Availability-Aware Resource Allocation for Containerized Network Functions

by

Zhuonan Huang

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Deploying virtual network functions (VNFs) such as WAN accelerators, network address translators (NATs) and 5G functions at the network edge (NE) can significantly reduce the experienced latency of delay-ultrasensitive applications (e.g., autonomous vehicles and Internet of things). Nonetheless, a major challenge to their anticipated large-scale deployment is the ability to efficiently allocate and manage the scarce NE resources hosting these functions. In this thesis, we describe a novel containerized infrastructure manager (cIM) that extends current managers, such as Kubernetes, with the necessary building blocks to provide an accurate yet elastic resource allocation service to containerized VNFs at scale. The proposed cIM treats the main modules of the VNFs, i.e., the containerized VNF components (cNFCs), as atomic special-purpose functions that can be rapidly deployed to form complex network services. The main component of the proposed cIM, the resource reservation manager (RRM), employs concepts of risk pooling in the insurance industry to accurately reserve the needed resources for the hosting containers. More precisely, to meet anticipated cNFCs demand fluctuation, the RRM accurately reserves a quota of additional resources that are shared by the containerized functions collected together in clusters. The reserved quota of resources ensures the desired availability level of the cNFCs without overprovisioning the scarce resources of the NE. The RRM considers three different situations namely that of a cNFC instance, a cluster of cNFCs or multiple cNFC clusters sharing the reserved resources. Different allocation approaches are then presented for each of these three situations. Simulation experiments are conducted to evaluate the performance of our reservation schemes from different aspects. The corresponding experimental results demonstrate that our proposed cIM can significantly improve the performance of the cNFCs and guarantee their desired availability with minimal resource reservation. Optimal allocation solutions of the resource pools are further proposed considering the desired availability level and the limit of resource pools. The evaluation results demonstrate that our optimization models and solutions obtain the best performance of relevant testing parameters, e.g., availability.

Acknowledgements

I wish to thank all the people whose assistance was a milestone in the completion of this project.

First of all, I wish to show my gratitude to my parents for the support and great love. They kept me going on.

Next, I wish to express my sincere appreciation to my supervisors, Professor Ahmed Karmouch and Professor Nancy Samaan, who convincingly guided and encouraged me to be professional and do the right thing even when the road got tough. Without their persistent help, the goal of this project would not have been realized.

Finally, I would also like to show my gratefulness to my friends. This work would not have been possible without their encouragement.

Table of Contents

List of Figures

Nomenclature

Abbreviations

API Application Programming Interface

BRC Basic Resource Consumption

CAPEX/OPEX Capital Expenditures And Operating Expenses

cIM containerized Infrastructure Manager

CPU Central Processing Unit

CSM Container Service Manager

DNSs Domain Name Systems

DRC Duty Resource Consumption

ETSI European Telecommunications Standards Institute

IDS Intrusion Detection System

IMM Infrastructure Monitoring Manager

IoT Internet of Thing

JPV Joint Path-VNF

MANO Management and Orchestration

MCC Mobile Cloud Computing

MDP Markov Decision Process

NATs Network Address Translators

NFV Network Functions Virtualization

NFVI NFV Infrastructure

NFVM NFV Manager

NFVO NFV Orchestrator

NSs Network Services

QoS Quality of Service

RAN Radio Access Network

RRM Resource Reservation Manager

SDN Software-Defined Network

SLA Service Level Agreement

VIM Virtualized Infrastructure Manager

VM Virtual Machine

VMs Virtual Machines

VNFs Virtual Network Functions

Chapter 1

Introduction

Network function virtualization (NFV) is a recently developed technology that breaks from the limitations of proprietary and dedicated hardware-based network operations. With NFV, network functions such as domain name systems (DNSs), network address translators (NATs), load balancers and firewalls are deployed as software components running on virtual machines (VMs) or containers hosted on commodity servers. In this manner, a typical network service can be described as a set of service function chains (SFCs), where each SFC is comprised of an ordered set of virtual network functions (VNFs). In turn, every VNF can be regarded as a virtualized network service moving traditional middlebox to common virtualization infrastructure software, programmed to implement a specific network function. With VNFs, the scalability and portability of the network can be improved, and better usage of network infrastructure resources can be guaranteed. Since VNFs are deployed on commodity virtualization software instead of physical hardware, power consumption can be reduced, which results in the increment of the availability of physical space. This paradigm shift in network operations has been gradually embraced by both academia and industry [5].

With the promise of capital expenditures and operating expenses (CAPEX/OPEX) savings, network operators have invested in developing a novel, European Telecommunications Standards Institute (ETSI) standardized VNF framework [6]. The architecture is composed of two layers, a physical infrastructure, of compute, storage and network

resources, that is hosting the second layer of VNFs using different means of visualization such as virtual machines (VMs) or containers. The management and orchestration (MANO) entity in the ETSI architecture maintains catalogues of offered network services (NSs), their SFC compositions and the available flavours of the VNFs. To manage the two layers, MANO is comprised of three main components: the NFV orchestrator (NFVO), the VNF manager (VNFM), and the virtual infrastructure manager (VIM). These components interact together and with the physical and VNF layers to manage the deployment and lifecycle of the offered network services, their SFCs and the physical and virtual resources.

The role of VNFs becomes more prominent at the network edge (NE) [7,8] in proximity to end-users. Placing these functions at the NE can significantly enhance the performance of low-latency applications such as those using virtual reality, video streaming and IoT applications. Moreover, it results in a significant reduction of traffic congestion in the core network. Recently, the deployment of VNFs tends to be completed on containers that are packaged as computing environments isolated from the rest of the system based on another virtualization technology. Compared to VMs, containers are more lightweight under the shared operating system (OS). Thus they can be moved across multiple environments with high efficiency.

1.1 Problem Statement

One of the main challenges in deploying VNFs at the NE is the limited availability of resources when compared to cloud-based deployments. The problem is further complicated by the anticipated large number of VNFs that must be deployed as the number of served users and applications increases with the wide deployments of 5G networks [9]. To overcome some of these limitations, many studies [9–11] have analyzed the main characteristics of the VNFs that are expected to be widely deployed at the NE. The majority of these approaches agree that edge VNFs are unique in that they can be regarded as reusable and specialized components with micro-operations [11] that can be instantiated rapidly to form more complex service function chains (SFCs) [10].

Existing research efforts have addressed several critical issues for VNF management including placement, resource allocation (RA), scaling and migration [12]. Nonetheless, the majority of these efforts model the VNF resource demands as fixed scalar values that do not account for possible workload fluctuations. On the contrary, a key ingredient to achieving VNF reliability at runtime is to ensure the availability of sufficient VNFI resources to meet varying demands [13] while ensuring the efficient management of the scarce resources at the NE.

However, industrial research mostly focuses on on-demand resource allocation rather than availability-aware allocation. To be more specific, the existing resource allocation schemes can be divided into two categories as proactive approaches which involve the action of predicting workload, and reactive approaches, defining how to react to the previous workload change. Although these schemes can allocate resources to VNFs, both of them mainly take the current resource demand of VNFs into account, neglecting the potential VNF crash or failure when workflows increase promptly. Thus, the resources allocated to each VNF are usually not enough to deal with the workload fluctuations. On the other hand, it is costly to predict the resource demand in proactive approaches as the algorithm calculation process consumes a large part of the resources. Moreover, the accuracy of proactive allocation is also hard to be guaranteed since the workload changes frequently. For those reactive ways, the latency of allocation is always difficult to solve as the algorithm takes some time to run but meanwhile the workload changes rapidly.

Considering the aforementioned problems in resource allocation, we develop a novel resource reservation scheme where a small amount of reserved resources can deal with workload fluctuations without latency, increasing the availability of VNFs. Nevertheless, an upcoming challenge is to take other factors that might influence the performance of reservation into account. For example, the limit of resource pools as the reservation will occupy a small part of server resources, the loss function which considers the difference between actual resource demand and allocated resources while doing the reservation. On the other hand, the profits of cloud service providers adopting the reservation scheme should also be concerned in order to facilitate and popularize the solution in practice. The

price of resources should be set appropriately as a tool to adjust user traffic and meanwhile maximize the service provides' interest.

1.2 Contribution

In this thesis, we focus on the functionality of the VIM within the ETSI VNF architecture as the main entity responsible for ensuring sufficient resources to the hosted VNFs at the NE server. We focus on containerized VNF components (cNFCs) as the main atomic units of the hosted VNFs. Similar to recent research efforts [14], we achieve resource usage efficiency by clustering together cNFCs with similar characteristics. It is worth noting that, the main motivation for VNF clustering in existing approaches was to reduce the redundancy in the resources used for VNF backup in case of main VNF failures. Contrary to these approaches, we cluster cNFCs with similar expected resource demand characteristics with the aim of reserving a shared minimal resource buffer that can be used by the hosted cNFCs during high workload fluctuations. More precisely, the main component of our proposed containerized Infrastructure manager (cIM), the resource reservation module (RRM) employs concepts of risk pooling in the insurance industry [15,16] to accurately reserve the needed resources for the hosting containers while ensuring that the scarce resources are not over-provisioned [17]. The RRM also solves some relevant optimization problems to achieve better performance, ensuring the profits of cloud service providers. More precisely, the main contributions of this thesis can be summarized as follow.

- We describe a novel containerized infrastructure manager that extends the ETSI VNF architecture to provide precise resource allocation for VNFs workload fluctuations.
- We propose novel mathematical models of resource reservation to allocate sufficient resources for the containers hosting the cNFCs. We focus on the allocation of the CPU cores, as the most constrained resources on the NE, but the presented work can be extended to other resources. Three different circumstances where a cNFC instance, a cNFC cluster or multiple cNFC clusters compose the resource pool are

taken into account. We provide the general solutions for accurate and sufficient resource allocation of cNFCs in these corresponding circumstances.

- To meet various service level requirements, we then discuss possible optimization problems in the reservation scheme. Based on different Quality of Service (QoS) requirements, we propose corresponding optimization solutions which can be combined with the basic allocation schemes. To be more specific, four optimization problems are raised in our research: the optimization problem of resource pool calculation given the specific availability requirement, the optimization problem of resource pool calculation given the expected fluctuation, the optimization problem of resource allocation considering the limit of available resources and penalties, a pricing model to maximize the CSP expected long-term revenue.

1.2.1 Publication

Zhuonan Huang and Nancy Samaan and Ahmed Karmouch. A Novel Resource Reliability-Aware Infrastructure Manager for Containerized Network Functions. In *2021 IEEE International Conference on Communications (ICC): Communication QoS, Reliability and Modeling Symposium (IEEE ICC'21 - CQRM Symposium).*

1.3 Thesis Outline

This reminder of the thesis is organized as follows.

- Chapter 2 gives a background of existing VIM architectures and discusses the related research efforts dedicated to resource reservation, VNF scaling and VNF backup and resource reservation redundancy. Some common models are also presented.

- Chapter 3 describes the architecture of our proposed cIM within the context of the ETSI framework.

- Chapter 4 describes our novel mathematical model pertaining to efficient resource reservation for cNFC clusters. The optimization problems considering different QoS requirements and corresponding solutions are also presented in this chapter.

- Chapter 5 discusses the performance evaluation results of all the aforementioned schemes considering different performance parameters. Specifically, the simulation experiments demonstrate the efficiency and accuracy of our proposed reservation schemes and corresponding optimization solutions.

- Chapter 6 gives the conclusion of this thesis and discusses possible future research directions.

Chapter 2

Background and Literature Review

The work proposed in this thesis focuses on the functionality of the virtual infrastructure manager (VIM) and NFV manager (NFVM). VIM is responsible for allocating and managing the physical resources in the VNFI [6]. NFVM fulfills the Management and Orchestration of VNF instances through the lifecycle, performance and fault management. In the following sections, the background related to the transition of current cloud computing and NFV technologies is first presented. Then we provide an overview of existing VIM and NFVM architectures and then discuss related research efforts dedicated to resource reservation, VNF scaling and VNF backup and resource reservation redundancy.

