer size ($x^* = [{x_1}^*, {x_2}^*, {x_3}^*] \in X$) that maximizes the system's availability (A) and minimizes power consumption (C) while also satisfying reliability (R) and response time (T) constraints (R_{min}, T_{max}). X is the problem's feasible solution space, x_1^i, x_2^i, x_3^i denote the number of VMs, CTNs and buffer size that compose the solution x^i, which are defined within $[Imin_j, Imax_j]$, with $j = 1, 2, 3$, respectively.

$$\begin{aligned} Maximize\ A(x)\ and\ Minimize\ C(x) \\ Subject\ to\ R(x) \geq R_{min}, \\ T(x) \leq T_{max}, \\ x^i \in X, x_j^i \in \mathbb{Z}^+, \\ with\ j = 1, 2, 3\ and\ x_j^i \in [Imin_j, Imax_j] \end{aligned} \tag{6.1}$$

Focusing on a specific objective may deteriorate others or violate constraints. To mitigate such effect, we propose an evolutionary scheme based on GA that copes with the given conflicting objectives and constraints that uses an adjusted dominance concept. In this work, we assume that the GA is adopted for the dimensioning phase, i.e., before proper operation. The scheme should be executed on a separate structure such as a ground MEC node or central cloud since the dimensioning phase may require additional resources and is usually not a critical, i.e., once the limits are defined, the scheme is not required to run run during the operation.

6.3 GA OPERATIONAL DETAILS

The GA relies on evolving a solution set given (chromosomes) through the combination of several possible sets through its operators: selection, crossover and mutation. In this work, the GA handles multiple MEC-enabled UAV node configurations simultaneously in each interaction.

6.3.1 Chromosome Structure and Fitness Function

In order to handle this multiobjective problem and reduce its complexity, we adopted a fitness function based on the dominance concept [Deb 2011], which takes each individual and compares it to the remaining population in terms of the already known four performance metrics (A,R,C and T).

The individual x (chromosome) represents the maximum number of VMs, CTNs and buffer size for the solution x^i, i.e., $x_1^i, x_2^i, x_3^i \in \mathbb{Z}^+$. The possible node configurations evolves as different generations and produces more appropriate solutions represented by the set of resources. In this respect, each individual x^i is compared to the entire population. Hence, equations (6.3)-(6.6) describe the number of dominated individuals for each metric.

In addition to the points scored in each comparison, the individual's fitness is increased by another constant ω_r and/or $\omega_t \in \mathbb{R}^+$, if the Reliability (R) and Response Time (T) overcome a predefined threshold (R_{min} and T$_{max}$, respectively), which is given by (6.7) and (6.8). For the population size considered in the following experiments (50), $\omega_r = 1$ and $\omega_t = 1$ represented reasonable values and thus were assumed. The resulting fitness function (6.2) sums up all the above mentioned parts.

$$\begin{gathered}
Fitness(x^i) = \sum_M D_M(x^i) + \sum_S B_S(x^i), \\
with\ M\ being\ A, R, C,\ and\ T, \\
S\ assuming\ R\ and\ T, \\
i, j \in \mathbb{Z}^+ and\ i, j \leq L, \\
where\ L\ is\ number\ of\ individuals
\end{gathered} \tag{6.2}$$

$$D_A(x^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 & ,\ if\ A(x^i) \geq A(x^j) \\ 0 & ,\ otherwise \end{cases} \tag{6.3}$$

$$D_R(x^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 & ,\ if\ R(x^i) \geq R(x^j) \\ 0 & ,\ otherwise \end{cases} \tag{6.4}$$

$$D_C(x^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 & ,\ if\ C(x^i) \leq C(x^j) \\ 0 & ,\ otherwise \end{cases} \tag{6.5}$$

$$D_T(x^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 & ,\ if\ T(x^i) \leq T(x^j) \\ 0 & ,\ otherwise \end{cases} \tag{6.6}$$

$$B_R(x^i) = \begin{cases} \omega_r & ,\ if\ R(x^i) \geq R_{min} \\ 0 & ,\ otherwise \end{cases} \tag{6.7}$$

$$B_T(x^i) = \begin{cases} \omega_t & ,\ if\ T(x^i) \leq T_{max} \\ 0 & ,\ otherwise \end{cases} \tag{6.8}$$

6.3.2 Genetic Operators and Parameters

We adopted the classical GA operators, i.e., the roulette wheel for selection and uniform operator for the crossover, hence, the highest fitness individuals are more likely to move to the next generation while creating new individuals from the crossover process, while a bit mutation operator was also employed [Deb 2011].

Multiple tests were conducted to define the crossover (P_c) and mutation (P_m) probabilities. Five values within the intervals [0.2, 1] and [0.01, 0.09] were tested for P_c and P_m, respectively

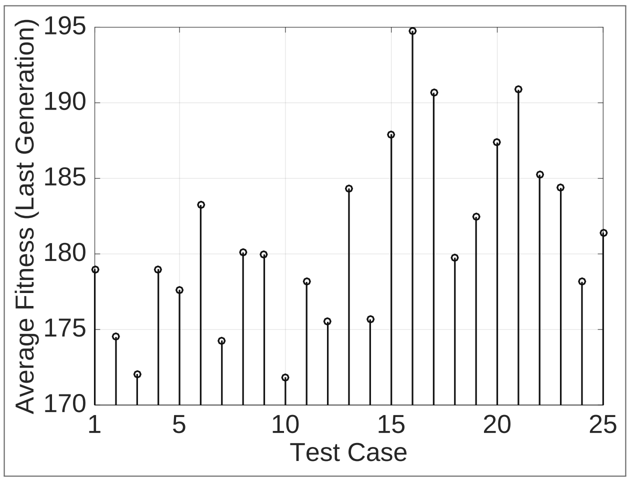
Table 18 – Crossover rate and Mutation Probability Tests

Parameter	Value
Crossover probability (P_c)	0.2/0.4/0.6/0.8/1
Mutation probability (P_m)	0.01/0.03/0.05/0.07/0.09

Source: The author (2021)

(Table 18). The test scenario was composed of up to 50 VMs, 50 Containers and 50 Buffer positions, with the remaining parameters being described in Table 20) and λ was fixed to a mid range value of 50. For each test case, five simulation instances were performed.