2.1 Background

2.1.1 From Centralized Clouds to Network Edges

With the emergence of Internet of Things (IoT), 5G technology and numerous intelligent mobile devices, data volumes are increasing dramatically at the Network Edge (NE). Traditional centralized clouds now are not adequate enough to process such astounding quantities of data, thus the shift of working mode from centralization to distribution has been commonly embraced by the industry (e.g., OpenStack++ cloudlets running open-source derivatives of the widely used OpenStack cloud computing platform [18]). Basically, net-

work edge, which refers to the end-points and the first hop (the closest connections) from the end-points into the core network, moves computation and storage resources closer to the terminal devices, instead of leaning on the remote center location with scarce resources and high response latency. Mobile Edge Computing (MEC) provides cloud computing, storage, and networking capabilities to mobile subscribers at the network edge. Besides, MEC holds great promise to open new frontiers for applications [18], network operators and service providers, supporting them to efficiently deploy various NSs in the area of NFV. Introducing VNFs at the NE, in close proximity of the end-users, is a feasible solution to reduce transmission latency and unnecessary workload of the core network while providing flexibility and efficiency for resource allocation [19].

2.1.2 From Virtual Machines to Containers

Traditionally, cloud service providers use virtual machines (VMs) to initiate multifunctional applications. Each virtual machine has its own guest operating system (OS), separating from other VMs. It uses the virtual resources provided by Hypervisor or host OS to support the upper-layer applications. Hypervisor guarantees the isolation between guest OSs, thus each guest OS works as if it is running directly on physical computer and it is usually isolated from other guest OSs that run on the same compute node.

Instead of using physical compute node, virtual machine (VM) relies on virtual software to deploy and run applications. Most virtual machines are powered by Hypervisors which is a software that provides isolation for virtual machines running on top of physical hosts. It is responsible for running different kernels on top of the physical host. This in turn makes the application and process isolation very expensive. Thus when scaling out new VNF replicas, the average startup time of each VM is costly, which may further result in the deterioration of VNF performance. In general, there are several problems with VM-based virtualization. Firstly, the provisioning time of resources is relatively long. The bulkiness of virtual disk image usually results in the long setup time of the virtual machine, taking up to approximately a minute. Secondly, a more important problem is that system resources are utilized inefficiently and a large part of available resources may even be wasted. This

is due to the fine-grained management of each resource that's theoretically accessible to the operating system. For example, when a guest OS detects that 1 CPU is accessible to it, it will take control of the whole CPU even though its upper-layer applications only uses a small part of the CPU.

However, a container holds a different approach than a Hypervisor. It can substitute for Hypervisor-based virtualization, using only one kernel for multiple isolated operating systems. In other words, a software container can be regarded as an isolated user-space instance of an operating system. This property makes the container lightweight, achieving high performance through the efficient usage of the single host kernel base on process isolation. Accordingly, container-based infrastructures can offer easier deployment and better performance. Containers can also eliminate the distinction between IaaS and non-virtualized servers (a.k.a., bare metal) since they offer the control and isolation of VMs with the performance of bare metal. Rather than maintaining different images for virtualized and non-virtualized servers, the same container image could be efficiently deployed on anything from a fraction of a core to an entire machine.

On the other hand, when doing containerization, we only build a virtualized operating system for our applications, instead of virtualizing the hardware to a VM. Thus multiple applications can run at the same time, and each one can obtain efficient performance, oblivious of the other containers it runs alongside. It's worth nothing that various limitations can be imposed on these apps' resource utilization in order to make full use of every available resource. In summary, containerization technology can achieve a set of goals including fast deployment, elasticity, scalability and high performance. Some of the most practical container implementation projects are FreeBSD Jails, Linux Containers (LXC), OpenVZ, rkt and Docker.

Docker containers

A docker container is the most popular container type by far [20]. The container usually involves the automatic creation of prepackaged software images which are stored in a local or remote repository. Each one of these images represents a complete file system

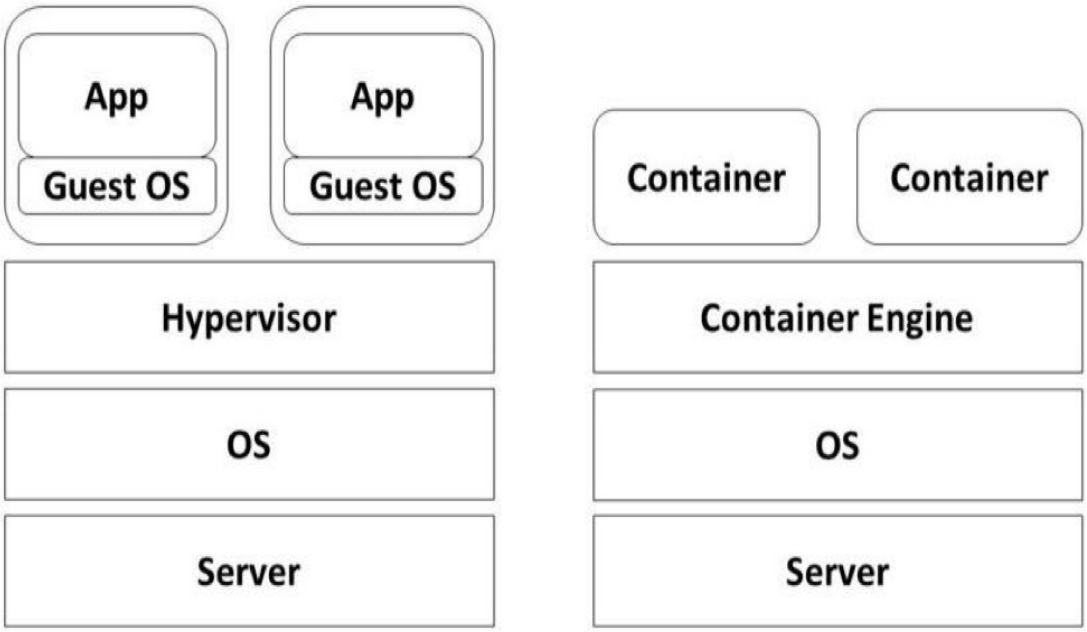


Figure 2.1: Virtual machines versus containers [1]

that contains everything that is needed to run one or more software applications. An end-user can deploy multiple applications into multiple containers simultaneously, greatly reducing the times and costs for installing and maintaining the software. Besides, each container image defines a fully reproducible environment, making the container portable and compatible. In this case, the container is capable to be moved to other systems, and even to be scaled into multiple instances of itself, or to be reset if the necessity of doing this arises. As shown in Figure 2.2, Docker provides registry infrastructures to store, publish and easily retrieve already made images from official or third-party sources; this allows everybody to run, modify and suit them to their needs, greatly simplifying how software is deployed and developed. A Docker container is an instance of a Docker image, managed by the Docker Engine. The container can be seen as an extremely small virtual machine, running on top of the same Linux kernel. Based on the aforementioned functionalities of Docker containers, the NFV layer can interact with Docker, creating and orchestrating VNFs contained inside containers. To be more specific, each VNF can be regarded as an image that is defined by a Dockerfile. The image is self-contained, providing all of the necessary software and scripts to accomplish the VNF's primary task.

As for the configuration variables and dependencies of this VNF, exporting them to the Docker container as environment variables will be done to instantiate the VNF and further compose the corresponding service function chain.

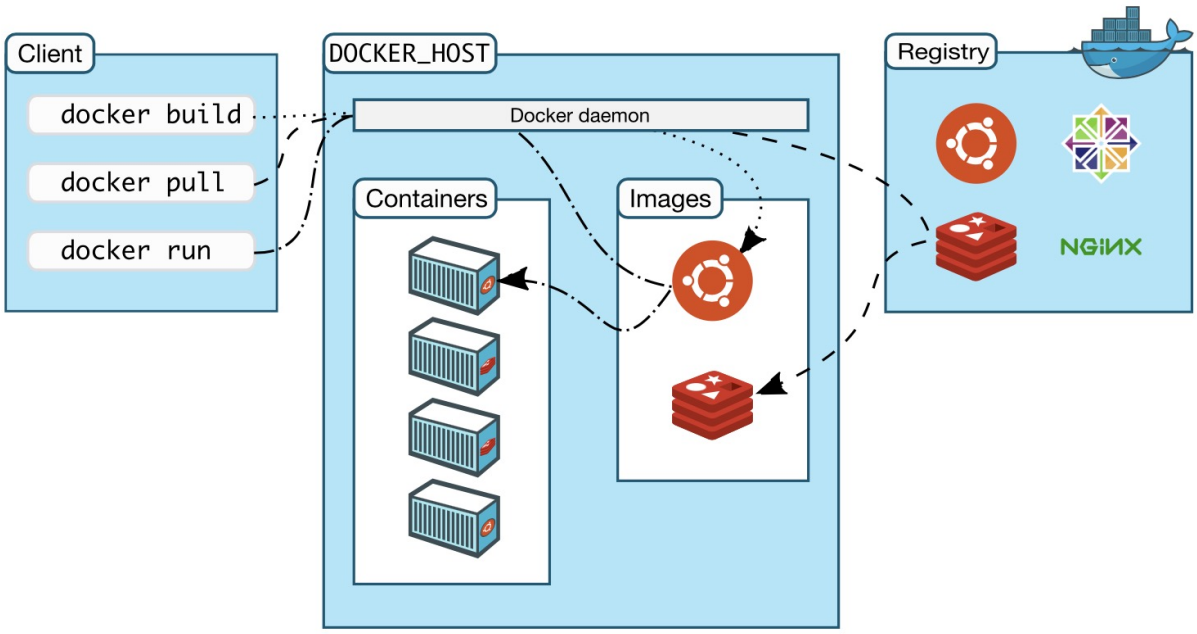


Figure 2.2: Docker architecture [2]

Kubernetes

Kubernetes is a container orchestration platform that provides automatic deployment and systematic management of containerized applications. It groups containers according to user demands and supports inter-container network communication. While Docker is a standalone software, enabling us to create and run container nodes on a OS, Kubernetes can allow us to automate container provisioning, networking, load-balancing, security and scaling across all these nodes [3]. Kubernetes offers DNS records to manage the networking of all container nodes, making sure that these container nodes are connected and interact in a secure way. A group of nodes managed by the same Kubernetes system can be regarded as a Kubernetes cluster. In other words, Kubernetes is a container orchestration and management platform.

As shown in Figure 2.3, a node is a basic unit that Kubernetes manages. It contains multiple pods and corresponding management services. More specifically, each node has a module namely kubelet to monitor and forward the status information of the node to the master node (the control plane of Kubernetes). Kubelet is also the primary node agent that helps in executing instructions given by the master. On the other side, Kube-Proxy in each node allows heterogeneous communication-related micro-services of the applications within the cluster. Each pod can communicate to other pods via this proxy. As for pod, it is the atomic deployment unit of Kubernetes. A pod contains one or more Docker containers that coexist to realize and provide a specific network function. Generally, a pod is comprised of a single container for the sake of simplicity. As for Kubernetes master, it consists of several functional blocks such as etcd, providing the controlling, scheduling, and networking of pods across multiple Kubernetes nodes. It guarantees the desired state of Kubernetes cluster.