Figure 35 – Test Cases for Pc and Pm



Source: The author (2021)

The highest average fitness value for the last generation's population was selected for each test case (Fig. 35). The best performance was obtained by test case 16, in which P_c and P_m assumed 0.8 and 0.01, respectively. Thus, these values were applied in the next section together with the population size (L) and number of generations (G), which were set to 50 and 100, respectively. The GA parameters are summarized in Table 19.

Table 19 – Selected GA Parameters

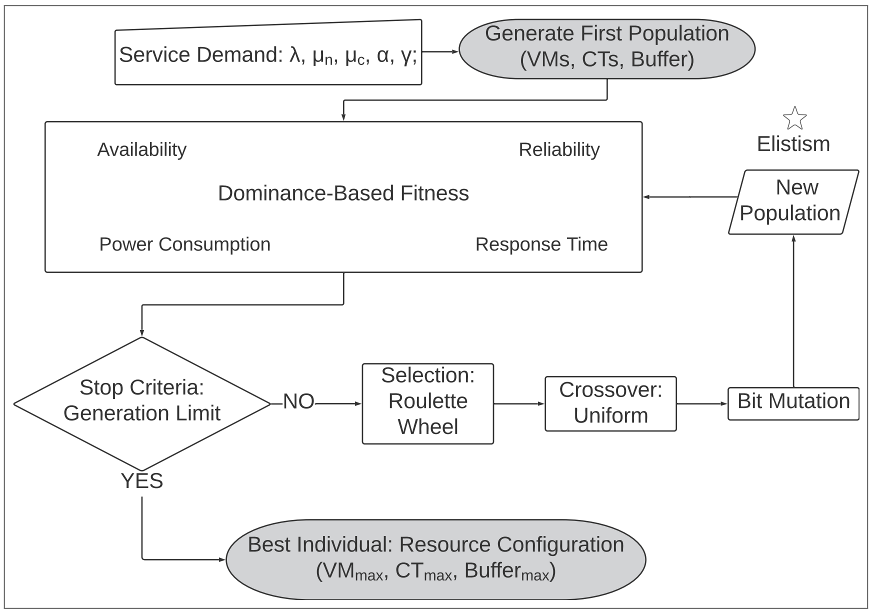
Parameter	Value
Population size (L)	50
Generations (G)	100
Crossover probability (P_c)	0.8
Mutation probability (P_m)	0.01

Source: The author (2021)

6.3.3 GA Scheme Execution Flow

Given a maximum estimated service load (requests/ms) and resource limits for VMs, Containers and Buffer size, an initial set of candidate solutions is randomly generated (first population). Then, the individuals are evaluated using the dominance-based fitness function (6.2), which takes four performance metrics, besides an additional score for each threshold that it overcomes. Moreover, the operators: selection, crossover and mutation are applied in this order, generating an entirely new population. In addition, to ensure that the best individual will not be lost during the selection process, we have also employed elitism, i.e., the most fit individual is guaranteed in the next generation unchanged. Finally, the process repeats until the maximum number of generations (G) is reached (stop criterion). Then, the best individual is chosen as the final solution, which represents the most suitable configuration (maximum number of VMs, Containers and Buffer size). The GA execution flow is depicted in Fig. 36.

Figure 36 – GA Scheme Execution Flow



Service Demand: λ, μ_n, μ_c, α, γ;
Generate First Population (VMs, CTs, Buffer)
Availability
Reliability
Dominance-Based Fitness
Power Consumption
Response Time
Elistism
New Population
Stop Criteria: Generation Limit
NO
Selection: Roulette Wheel
Crossover: Uniform
Bit Mutation
YES
Best Individual: Resource Configuration (VM_{max}, CT_{max}, $Buffer_{max}$)

Source: The author (2020)

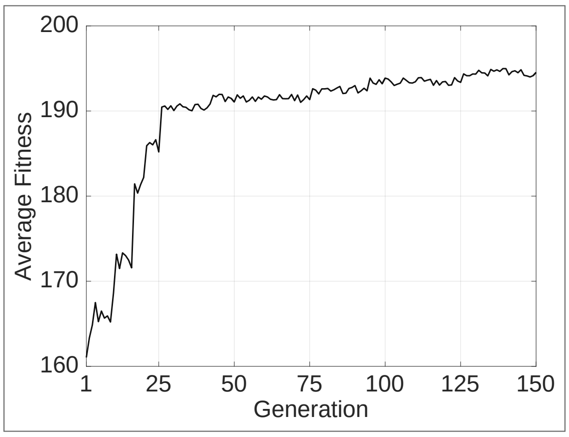
6.4 SCHEME EVALUATION

The following lines describe the evaluation scenarios, convergence time of the proposed GA scheme and its results compared to two other approaches that are based on the first-fit strategy. The evaluation scenarios share the same parameter values such as service, failure and setup rates, as well as the same resource limits, but differ in terms of request loads, being characterized as Low, Mid and High. The goal is to assess the proposed scheme and the other strategies under different request rates.

6.4.1 GA Convergence

The GA convergence was examined with regards to the population average fitness. To this end, an intermediate load fixed value $\lambda = 50$ was selected. We extended the GA's evolution process by adopting 150 generations to verify if the average fitness would significantly change after 50 generations (see Fig. 37). For each point, 30 instances were performed and the average results are presented considering a 95 confidence level, which were obtained by the Bootstrap method, with 'resample' size and number of (re)samplings equal to 30 and 1000, respectively. No bars were drawn due to a small difference between upper and lower bounds.

Figure 37 – GA Convergence Test



Source: The author (2021)

6.4.2 Evaluation Scenarios

It was observed that the average fitness increases sharply in the first 25 generations as the GA explores the search space. In the next 75 generations (from the $25th$ approximately up to the $100th$), the average fitness also rises but in a much longer slope, which indicates that the GA is refining already existing solutions. Lastly, from generation 100 onwards, the average fitness seems to becomes stable, with no significant changes taking place, denoting that the individuals from these populations have similar fitness values.

The evaluation takes multiple network loads (λ) segmented in Low, Mid and High as follows. The request arrival rate of each channel was defined within the interval 10 to 30 for the low load, 40 to 60 for intermediate (mid) load and 80 to 120 for high range. A total of 10 values of λ are drawn for each range and the experiment is repeated 10 times. Then, the mean values for each performance metric (A,R,C and T) in each of these scenarios are calculated.

The remaining parameters are kept the same for each scenario as follows: container setup and failure rates α and γ are fixed 0.1 and 0.001, whereas the container service rate is fixed to 1. However, since the adoption of the degradation factor $d = 0.01$, the VM service rate is defined in (5.1), considering the baseline rate $\mu = 1$. Hence, its final value depends on the adopted number of VMs, which is varied between 1 and 50 with step of 1 unit. The same range and step are applied to the containers, while the buffer size varies between 10 and 50 with a step of 10. Thus, the size of the search state space if given by $5(50^2) = 12,500$ possible resource configurations. These parameters are summarized in Table 20. Please assume that the power consumption constants are the same as in Table 15.

Table 20 – Scheme Evaluation Parameters

Parameter	**Value**
VMs Range	[1,50]
CTs Range	[1,50]
Buffer Sizes	[10,20,30,40,50]
VM/CT Service rates	1
Overhead Degrad. Factor (d)	0.01
Container Failure Rate (γ)	0.001
Container Setup Rate (α)	0.1
λ Low Range	[10,30]
λ Mid Range	[40,60]
λ High Range	[80,120]
Reliability Threshold (R_{min})	0.9995
Response Time Threshold (T_{max})	5 ms

Source: The author (2021)

In addition to these different scenarios, we analyzed the effectiveness of our scheme by comparing it to schemes based on the first-fit strategy, similar to [Emara et al. 2021]. The First-Fit scheme (denoted as FF) tests each configuration possibility, i.e., tries out every possible configuration set from 1 VM, 1 CT and 10 buffer spaces up to 50 VMs, 50 CTs and 50 buffer spaces. The first tuple that overcomes the predefined reliability (R_{min}) and response time (T_{max}) limits is considered the final answer. Please note that no restrictions were defined for the Availability nor the Power Consumption.

The First Fit strategy was segmented in two alternatives, which differ in terms of the exploration order of the available state space. While the FF_{min} searches for the appropriate configuration departing from the minimum established values, i.e., VM $= 1$, CT $= 1$ and buffer size $= 10$, upgrading first the VMs, second the CTs and last the buffer dimension, the FF_{max} tries the opposite, i.e., starts the search from the maximum configuration values, i.e., VM $= 50$, CT $= 50$ and buffer size $= 50$, downgrading first the buffer variable, then the containers and only then the VM amounts. Although both may reach the same answer,if only one matches the constraints, FF_{min} most likely results in configurations with a smaller number of resources, while FF_{max} should respond with a greater number of resources.

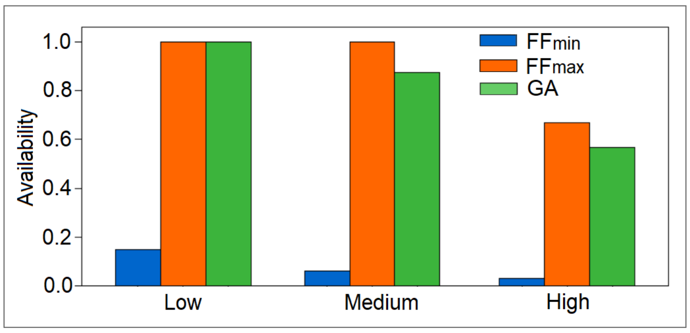
6.4.3 Numerical Results

This section draws a comparison between our GA-based scheme and the First-Fit variants FF_{min} and FF_{max}, defined in Section 6.4.4.

6.4.4 Evaluation Scenarios

Considering the Availability (Fig. 38), it was noted great discrepancy among scenarios (Low, Medium and High loads) and adopted schemes (FF_{min}, FF_{max} and GA). The first reason for such a great gap between their results is the lack of the inferior bound for this metric. For instance, the FF_{min}, is free to select any given resource configuration as long as it produces at least the predefined reliability (R_{min}) and response time (T_{max}). Hence, this scheme resulted in poor availability values (under provisioning) for each scenario (14.98%, 5.96% and 2.98%, respectively). The same applies to the FF_{max} scheme, except that it first handles large resource configurations, which will often result in over provisioned configurations, i.e, high availability (99.99%, 99.99% and 66.81%, respectively for low, medium and high loads). On the other hand, the GA scheme has a more balanced approach due to the design of its fitness function (6.2), achieving a similar result to the FF_{max} in the first scenario (low load), but then dropping to 87.42% and 56.85% in the medium and high load scenarios, resulting in a difference of 12.58% and 9.96%, respectively. This difference was somehow expected since maximizing the Availability directly impacts conflicting metrics (e.g., power consumption), which should be minimized. Thus, the GA opted opted for least available configurations so as to balance off these other performance metrics.