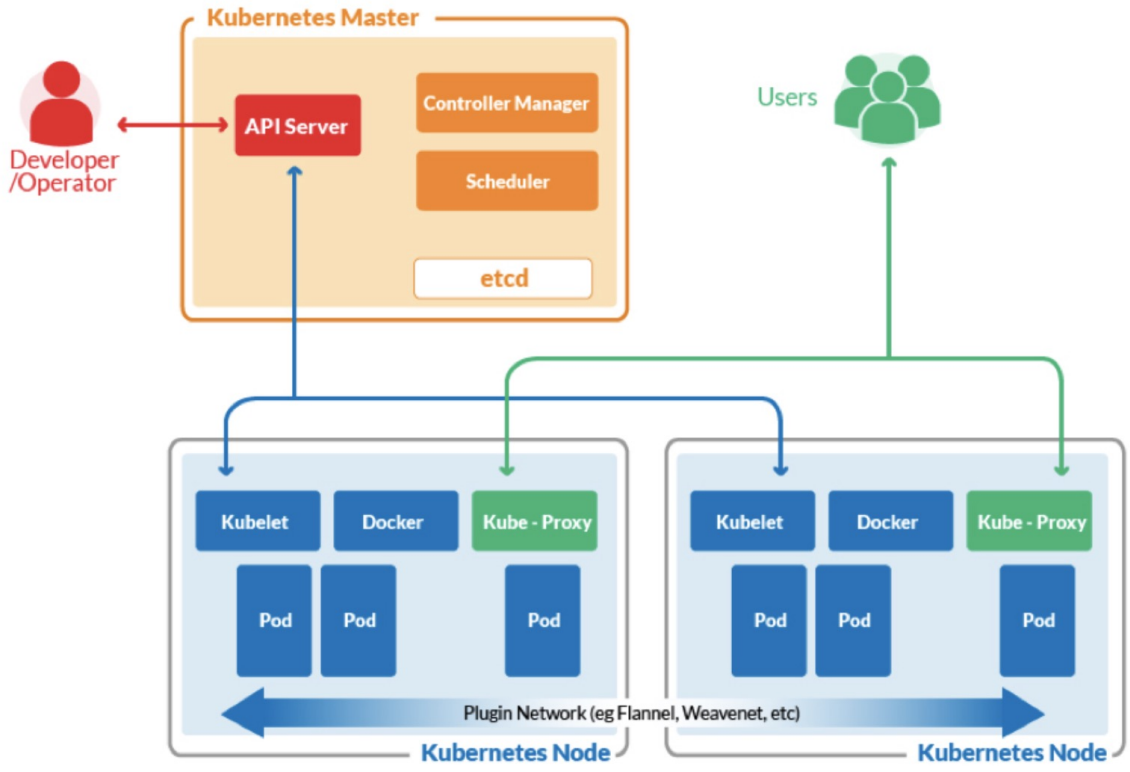


Figure 2.3: The architecture of Kubernetes with Docker [3]

In general, while Docker provides us with an efficient way to instantiate VNFs, Kubernetes allows us to manage the provisioning, networking, chaining, scaling and monitoring of these containerized VNFs. Consequently, Kubernetes and Docker can be highly exploited for NFV research, conforming to the basic NFV MANO standard.

2.2 VNFs

VNFs are the atomic function units in NFV architecture. Each VNF instance can be regarded as a virtualized network service that runs on a commodity compute node, replacing proprietary hardware. Through virtualization technology, these VNFs are compatible with various operating systems and can run consistently on different virtualization environments. The diversity of VNFs can also be guaranteed without huge physical resource consumption as they are mostly deployed in VM software and can be powered by various software images. Typical VNFs such as firewalls, IDSs and NAT services are widely embraced by the industry. Moreover, these individual VNFs can be standalone or connected together to form various ordered service function chains. Although the concept of service chaining is not new, the process of chaining is simplified and improved through VNF technology [21]. For example, the problem of long-distance communication can be eliminated, there is no need to consider the physical location of these network functions as they are implemented in software and rely on virtual resources. On the other hand, VNFs improve the scalability and elasticity of network services, offering agile scaling operations of NFV infrastructure resources. Since VNFs can be substituted for dedicated hardware, the goal of reducing power consumption is realized, increasing the available physical resource space. This also results in CAPEX/OPEX savings.

2.3 Existing NFV-MANO architectures

NFV management and network orchestration (MANO) module plays the management role in NFV. It contains several building blocks, each of them has a specific set of functionalities

and responsibilities applying management and orchestral operations on the corresponding entities. Different building blocks can interact with each other, leveraging monitoring information or services provided by other blocks. As shown in Figure 2.4, the MANO architecture consists of three core building blocks:

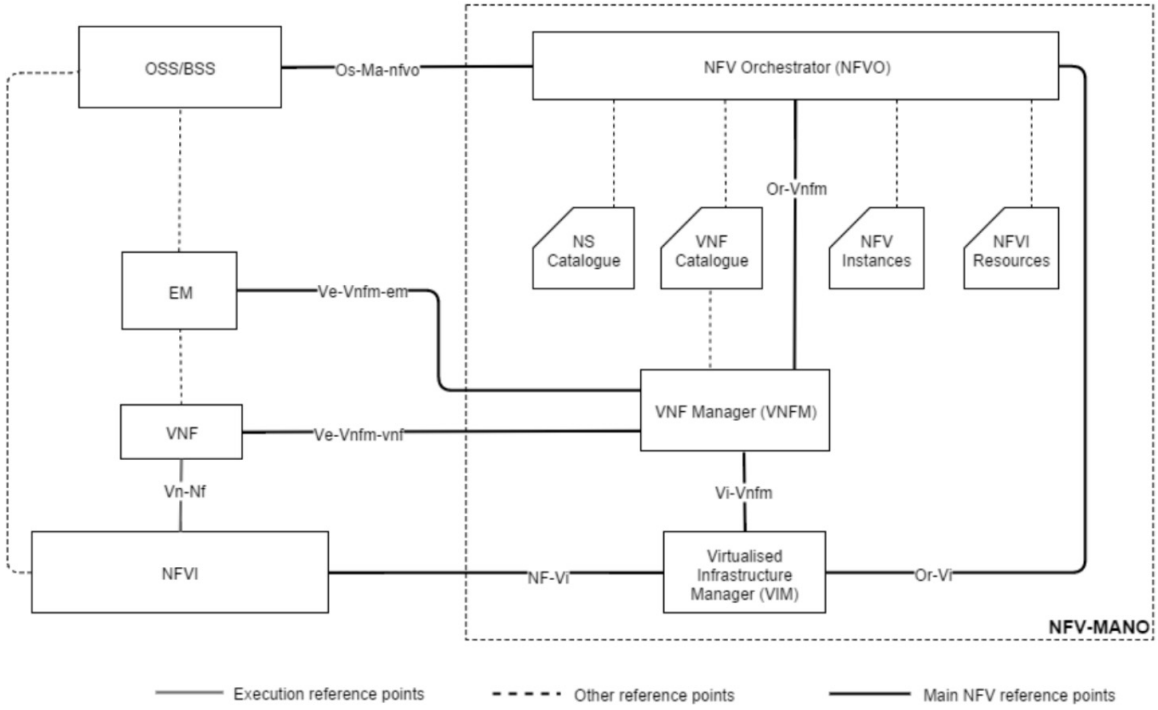


Figure 2.4: NFV-MANO architecture [4]

- *NFV Orchestrator (NFVO)*: NFVO is the component managing the orchestration of NFV infrastructure (NFVI) resources across multiple VIMs and the lifecycle of Network Services (NSs). In other words, NFVO is mainly responsible for resource orchestration and network service orchestration.

- *NFV Manager (NFVM)*: The NFVM provides the management and orchestration of performance, lifecycle and fault of different VNF instances. Each NFV manager may handle several instances of the same or different types, however, a single VNF instance is uniquely managed by a specific NFVM in turn.

- *Virtualized Infrastructure Manager (VIM)*: The VIM is the functional block that is

in charge of the control and management of the physical resources in NFVI, including the compute, memory and network resources such as bandwidth.

2.3.1 Existing VIM architectures

VIM as a main functional block of the MANO framework is in charge of the managing and monitoring work of the NFVI resources. A VIM can specialize in handling a certain type of NFVI resource which may be compute-only, storage-only or networking-only, however, it can also be capable of managing heterogeneous types of NFVI resources simultaneously while exposing a northbound interface to other functional blocks for communication. VIM manages an inventory that contains the information of the NFVI resources. With the inventory, VIM can orchestrate the allocation, release and reclamation of NFVI resources, managing their association with corresponding physical compute, memory and network resources such as images [22]. Also because VIM organizes virtual resources such as virtual links and ports, it provides basic support in managing VNF forwarding graphs.

Currently, there is a large number of VIM implementations that are provided through a number of projects. Examples of complete MANO or VIM implementations include: OpenStack [23], OpenBaton [24], Nomad [25] and Open VIM [26]. The first is a cloud infrastructure manager with SDN functionalities that can be tailored to VNF management. Nomad is a specialized micro-services-based architecture that can also be used for VNF deployments. Open VIM emerged to implement the ETSI MANO model. Finally, Kubernetes [27] is an open-source container orchestration platform that has been used to manage containerized VNF. It provides basic mechanisms for deployment, monitoring, and scaling of VNFs.

The majority of existing research efforts relevant to VIMs focus on evaluating the performance of the aforementioned existing implementations. For example, the authors in [7] provided a performance comparison between two VIM deployment solutions, namely, Openstack with VM-based deployment and Docker using containerization for VNFs as VIMs. The authors show that the use of containers represents a much more efficient solution for NEs due to the lightweight deployments of containers. On the other hand, a

very limited number of existing approaches have targeted enhancements for the ETSI VIM model. For example, the work in [28] focuses on examining the advantage of creating VIMs on demand for sliced data center resources. Also, one notable relaxant approach is that in [11] which extends existing VIMs in order to enable the instantiation of small-scale, or micro, VNFs.

In general, with the exception of some recent VIM implementations using unikernels (e.g., [11]), existing VIM architectures provide the needed functionalities for cloud-based implementations of MANO, but perform poorly in a resource-constrained environment such as the NE.

2.3.2 Existing NFVM architectures

Similar to VIM, NFVM also has a number of unique functionalities and communicates with other MANO modules. These functionalities are mainly related to:

- Instantiating, configuring or terminating VNF, checking if the instantiated VNF is running correctly.
- Modifying a running VNF instance such as implementing the scaling out(in) or up(down) operations of VNF instances.
- Collecting NFVI performance data and essential events related to its VNF instances.
- Managing VNF instances through their lifecycle. Communicating with VIM and NFVO through configuration and event reporting.

As for the projects providing NFVM implementations, Alcatel-Lucent, Ericsson, HP ES CMS have multiple telecommunication products, which act as NFV manager [29]. The first platform comprises the CloudBand Management System and the CloudBand Node, which make the VNFM part and parcel of their overall solution [30]. Ericsson Manager enables the creation, orchestration and monitoring of services running on virtualized IT [31]. HP offers compatible management of VNFs, enabling them to efficiently run on various virtualization environments and hardware platforms [32].

2.4 Resource reservation and scaling

Resource reservation is an effective methodology to improve QoS. It guarantees the availability of resources of the specific VNFs at a specific time and has been incorporated into cloud computing recently [33]. While resource reservation plays an important role in resource management strategies, elastic scaling is also widely adopted in industry to support Service Level Agreement (SLA). Elastic scaling in NFV is the ability to automatically adjust the amount of computing, storage or networking resources in accordance with the dynamic traffic patterns of applications. It can be categorized as horizontal scaling, including scaling out/in operations (creating/destroying container replicas) and vertical scaling, which is comprised of scaling up/down operations (resizing the capability on containers). It's notable that resource reservation is complementary to elastic scaling. With resource reservation, the efficiency of scaling can be highly improved. We discuss the related research efforts on resource reservation and elastic scaling in the following sections.

2.4.1 Reservation process

The resource reservation can be divided into two sub-processes as VNF classification and resource allocation. To be more specific, VNF classification is based on workload characterization and profiling. Resource allocation is the process of allocating a specific amount of resources to specific VNFs. RA is usually static, but a lot of dynamic allocation schemes are proposed to meet the changing demand of VNFs and guarantee NFV efficiency.

Classification of VNFs

VNFs are diverse in workload patterns and resource sensitivity. The fact that different VNF instances have different demands, imposes the need for dedicated profiling schemes to manage co-located heterogeneous types of VNFs [34]. A lot of research has analyzed Google cluster to reveal different workload patterns and characteristics [35]. For example, Di et al. [36] computed statistics about resource utilization, then characterized Google workload at the application level and job level. Mishra et al. [37] presented a multi-level

mechanism of task and workload classification. They also profiled heterogeneous applications through workload classification, task scheduling and capacity presetting. Dezhabad et al. [34] proposed a novel scheme to categorize workloads based on their resource usage. The authors used a hierarchical clustering algorithm, getting three workload and resource demand profiles for applications designated as low, moderate and high.

In general, VNFs belonging to the same type share the same time-varying behaviour and have the same network workload demands [38]. For example, the same type of network functions that fulfill similar service tasks in the same working period have the same traffic fluctuation and resource demand. Consequently, all the VNFs having the same workload pattern and a similar degree of CPU-sensitivity are categorized into the same class and assigned the same availability level.

Resources allocation

The problem of VNF resource reservation has been addressed extensively in the Literature [12]. For example, Roy et al. [39] proposed a predictive approach to minimize the resource provisioning costs while guaranteeing the application QoS. Similarly, Rao et al. [40] proposed a resource allocation scheme that guarantees the applications QoS. Joint VNF placement and resource allocation has been considered by Zhang et al. [41]. The authors proposed a novel VNF chain placement solution that maximizes resource utilization and minimizes response latency. Similarly, Ghaznavi et al. [42] proposed an optimal placement solution of VNF instances according to demanded workload. Tang et al. [43] designed an efficient algorithm solving the provisioning of VNF instances and SFCs based on traffic forecasting and further enabled VNF instances to scale in/out.

The main limitation of VNF placement and resource allocation schemes, in general, is that first they assume that VNF resource demands can be modelled as a simple scalar value, which does not account for resource demand fluctuation. Secondly, they don't account for the additional resources that might be needed by these VNFs. Hence, in the case of a sudden increase in the demanded resources scaling operations can fail due to insufficient resource availability.

2.4.2 Scaling

A large body of research efforts has also focused on vertical and horizontal scaling. For example, Li et al. [44] analyzed the advantages and disadvantages of vertical scaling and demonstrated that some VNFs can not improve their performance through vertical scaling. On the other hand, Rankothge et al. [45] illustrated that startup time is costly when horizontally scaling new VNFs. Yu et al. [46] proposed a fine-grained hybrid scaling scheme for SFCs to achieve NFV scaling efficiency while trying to minimizing resource cost for each SFC. Similarly, in ElasticNFV [47], the resource consumption of the hosting VM is monitored in order to achieve a balance between vertical scaling and migration whenever there are conflicts in resource demands.

Our proposed work is complementary to the aforementioned approaches; using our reservation scheme, the need for horizontal scaling is eliminated by allowing clusters of VNFCs to share the same pool of resources. A temporary need of excess resources by one or more instances can be immediately met using that shared pool without the need of invoking the VIM. On the other hand, horizontal scaling schemes can work alongside our proposed scheme by adding new instances within the appropriate VNF clusters.

2.5 Resource availability and backup models

The availability problem of SFCs, in general, has been mostly addressed by providing additional backup resources to re-instantiate the failed VNFs. Wang et al. [48] compared different existing backup models and presented a Joint Path-VNF (JPV) backup scheme to jointly considers the resources needed for path and VNF backups. similarly, Fan et al. [49] considered the problem of optimal availability-aware SFC mapping given the probability of VNF failures. The authors then developed an algorithm to minimize resource the SFCs consumption while guaranteeing their availability. VNF resource consumption was divided into two main components in [50]: basic resource consumption (BRC) and duty resource consumption (DRC). The former is used to maintains the VNF image and libraries while the latter represents the resources used by the VNF as it serves the incoming flows. The

authors then employed multi-tenancy and shared redundancy-based backup of resources to achieve SFC availability.

Our proposed work shares with these approaches the key idea of reserving shared resources for a pool of VNFs. However, while these reserved shared resources are used for backup redundancy, we utilize the shared pool to meet the fluctuations in the resource demands of the VNFs. Moreover, our approach is unique in that it utilizes simple statistical measurements about the running VNFs such as the resource usage means, variance and covariance to create accurate resource pooling without any over-provisioning of the NE limited resources.

2.6 Insurance and risk theory

The field of risk and insurance can be well used for designing economical NFV resource allocation scheme considering the statistical and financial aspects. Regarding insurance pricing, a lot of studies have been conducted to approach it. Yow et al. [51] adpoted a shareholder and total firm profits maximizing model to determine the optimal pricing scheme and capital allocation for the insurer. He et al. [52] proposed a novel insurance pricing model for maximal profit, incorporating driving risk from the mobility perspective. Kliger et al. [53] used economic and probability arguments to develop an approach for pricing insurance contracts. Mao et al. [54] extended Kliger's approach [53], maximizing the expected profit of the insurers. Cummins [15] presented two core paradigms of the field of risk and insurance as the statistical model and financial model. In the journal, the insurance risk pooling scheme is proposed based on probability theory (the law of large numbers and central limit theorem) and actuarial science. Zaks et al. [16] applied the idea of risk pooling to the use case where a portfolio consists of heterogeneous risks. It also raised the dual optimization problems considering the trade-off between the risk level and the premium distance. In this thesis, we mainly exploit the concepts of risk pooling for resource reservation from the studies of Cummins and Zaks, combining them with the real NFV usage scenario.

2.6.1 Statistical analysis

The law of large numbers and the central limit theorem play important roles in risk theory regardless of homogeneity or heterogeneity.

The Law of Large Numbers [15]

- *Version 1*: Let a random variable Y be equal to the number of successes in N repetitions of an experiment with probability of success equal to p. (I.e., the probability distribution of Y is the binomial.) Define the relative frequency of success as Y/N. Then for every $\varepsilon > 0$,

$$\lim_{N\to\infty} Pr(|\frac{Y}{N} - p| < \varepsilon) = 1 \tag{2.1}$$

- *Version 2*: Let $X_1, ..., X_N$ be a random sample from a distribution with mean μ and variance σ^2. Let $\overline{X}$ denotes the sample mean, which is equal to $\sum_i X_i/N$. Then we can get:

$$\lim_{N\to\infty} Pr(|\overline{X} - \mu| < \varepsilon) = 1 \tag{2.2}$$

Based on the $Version1$ of the law of large numbers, the probability that the observed proportion of successes will approach the probability of success (p) is close to 1 as the number of trials approaches infinity. One typical example is that, when doing the experiment of coin tosses, the probability that the observed side will be head approaches 1/2 as the number of tosses (N) becomes close to infinity.

As for the second version of the law of large numbers, it demonstrates that with probability being close to 1, the sample mean will arbitrarily approach the true distributional mean as N approaches infinity. This can be easily derived from Chebyshev's inequality. It's worth nothing that the $Version2$ is frequently used in insurance industry, defining the risk of insurers.

The Central Limit Theorem [15]

Let $X_1, ..., X_N$ be a random sample from a probability distribution with mean μ and variance σ^2. Define the random variable Y, with probability distribution $F_N(Y)$:

$$Y = \frac{\sum_{i=1}^{N} X_i - N\mu}{\sigma\sqrt{N}} = \sqrt{N}(\frac{\overline{X} - \mu}{\sigma}) \tag{2.3}$$

Then, as N becomes close to infinity, the probability distribution $F_N(Y)$ approaches $N(y)$. It should be noted that $N(y)$ is the standard normal distribution.

With the second version of the law of large numbers and the central limit theorem, insurer's risk can be defined as follows,

$$\lim_{N\to\infty} Pr(|\overline{X} - \mu| < \varepsilon) = \lim_{N\to\infty} Pr(|\overline{X} - \mu| < k\frac{\sigma}{\sqrt{N}}) \geq \lim_{N\to\infty}(1 - \frac{\sigma^2}{\varepsilon^2 N})) = 1 \tag{2.4}$$

X_i are assumed to be independent and identically distributed, representing claims of exposure unit i. And with probability becoming close to 1, the average loss per exposure unit is in close proximity to the true mean of the loss distribution as N approaches infinity. Then (2.4) can be extended to describe the total amount of claims of N exposure units comprising the pool in a specified period of time:

$$\lim_{N\to\infty} Pr(|\sum_{i=1}^{N} X_i - N\mu| < N\varepsilon) = \lim_{N\to\infty} Pr(|\sum_{i=1}^{N} X_i - N\mu| < k\sigma\sqrt{N}) = 1 \tag{2.5}$$

Consequently, we can deduce the amount of surplus fund that is required for maintaining a predefined probability of insurance pool failure. Substituting k with y_α, where y_α is the standard normal deviate such that $N(y_\alpha)$ is equal to 1 - α, we can get:

$$Pr(|\sum_{i=1}^{N} X_i - N\mu| < y_\alpha\sigma\sqrt{N}) = 1 - \alpha \tag{2.6}$$

It is not hard to find that if the insurance company wants to guarantee a probability $1-\alpha$ that all claims are paid, it must reserve a total surplus fund of $y_\alpha\sigma\sqrt{N}$. Consequently, the surplus fund per exposure unit should be $y_\alpha\frac{\sigma}{\sqrt{N}}$. And this conclusion can be used in

VNF resource allocation, taking the possible workload fluctuations and changing resource demands into account. The availability of VNFs can be guaranteed at a specific level by introducing the statistical risk pooling model. NSs can be provided to end-user with high performance as a part of surplus resources has been reserved to deal with workload increment efficiently.

2.7 Summary

This chapter provided the background of Network Edges and a brief introduction of NFV related technology. We first raised the limitations of centralized clouds and illustrated the emerging Network Edge cloudlet. Then we introduced the modern virtualization technology and highlighted the transition in the industry, from VMs to containers. We discussed Docker containerization principle and Kubernetes management platform. Next, we presented existing NFV-MANO architectures especially VIM and NFVM architectures. The studies and research related to resource reservation and elastic scaling are listed in the subsequent section. In particular, we discussed resource reservation from two aspects as VNF classification and resource allocation. We also discussed recent literature on availability problems and backup models. Following the basic idea of insurance and risk theory, we first applied the statistical theories, i.e., The Central Limit Theorem, to risk pools which can further be used in NFV scenario for resource reservation. Considering the advantages and disadvantages of previous works, we will design a novel containerized NFV-MANO architecture and resource reservation scheme to ensure VNFs' availability and QoS in the following chapters.

Chapter 3

Proposed System Model

3.1 Introduction

Figure 3.1 depicts an example of our adopted NE mode. In our scenario, a server at the NE serves a large number of chained containers. These containers host various cNFCs and are driven by corresponding images (e.g., IDS VNF driven by Snort image). The server passes different VNF requests of clients to specific cNFCs and monitors the status of workflows. In turn, the server communicates and interacts with other decentralized servers through the core network.

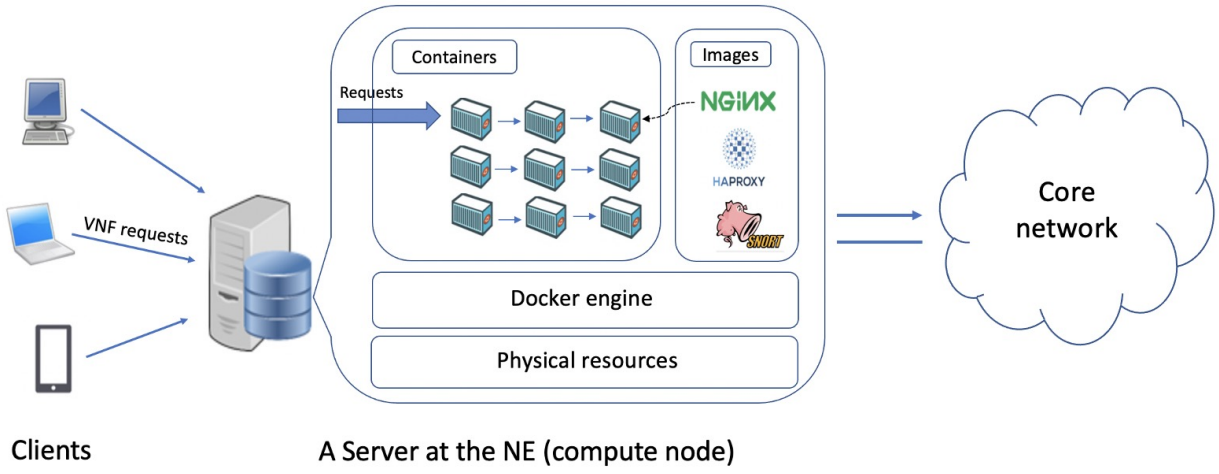


Figure 3.1: An example of a NE hosting VNFs

As mentioned in the previous chapters, the limited availability of resources in the server becomes a vital obstacle when workflows surge. Any insufficient allocation may cause the deterioration of VNF performance and even result in VNF crashes or failures. Therefore, developing a reservation scheme that ensures resource availability is necessary for dealing with unexpected workload fluctuations. On the other hand, once the reservation scheme is proposed, a non-negligible problem is how to integrate the scheme into industrial scenarios.

To solve the aforementioned problems, in this chapter, we first propose a containerized ETSI model including the Resource Reservation Manager and other containerized building blocks. And the corresponding embedded resource reservation schemes for the Resource Reservation Manager module are formulated subsequently. Specifically, section 3.2 presents the containerized ETSI system model. And section 3.3 formulates the allocation scheme of the Resource Reservation Manager considering different scenarios.