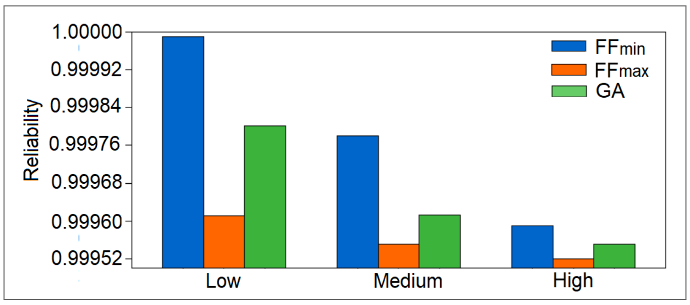
Figure 38 – Availability



Source: The author (2021)

With respect to the Reliability defined in (5.18), Fig. 39 shows that the FF_{min} scheme outperforms both FF_{max} and the GA. Again, this results reflects the GA's balanced approach, which approaches the FF_{min} in the low load scenario, whereas sits closer to the FF_{max} on the others. The FF_{min} scheme is far superior in both low and medium scenarios since it activates the minimum container amounts, thus very few failures occur. Contrarily, the FF_{max} will usually respond with a higher number of container in its configurations, which is likely to result in more failures. Lastly, the high load scenario presents the least significant difference among the schemes, although there were great discrepancies for the Availability in the previous experiment (Fig. 38). In other words, each scheme opted for very different resource configurations, yet, since the Reliability is dependant on the amount of failure-prone resources (containers) and that the high load forces container activation, the failure probability among each scheme is likely to converge to the inferior bound of 0.9995. However, this does not mean that the schemes converge in total number of containers since the Reliability only takes into account the requests that were accepted in the system, not the overall arrival rate.

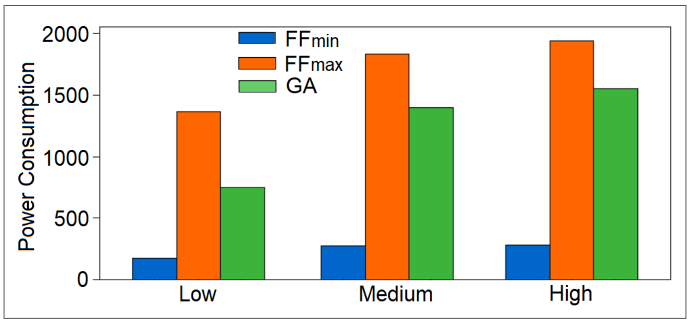
Figure 39 – Reliability



Source: The author (2021)

With regards to the motivational scenario of MEC-enabled UAVs, the power consumption metric is by far the most interesting. Indeed, this evaluation (Fig. 40) is closely related to the availability (Fig. 38), and in most cases, the schemes will share similarities among both experiments. For instance, the least available scheme (FF_{min}) is also the one that consumes less energy, whereas FF_{max} is the opposite, being the largest consumer. However, the GA, which shares a similar availability to the FF_{max} in the low load scenario and differs 9.96% from that same scheme in the high load case, differs 44% and 19% in terms of power consumption for the same low and high scenarios, respectively. The reason for the large gap relies on the fact that FF_{max} stops at the first resource configuration that meets the Reliability and Response Time thresholds, thus, large resource amounts can be adopted, whereas the GA tries multiple possibilities not only with respect to those limits but also minimizing power consumption.

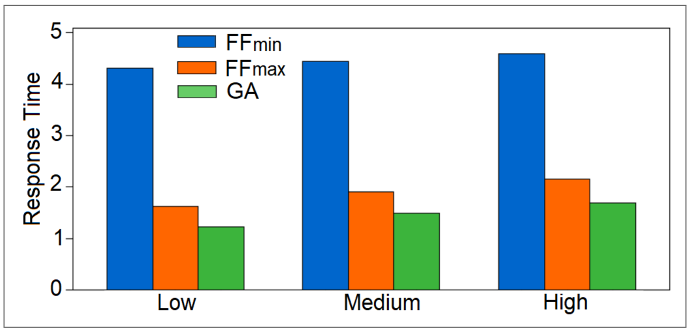
Figure 40 – Power Consumption



Source: The author (2021)

The last comment is dedicated to the results regarding the Response Time metric (Fig. 41). First, it was noted that the FF_{min} scheme responds with the highest delays, surpassing $4ms$ in each scenario, reaching $4.59ms$ in the high load case, which is close to the adopted superior bound of $5ms$. This denotes its design that most often will select configurations with higher number of VMs, which in turn, is the main cause for higher Response Times due to its associated degradation factor d. On the contrary, both FF_{max} and GA respond with much faster response times, under $2ms$ in most cases. However, please note that the GA overcomes the FF_{max} in every scenario, with large differences of 24%, 42% and 21% for the low, medium and high load scenarios, respectively. Again, we believe that two factors are key: (1) the availability (Fig. 38) and (2) the distribution of requests to each resource type. In the first case, the fact that the GA allows less requests to be processed in the medium and high load scenarios compared to the FF_{max}, with differences of 12.58% and 9.96%, respectively, allows gains in terms of response time of 42% and 21%, compared to the FF_{max}. On the latter, for the low load scenario, although both schemes share similar availability (close to 100%), The GA probably selects configurations with relatively more containers than VMs, which significantly decreases the overall response time (24%).

Figure 41 – Response Time



Source: The author (2021)

6.5 CHAPTER SUMMARY

In this Chapter, the multi-objective problem of MEC node dimensioning in terms of virtual resources was formulated and a scheme based on GA was designed to solve it. The scheme adopted a classical structure to deal with the objectives and to reduce the problem complexity. In addition, in order to verify its effectiveness, two schemes based on the First Fit approach were adopted and an evaluation was performed. Considering the context of MEC-enabled UAVs, the GA-based scheme achieves a reasonable tradeoff between the objectives and, in general, a better overall performance than First-Fit variants.

Table 21 – Chapter 6 Summary

Section	Goal(s)	Output(s)
6.1 MEC-ENABLED UAVS	a) Describe the role of MEC on MEC-enabled UAV. b) Explain the importance of resource optimization in terms of maximum number of VMs, containers and buffer size, representing a single MEC-enabled UAV node.	a) Data Plane architecture mapping example - 3GPP 5G system and MEC-NFV Fig. 34
6.2 MEC NODE DIMENSIONING OPTIMIZATION PROBLEM	Describe the proposed MEC-enabled UAV resource dimensioning problem.	The dimensioning optimization problem is formulated and characterized as a multi-objective problem with the formulation (6.1).
6.3 GA OPERATIONAL DETAILS	Describe the GA parameters and scheme flow (Selection, Crossover and Mutation).	a) The fitness function (6.2); b) Population size (L) of 50, Generations (G) of 100, Crossover probability (P_c) of 0.8 and Mutation probability (P_m) of 0.01. c) Scheme flow in Fig. 6.3.
6.4 SCHEME EVALUATION	a) Describe evaluation scenarios; b) Describe other strategies to be compared with our scheme. c) Compare proposed scheme results to those obtained by the First Fit strategies.	a) The evaluation takes multiple network loads (λ) and other parameters in 20; First Fit strategies: FF_{min} and FF_{max}. c) The proposed scheme achieves greater balance among the proposed performance metrics.

Source: The author (2021)

7 CONCLUSION AND FUTURE WORKS

This chapter concludes this thesis by offering some considerations, showing its main value as a contribution to studies in the field, and proposing future studies.

7.1 FINAL CONSIDERATIONS

We have addressed the combination of MEC, NFV and dynamic virtual resource allocation in the context of URLLC in order to overcome the problem of resource dimensioning in MEC-enabled UAVs. We took character of the MEC-NFV architecture into account to model and analyze how requests are processed by the underlying virtualization resources of a single MEC node, focusing on the limits of the URLLC service category. A CTMC-based model was proposed to characterize dynamic virtual resource allocation together with four performance metrics that are both relevant for URLLC (e.g., reliability and response time) and for service providers (e.g., availability and power consumption). In order to yield the model more practical, the effect of resource failures, setup (repair) times and processing overheads were embedded into the formulation, since they may greatly affect the stringent requirements of URLLC applications. Finally, a multi-objective problem related to MEC-enabled UAV node dimensioning in terms of virtual resources (VMs, containers and buffer positions) was formulated and a GA-based approach was adopted to solve it. The proposed scheme achieved a better tradeoff in terms of availability, reliability, power consumption and response time compared to the approaches based on the First-fit strategy. We hope that this work will encourage further research in the MEC-NFV domain, provide guidelines for the design of MEC-NFV architecture, business models and mechanisms that can help to deal with communication constraints.

7.2 CONTRIBUTIONS

The main contributions of this work can be summarized as follows:

- Description and classification on the main works in the field of MEC-NFV resource allocation focusing on mathematical models.
- Description of the main benefits and drawbacks related to the virtualization layer elements that compose the MEC-NFV environment.

- Modeling of a MEC-NFV node considering a hybrid virtualization layer that scales dynamically according to the load and formulation of the most relevant performance metrics related to URLLC.
- Formulation of the multi-objective problem in the context of MEC-enabled UAV node dimensioning in terms of virtual resources (VMs, containers and buffer positions).
- A GA-based scheme to solve the multi-objective problem and its evaluation.

7.2.1 Publication List

This section describes the author's list of published papers during the doctorate program (Table 22) and future publication plans. The first two papers in Table 22 [Falcao et al. 2021] and [Souza et al. 2021] were derived from the content of this thesis, especially with regards to the Related Works chapter (Chapter 3) and the description and validation of the base model (Chapter 4). The remaining works in Table 22 are mostly related to our previous topic of interest: Cognitive Radio Networks, however, it also involved related mathematical tools and/or optimization schemes in the field of mobile communications.

Table 22 – Publication List

Reference	Source	Title
[Falcao et al. 2021]	Journal of Supercomputing	An analytical framework for URLLC in hybrid MEC environments
[Souza et al. 2021]	IEEE Latin America Transac.	Modelling and Analysis of 5G Networks Based on MEC-NFV for URLLC Services
[Balieiro et al. 2021	IEEE LATINCOM	A Fuzzy-Genetic Approach for 5G/6G Opportunistic Slicing
[Balieiro et al. 2019]	Wireless Comm. and Mobile	An Evolutionary Scheme for Secondary Virtual Networks Mapping onto Cognitive Radio Substrate
[Falcao et al. 2018]	Computer Networks	A flexible-bandwidth model with channel reservation and channel aggregation for three-layered Cognitive Radio Networks
[Balieiro et al. 2017]	IEEE Comm. Letters	Secondary Virtual Network Mapping onto Cognitive Radio Substrate: A Collision Probability Analysis
[Falcao et al. 2017]	SBRC	Um Modelo de Largura de Banda Flexível para Redes de Rádios Cognitivos Baseadas em Prioridade
[Falcao et al. 2016]	IEEE LATINCOM	Three-layered prioritized cognitive radio networks with channel aggregation and fragmentation techniques

Source: The author (2022)

Furthermore, by the time this thesis was delivered, two other works have been submitted and are waiting for a decision. These works are related to the model extension (Chapter 5) and optimization problem (Chapter 6). Another likely submission is related to the related works (Chapter 3), which compares multiple analytical works related to the MEC-NFV virtual resource topic. Thus, we could extend this chapter and add other characteristics to the comparison in order to create a short survey on the topic.

7.3 FUTURE WORKS

This section is segmented into three groups of future works related to the content of this thesis. The first is related to improvements on the proposed model. The second is concerned with the other schemes to solve the multi-objective problem. The last group contains other research approaches towards the same problem (e.g., Testbed).