3.2 System architecture

Figure 3.2 depicts our proposed system model. Based on existing ETSI architecture, our ETSI model chooses containers as virtualized function units, integrating with resource reservation module. In the following subsections, we first introduce our containerized NFVI architecture, then we mainly discuss the core functional blocks in our proposed MANO architecture, which participate in the resource reservation process.

3.2.1 Proposed NFVI architecture

The NFV infrastructure contains the physical resources at the lower layer of the architecture and represents the available compute, storage and network resources. These physical resources host the VNFs, which represent the second layer, using different resource visualization approaches. In our research scenario, we adopt containerization technology for virtualization considering its efficiency and elasticity. Thus containers are the basic units to host VNFs and fulfill service tasks. Basically, each containerized VNF can be decom-

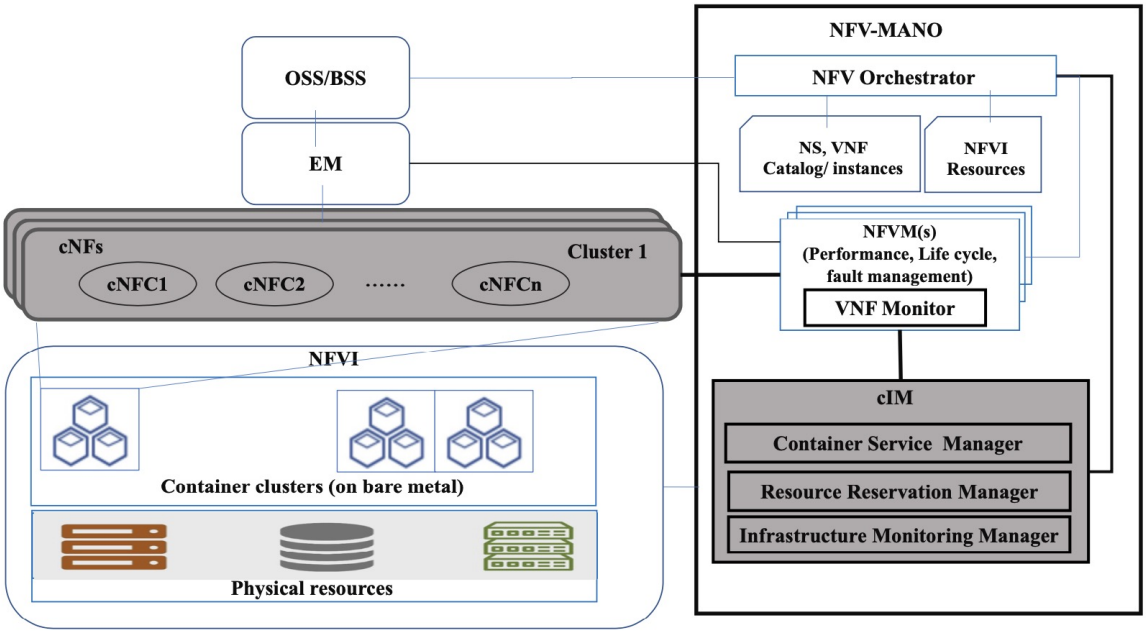


Figure 3.2: Proposed cIM in relation to the ETSI Model

posed into a number of VNF components (VNFCs) and each component is hosted on one container.

3.2.2 Proposed NFV-MANO architecture

The management and Orchestration (MANO) module represents the main component within the ETSI NFV architecture [6] as shown in Figure 3.2. The aforementioned two layers are managed by two corresponding modules in MANO. The NFVI layer is managed by the containerized infrastructure manager (cIM) while the VNFs' resource consumption monitoring, life cycle, performance and fault management is performed by the NFV manager (NFVM). On the other hand, the NFV orchestrator (NFVO) maintains a catalogue describing available network services (NSs) as well as NVF descriptors. The latter describes the decomposition of each containerized VNF into corresponding set of VNFCs and the possible flavours or configurations for each component. These VNFCs are then mapped into containers.

Proposed NFVM architecture

In our system model, NFV manager is responsible for monitoring the traffic workload and resource consumption of VNFs since it provides the management and orchestration from the perspective of VNF instances, recording these VNFs' detailed information. However, existing NFVM architecture is not enough to fulfill resource reservation as reservation needs statistical analysis and calculation especially the changes of traffic and resource demands. Accordingly, a VNF monitoring module namely VNF Monitor is specialized in the NFVM as shown in Figure 3.2. The VNF Monitor is responsible for monitoring the resource consumption of correlated VNFs dynamically, analyzing collected data to get the main parameters (the mean and the variance) for resource reservation. In general, the functionalities offered by our NFVM module are quite similar to the existing NFVM except for the dedicated VNF monitor proposed for resource reservation.

Proposed cIM architecture

Our proposed cIM provides interfaces to both the NFVM and the NFVO in order to define the resource requirements of the deployed VNFCs and then efficiently allocates and manages these resources at the NFVI layer. As shown in the figure, cIM has three building blocks, namely, the container service manager (CSM), the resource reservation manager (RRM) and the infrastructure monitoring manager (IMM). The CSM receives requests from the NFVM to instantiate cNFCs. It then employs existing placement and clustering schemes to place each cNFC into the appropriate cluster after consulting the RRM which decides the amount of needed resources as described in the next section. The CSM then boots the requested cVNFCs. Once cNFs run successfully, the RRM uses monitoring information fed from the VNF monitoring entity in NFVM and the IMM to accurately recalculate the reserved amount for any affected cNFC clusters using risking pool model. The CSM then enforces the desired reservation, allocating the reserved amount of system resource to the specific cNFCs. The next section details the methodology used by the RRM to calculate the reserved pool of resources based on the NFVM requirements in terms of whether higher availability is required to be guaranteed per a single cNFC, a cluster or

multiple clusters of cNFCs. These requirements are assumed to be negotiated between the OSS/BSS and the NFVO.

3.3 Resource Reservation Manager

As discussed in the previous section, the main functionality of the RRM is to reserve sufficient resources for the containers hosting the cNFCs. Allocated resources include computation (CPU cores), memory, storage and bandwidth. Similar to the work in [9], we focus on the allocation of the CPU cores, as the most constrained resources on the NE, however, the presented work can be extended to other resources. We also focus on clustering cNFS instances of identical types, however, other clustering approaches [14] can also be employed. Let I denote the number of different types of cNFs that can be hosted on a single server NE or MEC [55] that is managed by the cIM. Examples of these functions are classifiers, packet analyzers, tunnelling functions (e.g., VPN clients) as well as service monitoring functionalities (e.g., meters and policies) [56]. Let $n_i >> 1$ be the number of expected average of instances in cluster $i \in I$, such that the CPU demand for an instance j, j=1, $\cdots$, n_i is represented by a random variable R_{ij} with a realized demand r_{ij} CPUs at any time. We rely on offline profiling [57] and runtime monitoring, carried out by the NFVM, to continuously estimate the mean μ_{ij} and variance σ_{ij}^2 of r_{ij}. The RRM is tasked with finding a resource allocation π_{ij} CPUs for each container hosting an instance j of type i if the availability constraints are strict with respect to every single instance. Otherwise, it can calculate an overall resource allocation π_i per cluster to achieve a more relaxed availability requirement that is guaranteed per cluster or a group of clusters. We use $1-\alpha$ (α is far less than 1) to denote the probability that the reserved resources are sufficient for workload fluctuations. This probability $1-\alpha$ represents the specified availability level at which the reserved resources are equal to or larger than actual resource demands. As shown in Figure 3.3, to meet various implementation scenarios, the resource reservation schemes for three different circumstances where a cNFC instance, a cNFC cluster or multiple cNFC clusters compose the resource pool are proposed.

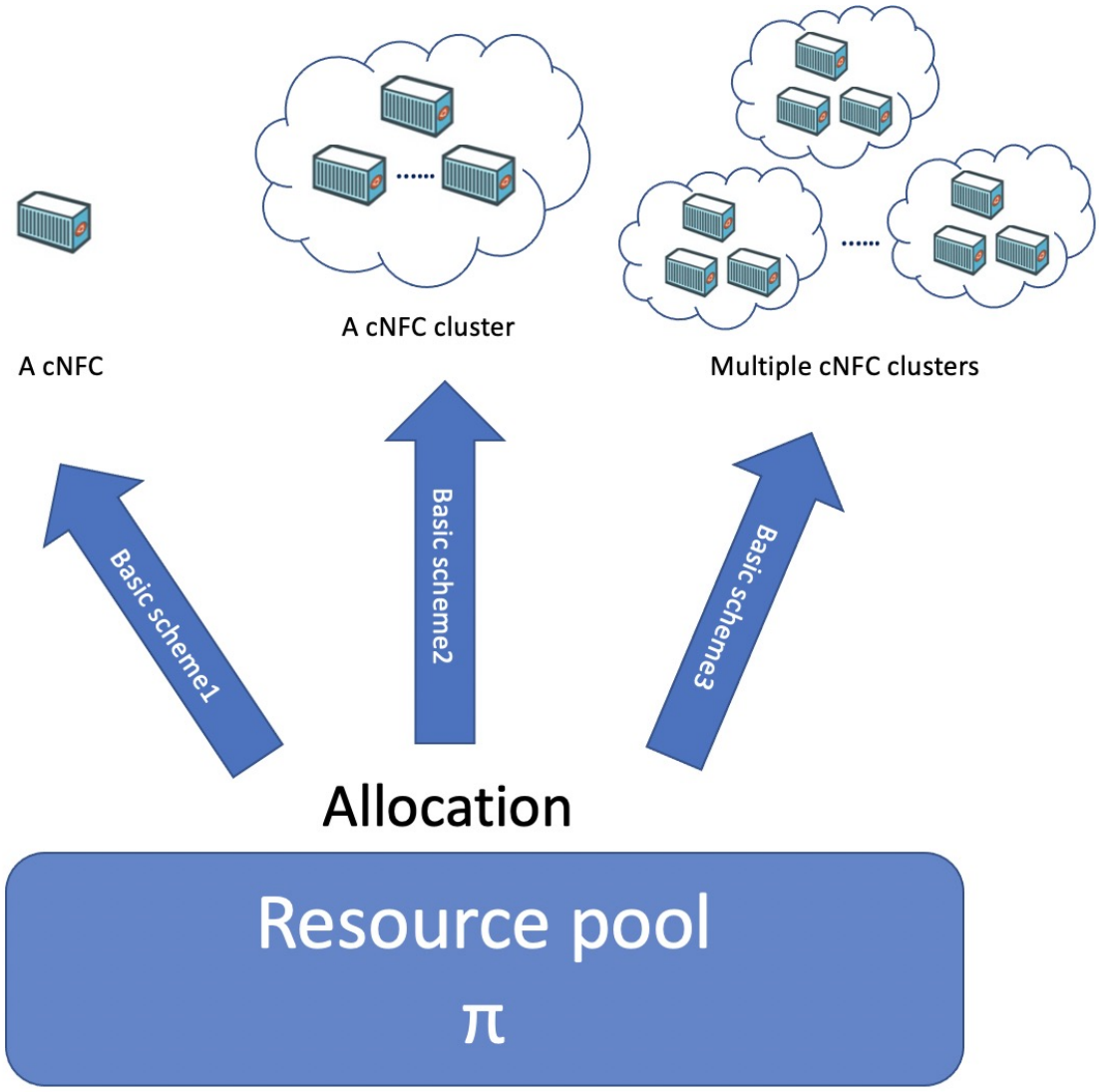


Figure 3.3: The proposed allocation schemes

3.3.1 Resource reservation per a cNFC instance

One straightforward approach to determining π_{ij} is the direct application of Cantelli's inequality for any random variable r_{ij} with a known mean and variance [58], which is defined by the following proposition:

Proposition 1. *Let r_{ij} and π_{ij} be the realized CPU demand and the assigned container CPU, respectively, of a cNFC j of cluster i such that r_{ij} has a monitored mean μ_{ij} and variance σ_{ij}^2. For the RRM to achieve an VNFM request with the resource availability defined as $\Pr(r_{ij} \geq \pi_{ij}) \leq \alpha$, with $0 < \alpha << 1$, π_{ij} is set such that:*

$$\pi_{ij} = \mu_{ij} + \sigma_{ij}\sqrt{\frac{1-\alpha}{\alpha}}. \tag{3.1}$$

Proof. The Cantelli 's inequality states that, given a scalar, $\lambda \geq 0$, we have [58]:

$$\Pr\left(\frac{r_{ij} - \mu_{ij}}{\sigma_{ij}} \leq \lambda\right) \geq \frac{\lambda^2}{1+\lambda^2} \tag{3.2}$$

setting $\lambda = \frac{\pi_{ij}-\mu_{ij}}{\sigma_{ij}}$, we get

$$\Pr(r_{ij} \leq \pi_{ij}) \geq \frac{(\pi_{ij} - \mu_{ij})^2}{\sigma_{ij}^2 + (\pi_{ij} - \mu_{ij})^2} \tag{3.3}$$

With some manipulations and equating $\alpha = \frac{\sigma_{ij}^2}{\sigma_{ij}^2 + (\pi_{ij}-\mu_{ij})^2}$, we obtain π_{ij}. □

Note that the value of π_{ij} specified by proposition 1 is dependent on the availability parameter α and approaches $\mu_{ij} + \sigma_{ij}$ as α diminishes.

3.3.2 Resource reservation per a cNFC cluster

Next, we take advantage of resource sharing by a number of cNFCs, which is already feasible in some cIMs implementations (e.g., using cgroupings in Kubernetes [59]). We will

first assume identical demands for cNFCs within each cluster i such that $\mu_{ij} = \mu_{io}$ and $\sigma_{ij}^2 = \sigma_{io}^2$, $\forall j = 1, \cdots, n_i$. Let r_i be their total CPU demand, such that:

$$r_i = \sum_{j=1}^{n_i} r_{ij},\ \mu_i = \mathrm{E}[r_i] = n_i \mu_{io},\ \sigma_i^2 = \mathrm{Var}[r_i] = n_i \sigma_{io}^2 \tag{3.4}$$

The RRM can allocate a shared pool of CPUs for the n_i cNFCs according to the following proposition.

Proposition 2. *Let r_i and π_i be the sum of the realized CPU demand and assigned CPUs, respectively, of n_j identical but independent cNFCs within cluster i, each with a monitored mean μ_{io} and variance σ_{io}^2. For the RRM to achieve an VNFM request with the resource availability defined as $\Pr(r_i \geq \pi_i) \leq \alpha$, with $0 < \alpha << 1$, π_i is set such that:*

$$\pi_i = n_i \mu_{io} + z_{1-\alpha} \sigma_{io} \sqrt{n_i} \tag{3.5}$$

where $z_{1-\alpha}$ is the standard normal deviate of the normal distribution $\phi(.)$, such that $\phi(z_{1-\alpha}) = \alpha$.

Proof. For a large n_i, the central limit theorem indicates that the probability distribution of the random variable $\frac{r_i - \mu_i}{\sigma_i}$ approaches the standard distribution $\phi(.)$ [15]. Hence, we have:

$$\Pr\left(\frac{r_i - \mu_i}{\sigma_i} \geq z_{1-\alpha}\right) \leq \alpha \tag{3.6}$$

Multiplying both sides inside the probability bracket by σ_i then adding μ_i to both sides and setting π_i as in (3.5), we obtain the result. □

The above proposition indicates that an additional amount of a shared pool of CPUs that is equal to the second term in (3.5), $z_{1-\alpha} \frac{\sigma_{io}}{\sqrt{n_i}}$, is sufficient to meet the cNFCs' demand that exceeds their mean values.

It is also worth noting that on average, the consumed resources per a single cNFC can

be calculated by dividing π_i equally among the n_i cNFCs, and we get,

$$\pi_{ij} = \mu_{io} + z_{1-\alpha}\frac{\sigma_{io}}{\sqrt{n_i}}. \tag{3.7}$$

In other word, the excess resource that the RRM needs to allocate for one additional cNFC, $z_{1-\alpha}\frac{\sigma_{io}}{\sqrt{n_i}}$, decreases as more cNFCs share the common resource pool π_i.

An immediate observation is that as the number of cNFCs sharing the same reserved pool increases, the additional quota needed for each additional cNFC decreases significantly. In other words, resources are saved by allowing more cNFCs to share the additional resources pool while maintaining the same resource availability requirements.

It is worth noting that the demand for the various cNFCs of the same cluster i may become interdependent as flows of served traffic are redistributed among cNFCs within the same SFC as a result of load balancing. In this case, the variance of r_i can be calculated taking into account the covariance of these cNFCs, as follows:

$$\mathrm{Var}[r_i] = n_i\sigma_{io}^2 + 2\sum_{j=2}^{n_i}\sum_{k=1}^{j-1}\sigma_{jk} \tag{3.8}$$

where σ_{jk} is the monitored covariance between two cNFCs j and k of cluster i.

Then the shared CPU reserved for the n_i containers becomes:

$$\pi_i = \sum_{j=1}^{n_i}\mu_{ij} + z_{1-\alpha}\sqrt{\sum_{j=1}^{n_j}\sigma_{ij}^2 + 2\sum_{j=2}^{n_i}\sum_{k=1}^{j-1}\sigma_{jk}} \tag{3.9}$$

Similar to the case of independent cNFCs, as n_i increases the main excess resource added per an additional amount of can be calculated as $z_{1-\alpha}\sqrt{\frac{2\sum_{j=2}^{n_i}\sum_{k=1}^{j-1}\sigma_{ij}}{n_i(n_i-1)}}$.

3.3.3 Resource reservation for multiple cNFC clusters

A further reduction in the needed excess pool of resources can be achieved if multiple clusters are allocated on a large single NE server where small cNFCs can consume part

of the shared quota. Following this scenario, let r be the total demand on the NE server, such that:

$$r = \sum_{i=1}^{I} r_i,\ \mu = \sum_{i=1}^{I} n_i \mu_{io},\ \sigma^2 = \sum_{i=1}^{I} n_i \sigma_{io}^2 \tag{3.10}$$

Then the following proposition defines the amount of CPUs to be reserved for a desired availability level.

Proposition 3. *Let r and π be the sum of the realized core CPU demand and allocations, respectively, of I clusters of cNFCs, where each cluster i has n_j identical and independent cNFCs, each with a monitored mean μ_{io} and variance σ_{io}^2. For the RRM to achieve a VNFM request with $\Pr(r \geq \pi) \leq \alpha$, such that $0 < \alpha << 1$, π is set such that:*

$$\pi = \sum_{i=1}^{I} n_i \mu_{io} + z_{1-\alpha} \sqrt{\sum_{i=1}^{I} n_i \sigma_{io}^2} \tag{3.11}$$

where $z_{1-\alpha}$ is the standard normal deviate such that $\phi(z_{1-\alpha}) = \alpha$.

Proof. As the number of cNFCs increases, the probability distribution of $\frac{r-\mu}{\sigma}$ approaches $\phi(.)$ [15] and we have,

$$\Pr\left(\frac{r-\mu}{\sigma} \geq z_{1-\alpha}\right) \leq \alpha \tag{3.12}$$

Multiplying both sides inside the probability bracket by σ and adding μ then setting π as in (3.11), we obtain the result. □

With the above proposition, it is worth noting that the excess resources $z_{1-\alpha}\sqrt{\sum_{i=1}^{I} n_i \sigma_{io}^2}$ needed is smaller than the sum of all individually allocated resources for each cluster as defined in proposition 2, $\sum_{i=1}^{n_i}(z_{1-\alpha}\sigma_{io}\sqrt{n_i})$. This is clear by noting that $\sum_{i=1}^{I}(\sqrt{n_i}\sigma_{io})^2 \leq (\sum_{i=1}^{I}\sqrt{n_i}\sigma_{io})^2$. This leads to NE resource savings while ensuring the desired availability.

The problem that remains is how to calculate the resource pool size needed for each cluster in order to provide an accurate pricing scheme as can be seen in chapter 4. Several approaches can be used to calculate the required pool size for each cluster $k_{c,i}$. First, let

the required pool size $x_e^{c,i}(t)$ for each cluster $k_{c,i}$ be defined as a function of $T_{c,i}$ as follows,

$$x_e^{c,i}(t) = n_{c,i}\mu_{c,i}(t) + z_{1-\alpha_c} n_{c,i} T_{c,i} \tag{3.13}$$

where $T_{c,1}, ..., T_{c,|K_c|}$ are non-negative numbers such that,

$$\sum_{i=1}^{|K_c|} n_{c,i} T_{c,i} = \sqrt{\sum_{i=1}^{|K_c|} n_{c,i}\sigma_{c,i}^2(t)} = \sigma_c(t) \tag{3.14}$$

Clearly, there is an infinite number of options to determine the value of $T_{c,i}$, where $i = 1, ..., |K_c|$. These approaches include the following:

1. ***Uniform allocation***

   The class standard deviation $\sigma_c(t)$ defined in (3.14), is split equally among all the applications within this class, that is $T_{c,1} = ... = T_{c,|K_c|} = \frac{\sigma_c(t)}{n_c}$,where $n_c = \sum_{i=1}^{|K_c|} n_{c,i}$. The reserved pool size for cluster $k_{c,i}$ is:

   $$x_e^{c,i}(t) = n_{c,i}\mu_{c,i}(t) + z_{1-\alpha_c} n_{c,i} \frac{\sigma_c(t)}{n_c} \tag{3.15}$$

2. ***Semi-uniform allocation***

   The class standard deviation $\sigma_c(t)$ is split equally among the clusters. Here $T_{c,i} = \frac{\sigma_c(t)}{n_{c,i}|K_c|}$. The reserved pool size for cluster $k_{c,i}$ is:

   $$x_e^{c,i}(t) = n_{c,i}\mu_{c,i}(t) + z_{1-\alpha_c} \frac{\sigma_c(t)}{|K_c|} \tag{3.16}$$

3. ***Proportional allocation***:

   The class variance $\sigma_c^2(t)$ is split among the clusters proportionally to their variances. Thus, $T_{c,i}$ is determined as $\frac{r_{c,i}\sigma_c(t)}{n_{c,i}}$, where $r_{c,i} = \frac{n_{c,i}\sigma_{c,i}^2(t)}{\sigma_c^2(t)}$. The reserved pool size for cluster $k_{c,i}$ is:

   $$x_e^{c,i}(t) = n_{c,i}\mu_{c,i}(t) + z_{1-\alpha_c} r_{c,i}\sigma_c(t) \tag{3.17}$$

At this point, the problem of accurately determining the required pool size for each cluster $k_{c,i}$ will be modelled as a non-linear optimization problem. These three allocation approaches stand for different allocation thoughts and can be chosen dynamically considering the difference between each cluster. For example, if the differences of resource demand or consumption between each cluster are lower than a preset threshold, we can adopt uniform allocation to simplify the allocation work as the weighted coefficient Tc, i of each cluster in uniform approach is the same. The three approaches balance the trade-off between distinct allocation and simplified allocation. In our simulations of multiple cNFC clusters composing the whole resource pool, we used the third allocation scheme (proportional allocation) to calculate the weighted resources reserved for each cluster since the workload pattern difference between each of our clusters is relatively large.

3.4 Summary

In this chapter, a novel containerized ETSI model was proposed with dedicated modules fulfilling the resource reservation process. We first introduced the basic building blocks of our system model such as NVF infrastructure, NFV manager and containerized infrastructure manager. In particular, we designed a unique resource reservation manager to calculate the amount of resources reserved for each VNF considering demand changes and traffic fluctuations. Then we derived the statistical calculation method of RRM based on risk pooling theory. We further discussed the reservation scheme in different usage scenarios where the resource pool consists of a cNFC instance, a cNFC cluster or multiple cNFC clusters. For the resource pool containing a portfolio of multiple clusters, the generalized reservation solution was presented first, which was followed by three specific allocation schemes as uniform allocation, semi-uniform allocation and proportional allocation, to illustrate the basic idea of this general approach.

The reservation scheme offers surplus resources for VNFs that suffer from resource scarcity and thus are unable to achieve high performance when workflow increases. The availability and efficiency can be guaranteed as our scheme precisely reserves resources

needed by each VNF to respond to traffic fluctuations based on historical workload pattern and resource consumption, avoiding resource waste. Based on the basic solution for resource reservation, we will evaluate and optimize our scheme from different perspectives, taking some crucial factors that affect the performance of VNFs and the profit of service providers into account.