7.3.1 Related Mathematical Models

7.3.1.1 Reducing model computational complexity

In general, the computational cost to solve Markov chains with Ω states by a naive algorithm is $O(\Omega)^3$ making it difficult to be solved in a short time. One more challenge is to modify some steps towards solving the proposed analytical model linear system, using reasonable approximations that should be lightweight in terms of computational cost.

7.3.1.2 Edge and Central Cloud Representation

The present thesis accounted for a single MEC node instance, which limits a holistic analysis. In order to widen the model's capacity, multiple MEC nodes and/or Central Clouds could be incorporated and other problems such as traffic scheduling could be tackled.

7.3.1.3 Multiple Service Classes

There is a perspectives of combining two or more service categories in 6G, e.g., combined Ultra mobile broadband (uMBB) and Ultra-high precision communication (uHPC), which should evolve from Enhanced Mobile Broadband (EMBB) and URLLC, respectively [Yeh and Jo 2022], addressesing their requirements simultaneously. Thus, another possible model could encompass the representation of multiple service categories.

7.3.1.4 Precision Models

Mathematical tools that characterize extreme events and precise bounds have been recently used for the design of low-latency and highly-reliable wireless networks. In this regard, extreme value theory, Meta distribution and network calculus are important methodologies. Instead of averages, these alternatives are interesting for allowing a more precise evaluation such as worst-case latency values (bound analysis). Hence, a new challenge is to compare the results obtained in this work to those produced by one or more of these alternatives.

7.3.2 Optimization Algorithms for Multi-Objective Problems

7.3.2.1 Bio-Inspired Approaches

We have restricted the solution of the multi objective problem to the adoption of GA. However, multiple bio-inspired approaches can be adopted and the comparison of their results could yield another work. We have recently adopted the Fuzzy approach together with GA in [Balieiro et al. 2021] towards the problem of opportunistic slicing.

7.3.3 Other Approaches Towards MEC-NFV Environment

7.3.3.1 Probability Distributions

We adopted Poisson and exponential distributions to express the arrival process and service times. However, other probability distributions such as Hyper-Exponential and Erlang General can be considered.

7.3.3.2 Testbed and Simulation

Small-scale network experiments are quite rare in the topic of MEC-NFV, probably due to a still emerging set of open source tools [Zhao et al. 2021]. Yet, we believe that those tools will become available in a near future, and hence, it would be possible to compare analytical to testbed or to other simulation results, specially those of preexisting frameworks known to the community such as OMNeT++, NS-3 and CloudSim.

REFERENCES

3GPP. System architecture for the 5g system (5gs). *White Paper*, 2020.

AGARWAL, S.; DUNAGAN, J.; JAIN, N.; SAROIU, S.; WOLMAN, A.; BHOGAN, H. Volley: Automated data placement for geo-distributed cloud services. *NSDI*, p. 17–32, 01 2010.

AL-SHUWAILI, A. Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling. *IEEE Transactions on Signal and Information Processing over Networks*, v. 3, n. 4, p. 787–802, 2017.

ALICHERRY, M. Network aware resource allocation in distributed clouds. *2012 Proceedings IEEE INFOCOM*, p. 963–971, 2012.

ANAND, A.; VECIANA, G. de. Resource allocation and harq optimization for urllc traffic in 5g wireless networks. *IEEE Journal on Selected Areas in Communications*, v. 36, n. 11, p. 2411–2421, 2018.

ARSHAD, Q. K. U. D.; KASHIF, A. U. A review on the evolution of cellular technologies. In: *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. [S.l.: s.n.], 2019. p. 989–993.

ASCIERTO, R. Uptime institute global data center survey. *Uptime Institute*, 2019.

BALIEIRO, A.; FALCAO, M.; SOUZA, C.; DIAS, K.; ALVES, E. A fuzzy-genetic approach for 5g/6g opportunistic slicing. *2021 IEEE Latin-American Conference on Communications (LATINCOM)*, p. 1–6, 2021.

BALIEIRO, A.; SOUZA, C.; FALCAO, M.; DIAS, K. Secondary virtual network mapping onto cognitive radio substrate: A collision probability analysis. *IEEE Communications Letters*, v. 21, n. 3, p. 600–603, 2017.

BALIEIRO, A.; SOUZA, C.; FALCAO, M.; DIAS, K. An evolutionary scheme for secondary virtual networks mapping onto cognitive radio substrate. *Wireless Communications and Mobile Computing*, v. 2019, p. 1–19, 03 2019.

BEKKOUCHE, O.; SAMDANIS, K.; BAGAA, M.; TALEB, T. A service-based architecture for enabling uav enhanced network services. *IEEE Network*, v. 34, n. 4, p. 328–335, 2020.

BELOGLAZOV, A. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, v. 28, p. 755 – 768, 05 2012.

BI, Y.; COLMAN-MEIXNER, C.; WANG, R.; MENG, F.; NEJABATI, R.; SIMEONIDOU, D. Resource allocation for ultra-low latency virtual network services in hierarchical 5g network. In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2019. p. 1–7.

BOLCH, G.; GREINER, S.; MEER, H.; TRIVEDI, K. Queueing networks and markov chains. *Wiley New York*, 2006.

BRUNEO, D. A stochastic model to investigate data center performance and qos in iaas cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, v. 25, n. 3, p. 560–569, 2014.

CHEN, M.-H. Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point. *INFOCOM*, p. 1–9, 05 2017.

COOPER, R. Introduction to queueing theory. *Elsevier North Holland*, v. 2, 1981.

COSTANZO, F.; LORENZO, L. Dynamic resource optimization and altitude selection in uav-based multi-access edge computing. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 4985–4989, 2019.

DEB, K. *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*. [S.l.]: Springer, New York, 2011.