Chapter 4

Optimization Problems and Corresponding solutions

4.1 Introduction

In the previous chapter, we have shown that allocating a common pool of resources π for I clusters of different types of cNFCs is more efficient than allocating individual resources per component or cluster. However, once the value of π is calculated, we need an efficient means to decide the initial distribution of that resource pool over individual clusters. In this chapter, we address this problem. More precisely, we address the following problem, given a reserved pool of resources π, what is the optimal allocation of resources to each cluster in order to maximize the performance of cNFCs in these clusters. We then address a different problem, that is how can the cIM signal the VNFM with the status of the VNFI resources in order to adjust the resource demands. As shown in Figure 4.1, when doing resource allocation, we add different optimization mechanisms in cIM. VNFM is responsible for passing the allocation requirement and monitoring information of each cNFC to cIM. In turn, cIM sends the optimally calculated allocation and pricing scheme to VNFM based on the obtained information from VNFM and IMM.

In the following sections, we solve these two problems using different assumptions with regards to approximating the functions that measure the performance loss of different

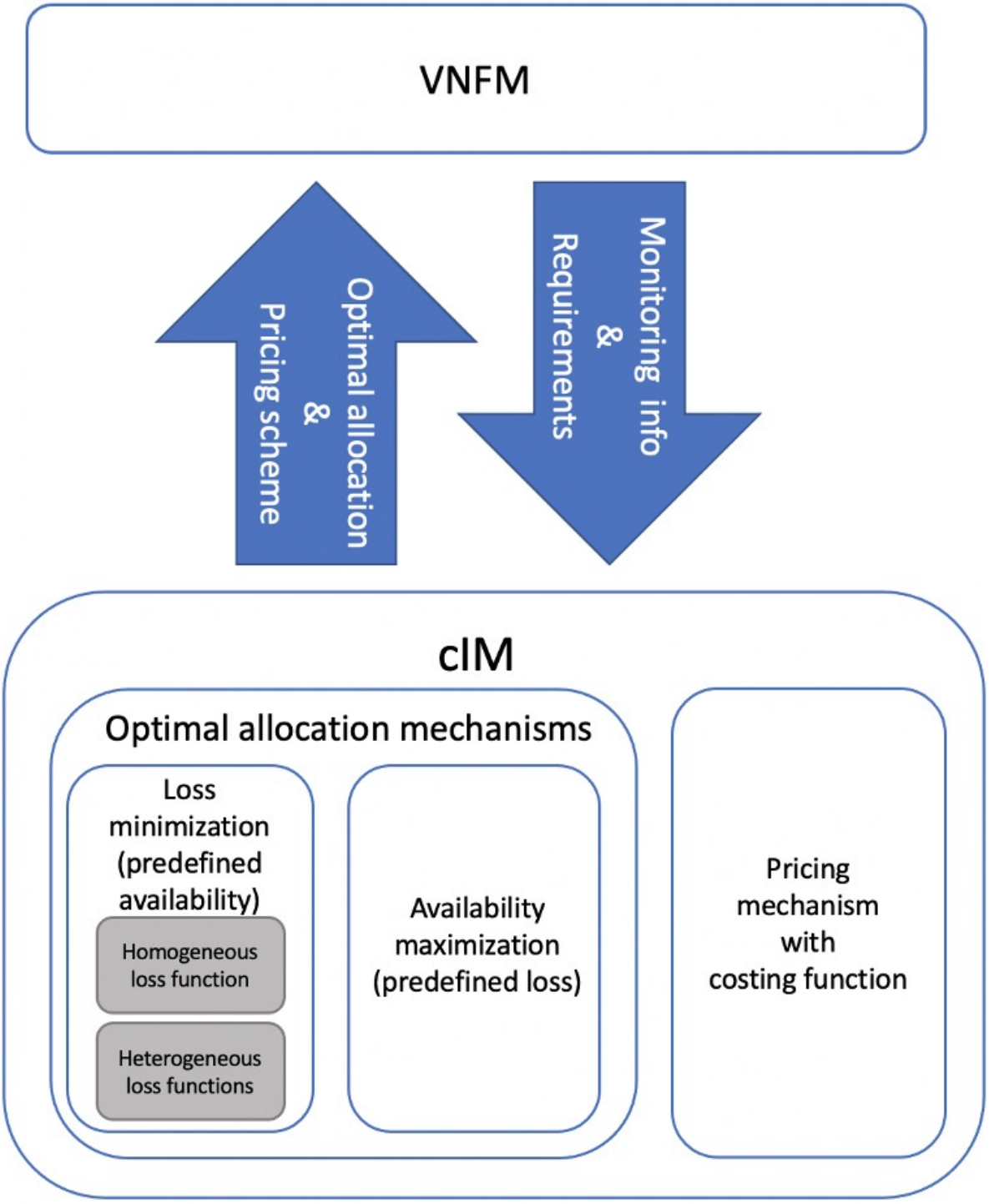


Figure 4.1: Optimization allocation between cIM and VNFM

cNFCs due to resource insufficiency. We employ concepts borrowed from the properties of vector majorization and Schur-convex functions [60].

4.2 Measuring the performance of cNFCs

To measure the performance of cNFCs, the cNFCs profiling work which is based on the previous research in the field of VNF profiling and the pattern analysis of VNF performance is needed. As discussed in the research [61], VNFs can behave differently in the SFC versus standalone situation, and also when the SFC's order is changed. Consequently, both standalone VNFs and heterogeneous SFCs need to be classified first to simplify the performance measurement process. In other words, cNFCs with similar workload patterns and resource demands are sorted into the same cluster. Then we only measure and analyze the performance of cNFCs in the same cluster. The research in the profiling work [62] demonstrates that not only the VNFs themselves influence the chain behaviour, but also the underlying support layer that connects these VNFs such as the networking platform and basic infrastructure platform, may make a difference to the VNF performance. However, in our implementation scenario, since the networking platform (Kubernetes) integrating with the infrastructure platform (Docker) is lightweight and nearly latency-free, the underlying forwarding plane has no significant effect on the whole chain performance. Thus, we only measure the performance of cNFCs themselves, monitoring the workflow status when passing each cNFC in the chain.

4.3 Optimal allocation of resource pools to cNFCs clusters

In this section, we build on the results obtained in Proposition 3 where an optimal allocation of a pool of resources is found in order to ensure that the probability of the demand of the cNFCs in the I clusters is met. However, the proposition did not specify how this pool of resources can be allocated among different clusters. Hence, the main objective

here is to find an optimal initial sharing of this pool along the clusters in order to ensure that the performance of the cNFCs in each cluster is optimal. The sharing policy must still satisfy the condition that the probability of the total demanded CPU exceeding the total reserved pool is a pre-determined small number α, $0 < \alpha < 1$. In addition, the reserved pool size should be proportional to the expected fluctuation for each cluster. In other words, the reserved pool size should increase as the expected fluctuation increases. This problem is modelled as a non-linear optimization problem such that the sum of the expected performance losses of different instances of various cNFCs is minimized. Since we are dealing with different cNFC types, we will first assume that the performance loss of all components can be estimated using the same loss function, then we will relax this assumption to employ a different performance loss function for each cluster.

4.3.1 Optimal sharing with a homogeneous loss function

Assume that, at runtime, the cIM has allocated a pool π, as calculated in the pervious chapter (eq. (3.11)), to the cNFCs in the I clusters and that at runtime we can monitor the deviation of the allocated resource π_i for each cluster i from the actual resource demand r_i. Using VNF profiling schemes, we can approximate the performance loss by a function $f(r_i - \pi_i)$, $f : \mathbb{R} \to [0, \infty]$. More precisely, f is a function that measures the performance degradation of a cluster of n_i cNFCs of type i. Our objective is, then, to find an allocation vector $\boldsymbol{\pi} = (\pi_1, \cdots, \pi_I)$ that minimizes these losses. We will first review some important properties related to vector majorization and Schur-convex functions [60].

Majorization and Schur-convexity

We first define the concept of majorization according to the following definition [60]:

Definition 1. *A vector* $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ *is said to majorize another vector* $\boldsymbol{y} =$

$(y_1, \cdots, y_n) \in \mathbb{R}^n$, *symbolically denoted* $\boldsymbol{x} \succ \boldsymbol{y}$, *if*

$$\sum_{i=1}^{k} x_{(i)} \geq \sum_{i=1}^{k} y_{(i)}, \quad \forall k = 1, \cdots, n-1, \tag{4.1}$$

$$\text{and } \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i. \tag{4.2}$$

and $x_{(i)}$ *is the* i^{th} *element in* $\boldsymbol{x}$, *after rearrangement of its element, such that* $x_{(1)} \geq x_{(2)} \geq \cdots x_{(n)}$.

The definition shows that majorization is a partial order relation on vectors with components that sum up to the same value. It is also a measure of how many elements within the same vector differ.

It can easily be noted that all allocation vectors majorize the uniform allocation vector where $x_1 = x_2 = \cdots = x_n$. Next, we provide a definition for the *Schur-convex* functions ([60], Ch.3).

Definition 2. *A function* $f : \mathbb{R}^n \Rightarrow \mathbb{R}$ *Schur-convex if*

$$\boldsymbol{x} \succ \boldsymbol{y} \Rightarrow f(\boldsymbol{x}) \geq f(\boldsymbol{y}). \tag{4.3}$$

Schur-convex functions are defined to be consistent with majorization. They map the majorization ordering of the vectors to an equivalent scalar ordering. A clear example of convex functions in this context is the maximum function on the elements of the vector. Other examples of convex functions that are used for loss functions are the powers of absolute values $|x|^a$, $a \leq 1$ on $\mathbb{R}$ and the powers x^a, $a \leq 1$ or $a < 0$ and logarithmic $\log x$ on real positive values $\mathbb{R}_+$.

As will be shown, they can be used to represent the collective performance loss for various types of cNFCs. The following theorem will also be useful in our context.

Theorem 1. *([60], Ch.3,C.1.) Let* $f : \mathbb{R} \to \mathbb{R}$ *be a convex function and* $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$, $\boldsymbol{y} = (y_1, \cdots, y_n) \in \mathbb{R}^n$, *then the function* $g : \mathbb{R}^n \to \mathbb{R}$, *such that* $g(\boldsymbol{x}) = \sum_{i=1}^{n} f(x_i)$, *is*

Schur-convex, i.e.,

$$\boldsymbol{x} \succ \boldsymbol{y} \Rightarrow g(\boldsymbol{x}) \geq g(\boldsymbol{y}). \tag{4.4}$$

Resource allocation using Schur-convexity properties

We are interested in employing Schur-convexity of combinations of Schur-convex functions in order to obtain the optimal resource assignment for each cluster as described by the following proposition.

Proposition 4. *Let π be the sum of the reserved CPU for I clusters of cNFCs, where each cluster, i, $i = 1, \cdots, I$ has n_i identical cNFCs, each with a monitored demand mean μ_{io} and variance σ_{io}^2. Let $f : \mathbb{R} \Rightarrow \mathbb{R}$ be a convex function representing the performance loss of any cluster. An allocation $\pi = (\pi_1, \cdots, \pi_I)$, that minimizes the clusters performance loss while ensuring that $\Pr(\sum_{i=1}^{I} r_i \leq \pi) \leq \alpha$, with $0 < \alpha << 1$, is the solution of the following optimization problem:*

$$\boldsymbol{\pi}^* = \quad \arg\min_{\boldsymbol{\pi}} \left\{ \sum_{i=1}^{I} (f(\pi_i - n_i\mu_{io})) \right\} \tag{4.5}$$

$$s.t. \quad \Pr(r \leq \pi) \leq \alpha, \tag{4.6}$$

$$\pi_i \geq n_i\mu_{io}, \quad i = 1, \cdots, I. \tag{4.7}$$

$$and \quad \sum_{i=1}^{I} \pi_i = \pi. \tag{4.8}$$

This problem has a unique solution when f is strictly convex that is obtained when $\pi_i = n_i\mu_{io} + \frac{z_{1-\alpha}\sqrt{\sum_{i=1}^{I} n_i\sigma_{io}^2}}{I}$, $i = 1, \cdots, I$.