DOAN, T. V.; NGUYEN, G. T.; SALAH, H.; PANDI, S.; JARSCHEL, M.; PRIES, R.; FITZEK, F. H. P. Containers vs virtual machines: Choosing the right virtualization technology for mobile edge cloud. In: *2019 IEEE 2nd 5G World Forum (5GWF)*. [S.l.: s.n.], 2019. p. 46–52.

EMARA, M.; ELSAWY, H.; FILIPPOU, M. C.; BAUCH, G. Spatiotemporal dependable task execution services in mec-enabled wireless systems. *IEEE Wireless Communications Letters*, v. 10, n. 2, p. 211–215, 2021.

FALCAO, M.; SILVA, G. A.; DIAS, K.; BALIEIRO, A. Three-layered prioritized cognitive radio networks with channel aggregation and fragmentation techniques. *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, p. 1–5, 2016.

FALCAO, M.; SOUZA, C.; BALIEIRO, A.; DIAS, K. Um modelo de largura de banda flexível para redes de rádios cognitivos baseadas em prioridade. *XXXV Simpósio Brasileiro de Redes e Sistemas Distribuídos*, 06 2017.

FALCAO, M.; SOUZA, C.; BALIEIRO, A.; DIAS, K. A flexible-bandwidth model with channel reservation and channel aggregation for three-layered cognitive radio networks. *Computer Networks*, v. 135, p. 213–225, 04 2018.

FALCAO, M.; SOUZA, C.; BALIEIRO, A.; DIAS, K. An analytical framework for urllc in hybrid mec environments. *The Journal of Supercomputing*, 06 2021.

FARHADI, V.; MEHMETI, F.; HE, T.; PORTA, T. L.; KHAMFROUSH, H.; WANG, S.; CHAN, K. S. Service placement and request scheduling for data-intensive applications in edge clouds. *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, p. 1279–1287, 2019.

FAUTREL, T.; GEORGE, L.; FAUBERTEAU, F.; GRANDPIERRE, T. An hypervisor approach for mixed critical real-time uav applications. *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, p. 985–991, 2019.

FENG, D.; SHE, C.; YING, K.; LAI, L.; HOU, Z.; QUEK, T. Q. S.; LI, Y.; VUCETIC, B. Toward ultrareliable low-latency communications: Typical scenarios, possible solutions, and open issues. *IEEE Vehicular Technology Magazine*, v. 14, n. 2, p. 94–102, 2019.

FILIPPOU, M. C.; SABELLA, D.; EMARA, M.; PRABHAKARAN, S.; SHI, Y.; BIAN, B.; RAO, A. Multi-access edge computing: A comparative analysis of 5g system deployments and service consumption locality variants. *IEEE Communications Standards Magazine*, v. 4, n. 2, p. 32–39, 2020.

GROSS, D.; SHORTLE, J.; THOMPSON, J.; HARRIS, C. Fundamentals of queueing theory. *John Wiley Sons, Inc*, v. 4, 2008.

GUNDALL, M.; STEGMANN, J.; HUBER, C.; SCHOTTEN, H. D. Towards organic 6g networks: Virtualization and live migration of core network functions. In: *Mobile Communication - Technologies and Applications; 25th ITG-Symposium*. [S.l.: s.n.], 2021. p. 1–6.

GUO, S. Resource modeling and scheduling for mobile edge computing: A service provider's perspective. *IEEE Access*, PP, p. 1–1, 06 2018.

HUANG, G. Auto scaling virtual machines for web applications with queueing theory. *2016 3rd International Conference on Systems and Informatics (ICSAI)*, p. 433–438, 2016.

ISLAMBOULI, R.; SHARAFEDDINE, S. Optimized 3d deployment of uav-mounted cloudlets to support latency-sensitive services in iot networks. *IEEE Access*, v. 7, p. 172860–172870, 2019.

JAIN, R. The art of computer systems performance analysis: Techniques for experimental design, measurement. *Wiley New York*, 1991.

JI, H.; PARK, S.; YEO, J.; KIM, Y.; LEE, J.; SHIM, B. Ultra-reliable and low-latency communications in 5g downlink: Physical layer aspects. *IEEE Wireless Communications*, v. 25, n. 3, p. 124–130, 2018.

KATOCH, S. A review on genetic algorithm: past, present, and future. *Multimed Tools Applications*, v. 184, p. 205–222, 2020.

KAUR, K.; DHAND, T.; KUMAR, N.; ZEADALLY, S. Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. *IEEE Wireless Communications*, v. 24, n. 3, p. 48–56, 2017.

KAYA, M. The effects of a new selection operator on the performance of a genetic algorithm. *Applied Mathematics and Computation*, v. 217, p. 7669–7678, 06 2011.

KEKKI, S.; FEATHERSTONE, W. Mec in 5g networks. *ETSI White Paper*, n. 28, p. 1–28, 2018.

KEMENY, J. Finite markov chains. *van Nostrand Company*, 1960.

KHAN, A. Key characteristics of a container orchestration platform to enable a modern application. *IEEE Cloud Computing*, v. 4, n. 5, p. 42–48, 2017.

KHERRAF, N.; ALAMEDDINE, H. A.; SHARAFEDDINE, S.; ASSI, C. M.; GHRAYEB, A. Optimized provisioning of edge computing resources with heterogeneous workload in iot networks. *IEEE Transactions on Network and Service Management*, v. 16, n. 2, p. 459–474, 2019.

KLEINROCK, L. Queueing systems. *John Wiley and Sons, New York*, v. 1, 1975.

LAL, S.; RAVIDAS, S.; OLIVER, I.; TALEB, T. Assuring virtual network function image integrity and host sealing in telco cloud. *2017 IEEE International Conference on Communications (ICC)*, p. 1–6, 2017.

LEE, S. Low cost mec server placement and association in 5g networks. *International Conference on Information and Communication Technology Convergence (ICTC)*, p. 879–882, 2019.

LI, C.; CAI, Q.; ZHANG, C.; MA, B.; LUO, Y. Computation offloading and service allocation in mobile edge computing. *The Journal of Supercomputing*, v. 77, p. 1–30, 12 2021.

LI, X.; YAO, H.; WANG, J.; XU, X.; JIANG, C.; HANZO, L. A near-optimal uav-aided radio coverage strategy for dense urban areas. *IEEE Transactions on Vehicular Technology*, v. 68, n. 9, p. 9098–9109, 2019.

LI, Z.; KIHL, M.; LU, Q.; ANDERSSON, J. A. Performance overhead comparison between hypervisor and container based virtualization. In: *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. [S.l.: s.n.], 2017. p. 955–962.

MA, S.; CHEN, X.; LI, Z.; CHEN, Y. Performance evaluation of urllc in 5g based on stochastic network calculus. *Mobile Networks and Applications*, v. 26, 06 2021.

MAVRIDIS, I.; PANTELIS, H. Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. *Future Generation Computer Systems*, p. 674–696, 2019.

MCCALL, J. Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, v. 184, p. 205–222, 2005.

MIJUMBI, R.; SERRAT, J.; GORRICHO, J.-L.; BOUTEN, N.; TURCK, F. D.; BOUTABA, R. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 236–262, 2016.

MORABITO, R. Power consumption of virtualization technologies: An empirical investigation. *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, p. 522–527, 2015.

NORRIS, J. Markov chains. *Cambridge University Press*, 1998.

PALAKONDA, V.; MALLIPEDDI, R. Pareto dominance-based algorithms with ranking methods for many-objective optimization. *IEEE Access*, v. 5, p. 11043–11053, 2017.

POCOVI, G.; SHARIATMADARI, H.; BERARDINELLI, G.; PEDERSEN, K.; STEINER, J.; LI, Z. Achieving ultra-reliable low-latency communications: Challenges and envisioned system enhancements. *IEEE Network*, v. 32, n. 2, p. 8–15, 2018.

REN, Y.; PHUNG-DUC, T.; CHEN, J.-C.; YU, Z.-W. Dynamic auto scaling algorithm (dasa) for 5g mobile networks. *2016 IEEE Global Communications Conference (GLOBECOM)*, p. 1–6, 2016.

ROCHWERGER, B. Reservoir: Management technologies and requirements for next generation service oriented infrastructures. *2009 IFIP/IEEE International Symposium on Integrated Network Management*, p. 307–310, 2009.

SAMANTA, A.; TANG, J. Dyme: Dynamic microservice scheduling in edge computing enabled iot. *IEEE Internet of Things Journal*, v. 7, n. 7, p. 6164–6174, 2020.

SANTOYO-GONZALEZ, A.; CERVELLO-PASTOR, C. Edge nodes infrastructure placement parameters for 5g networks. *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, p. 1–6, 2018.

SARRIGIANNIS, I.; RAMANTAS, K.; KARTSAKLI, E.; MEKIKIS, P.-V.; ANTONOPOULOS, A.; VERIKOUKIS, C. Online vnf lifecycle management in an mec-enabled 5g iot architecture. *IEEE Internet of Things Journal*, v. 7, n. 5, p. 4183–4194, 2020.

SILVA, D.; ALVES, G.; NETO, P. D. M.; FERREIRA, T. Measurement of fitness function efficiency using data envelopment analysis. *Expert Systems with Applications*, v. 41, p. 7147–7160, 11 2014.

SKARLAT, O. Towards qos-aware fog service placement. *ICFEC*, 05 2017.

SOUZA, C.; FALCAO, M.; BALIEIRO, A.; DIAS, K. Modelling and analysis of 5g networks based on mec-nfv for urllc services. *IEEE Latin America Transactions*, v. 19, n. 10, p. 1745–1753, 2021.

SYSWERDA, G. Uniform crossover in genetic algorithms. *Proc. 3rd Intl Conference on Genetic Algorithms 1989*, 01 1989.

TAKO, A.; ROBINSON, S. Comparing discrete-event simulation and system dynamics: Users' perceptions. *JORS*, v. 60, p. 296–312, 03 2009.

TAN, H.; HAN, Z.; LI, X.-Y.; LAU, F. Online job dispatching and scheduling in edge-clouds. p. 1–9, 05 2017.

TONG, L. A hierarchical edge cloud architecture for mobile computing. *INFOCOM*, p. 1–9, 04 2016.

TONG, Z.; ZHANG, T.; ZHU, Y.; HUANG, R. Communication and computation resource allocation for end-to-end slicing in mobile networks. *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, p. 1286–1291, 2020.

YALA, L.; FRANGOUDIS, P. Latency and availability driven vnf placement in a mec-nfv environment. *2018 IEEE Global Communications Conference (GLOBECOM)*, p. 1–7, 2018.

YANG, Z.; PAN, C.; WANG, K.; SHIKH-BAHAEI, M. Energy efficient resource allocation in uav-enabled mobile edge computing networks. *IEEE Transactions on Wireless Communications*, v. 18, n. 9, p. 4576–4589, 2019.

YEH, C.; JO, G. Perspectives on 6g wireless communications. *ICT Express*, p. 1–5, 2022.

ZHANG, J.; ZHOU, L.; TANG, Q.; NGAI, E. C.-H.; HU, X.; ZHAO, H.; WEI, J. Stochastic computation offloading and trajectory scheduling for uav-assisted mobile edge computing. *IEEE Internet of Things Journal*, v. 6, n. 2, p. 3688–3699, 2019.

ZHAO, L.; ZHOU, G.; ZHENG, G.; I., C.-L.; YOU, X.; HANZO, L. Open-source multi-access edge computing for 6g: Opportunities and challenges. *IEEE Access*, PP, 11 2021.