Proof. Let $\sigma = \sqrt{\sum_{i=1}^{I} \sigma_i^2} = \sqrt{\sum_{i=1}^{I} n_i\sigma_{io}^2}$. From proposition (3.11), the constraint (4.6) is satisfied when $\pi = \mu + z_{1-\alpha}\sigma$, where $\mu = \sum_{i=1}^{I} n_i\mu_{io}$ Let $x_i = (\pi_i - n_i\mu_{io})$, the above

problem can be rewritten as

$$\boldsymbol{x}^* = \quad \arg\min_{\boldsymbol{x}} \left\{ \sum_{i=1}^{I} f(x_i) \right\} \tag{4.9}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} x_i = z_{1-\alpha}\sigma \tag{4.10}$$

$$\text{and} \quad x_i \geq 0, \quad i = 1, \cdots, I. \tag{4.11}$$

from Theorem 1, the solution to the above problem is a vector $\boldsymbol{x}^*$, that minimizes $g(\boldsymbol{x}) = \sum_{i=1^I} f(x_i)$, that is majorized by all other vectors in the feasible solutions set where $\sum_{i=1}^{I} x_i = z_{1-\alpha}\sigma$. This vector is the uniform vector where $x_i = \frac{z_{1-\alpha}\sigma}{I}$, $\forall i$. Hence, the optimal solution for our problem is $\pi_i = n_i\mu_{io} + \frac{z_{1-\alpha}\sigma}{I}$. □

Using the above proposition, it is sufficient to measure the overall variance σ of the CPU consumption of the I clusters and assign to each cluster i, its average consumption $n_i\mu_{io}$ in addition to an additional pool of resources $z_{1-\alpha}\frac{\sigma}{I}$, here $z_{1-\alpha}$ represents the $(1-\alpha)$ percentile of the normalized total consumption.

4.3.2 Optimal sharing with heterogeneous loss functions

In the previous section, the required pool size for each cluster was found such that a uniform expected loss function was assigned to all the cNFCs.

Proposition 5. *Let π be the sum of the reserved CPU for I clusters of cNFCs, where each cluster, i, $i = 1, \cdots, I$ has n_i identical cNFCs, each with a monitored demand mean μ_{io} and variance σ_{io}^2. Let $f_i : \mathbb{R} \Rightarrow \mathbb{R}$ be a convex function representing the performance loss of cNFCs in each cluster i. An allocation $\pi = (\pi_1, \cdots, \pi_I)$ that minimizes the cluster performance loss while ensuring that $\Pr(\sum_{i=1}^{I} r_i \leq \pi) \leq \alpha$, such that $0 < \alpha << 1$, π is the*

solution of the following optimization problem:

$$\boldsymbol{\pi}^* = \quad \arg\min_{\boldsymbol{\pi}} \left\{ \sum_{i=1}^{I} f_i(\pi_i - n_i\mu_{io}) \right\} \tag{4.12}$$

$$s.t. \quad \Pr(\sum_{i=1}^{I} r_i \leq \pi) \leq \alpha \tag{4.13}$$

$$\pi_i \geq n_i\mu_{io}, \quad i = 1, \cdots, I, \tag{4.14}$$

$$and \quad \sum_{i=1}^{I} \pi_i = \pi \tag{4.15}$$

This problem has a unique solution when $f_i()$, $i = 1, \cdots, I$ are strictly convex that is obtained when

$$\frac{\partial f_1(\pi_1 - n_1\mu_{1o})}{\partial(\pi_1 - n_1\mu_{1o})} = \cdots = \frac{\partial f_i(\pi_i - n_i\mu_{io})}{\partial(\pi_i - n_i\mu_{io})} = \cdots = \frac{\partial f_I(\pi_I - n_I\mu_{Io})}{\partial(\pi_I - n_I\mu_{Io})} \tag{4.16}$$

and $\sum_{i=1}^{I}(\pi_i - n_i\mu_{io}) = z_{1-\alpha}\sigma$

Proof. Let $\mu_i = n_i\mu_{io}$ and $x_i = (\pi_i - \mu_i)$, the Lagrangian of the above problem can be written as follows:

$$\mathcal{L}(\boldsymbol{x}, \lambda) = \sum_{i=1}^{I} f_i(x_i + \lambda(z_{1-\alpha}\sigma - \sum_{i=1^I} x_i) \tag{4.17}$$

the KKT conditions:

$$\frac{\partial f_i(\pi_i - \mu_i)}{\pi_i - \mu_i} + \lambda = 0, \quad i = 1, \cdots, I \tag{4.18}$$

$$\lambda(z_{1-\alpha}\sigma - \sum_{i=1^I} x_i) = 0 \tag{4.19}$$

solving for x_i leads to the solution equations. □

The following lemma considers a special case where the performance loss function is measured as the weighted sum of the square differences between the CPU demand and allocation. In other words, performance loss occurs in both cases of over- and under-provisioning of resources to the cNFCs. The weights assigned to each cluster, however, dictate the criticality of allocation loss or excess for each cluster.

Lemma 1. *Let the performance loss function of any cNFC be measured as the square difference between the CPU demand and allocation, such that the performance of the cIM is represented as a weighted sum of these functions, then CPU resource allocation problem can be defined as follows,*

$$\boldsymbol{\pi}^* = \quad \arg\min_{\boldsymbol{\pi}} \left\{ \sum_{i=1}^{I} \left[\frac{1}{\omega_i} (n_i \mu_{io} - \pi_i)^2 \right] \right\} \tag{4.20}$$

$$s.t. \quad \Pr(sum_{i=1}^{I} r_i \leq \pi) \leq \alpha, \tag{4.21}$$

$$\sum_{i=1}^{I} \pi_i = \pi \tag{4.22}$$

$$and \quad \sum_{i=1}^{I} \omega_i = 1. \tag{4.23}$$

This problem has a unique solution which is:

$$\pi_i^* = n_i \mu_{io} + z_{1-\alpha} \omega_i \sigma \qquad i = 1, \ldots, I \tag{4.24}$$

Proof. The solution to the above problem can be derived directly from Proposition 5, where $f_i(\pi_i - \mu_i) = \frac{1}{\omega_i} g(\pi_i - \mu_i)$,and $g(x_i) = x_i^2$. Hence, $\frac{\partial f_i(\pi_i - \mu_i)}{\partial(\pi_i - \mu_i)} = \frac{2(\pi_i - \mu_i)}{\omega_i}$, and we have

$$\frac{2(\pi_1 - \mu_1)}{\omega_1} = \cdots = \frac{2(\pi_i - \mu_i)}{\omega_i} = \cdots = \frac{2(\pi_I - \mu_I)}{\omega_I} \tag{4.25}$$

which can be written as

$$\frac{(\pi_i - \mu_i)}{\omega_i} = \frac{\sum_{i=1}^{I} (\pi_i - \mu_i)}{\sum_{i=1}^{I} \omega_i} = z_{1-\alpha} \sigma \tag{4.26}$$

and we have $\pi_i = \mu_i + \omega_i z_{1-\alpha} \sigma$ □

4.3.3 CPU distribution on clusters given predefined loss

In this section, we start by assuming a limited resource availability at the VNFI. Hence, we solve the problem of CPU pool distribution on I clusters in order to minimize the

probability of not satisfying the cNFCs resource while ensuring that the performance loss does not decrease below a given threshold.

The new optimization problem can be modeled as follows,

$$\boldsymbol{\pi}^* = \quad \arg\min_{\boldsymbol{\pi}} \left\{ Pr\left(\sum_{i=1}^{I} r_i > \sum_{i=1}^{I} \pi_i \right) \right\} \tag{4.27}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \left[\frac{1}{\omega_i} (r_i - \pi_i)^2 \right] \leq \psi \tag{4.28}$$

$$\sum_{i=1}^{I} \pi_i = \pi, \tag{4.29}$$

$$\mu_i \leq \pi_i \qquad \forall i = 1, \ldots, I \tag{4.30}$$

$$\tag{4.31}$$

Proposition 6. *The minimization problem (4.27) has a unique solution which is:*

$$\pi_i{}^* = n_i \mu_{io} + \omega_i \sqrt{\Psi} \qquad \forall i = 1, \ldots, I \tag{4.32}$$

Where $\Psi = \psi - \sum_{i=1}^{I} \frac{n_i \sigma_{io}^2}{\omega_i}$

Proof. First, both sides of the objective function inequality are normalized as follows,

$$Pr\left(\sum_{i=1}^{I} r_i > \sum_{i=1}^{I} \pi_i \right) = Pr\left(\frac{\sum_{i=1}^{I} (r_i - \mu_{io})}{\sigma} > \frac{\sum_{i=1}^{I} \pi_i - \mu_i}{\sigma} \right) \tag{4.33}$$

Minimizing the left-hand side of the objective function of (4.33) with respect to π_i, where $i = 1, \ldots, I$ is equivalent to the maximization of $\sum_{i=1}^{I} (\pi_i - n_i \mu_{io})$. The problem can then be rewritten as follows,

$$\boldsymbol{\pi} = \quad \arg\max_{\boldsymbol{\pi}} \left\{ \sum_{i=1}^{I} (\pi_i - n_i \mu_{i0}) \right\} \tag{4.34}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \frac{n_i \sigma_{io}^2}{\omega_i} + \sum_{i=1}^{I} \frac{1}{\omega_i} (\pi_i - n_i \mu_{io})^2 = \psi \tag{4.35}$$

$$n_i \mu_{io} \leq \pi_i \qquad \forall i = 1, \ldots, I \tag{4.36}$$

$$\tag{4.37}$$

Let $\Psi = \psi - \sum_{i=1}^{I} \frac{n_i \sigma_{io}^2}{\omega_i}$, and $x_i = \pi_i - n_i \mu_{io}$. Problem (4.37) can then be rewritten as follows,

$$\arg\max_{\boldsymbol{x}} \left\{ \sum_{i=1}^{I} x_i \right\} \tag{4.38}$$

$$\sum_{i=1}^{I} \frac{1}{\omega_i} x_i^2 - \Psi = 0, \tag{4.39}$$

$$x_i \geq 0 \qquad \forall i = 1, \ldots, I \tag{4.40}$$

To solve this problem, Lagrange multipliers are used as follows,

$$\mathcal{L}(\boldsymbol{x}, \lambda) = \sum_{i=1}^{I} x_i + \lambda \left(\sum_{i=1}^{I} \frac{1}{\omega_i} x_i^2 - \Psi \right) \tag{4.41}$$

Differentiating $\mathcal{L}(\boldsymbol{x}, \lambda)$ with respect to $x_1, \cdots, x_I$ and λ yields a system of I equations:

$$\begin{cases} \frac{\partial \mathcal{L}(\boldsymbol{x}, \lambda)}{\partial x_i} = 1 + 2\lambda \frac{x_i}{\omega_i} = 0 \qquad \forall i = 1, \ldots, I \\ \frac{\partial \mathcal{L}(\boldsymbol{x}, \lambda)}{\partial \lambda} = \sum_{i=1}^{I} \frac{1}{\omega_i} x_i^2 - \Psi = 0 \end{cases} \tag{4.42}$$

From the first I equations of (4.42) the following expression is obtained:

$$x_i = -\frac{\omega_i}{2\lambda} \qquad \forall i = 1, \ldots, I \tag{4.43}$$

Since $x_i \geq 0$ then λ has to be negative. Substituting (4.43) in the last equation of (4.42) yields,

$$\sum_{i=1}^{I} \frac{\left(-\frac{\omega_i}{2\lambda}\right)^2}{\omega_i} = \Psi \tag{4.44}$$

hence,

$$\frac{1}{4\Psi} \sum_{i=1}^{I} \omega_i = \lambda^2 \tag{4.45}$$

with $\sum_{i=1}^{I} \omega_i = 1$. Since $\lambda < 0$, substituting the negative root of λ from (4.45) in the first I equations of (4.42) yields a unique solution which is:

$$x_i^* = -\frac{\omega_i}{2\lambda} = \omega_i \sqrt{\Psi} \qquad \forall i = 1, \ldots, I \tag{4.46}$$

Hence, (4.32) is the unique solution to (4.31). □

4.3.4 Pricing as a signal for resource consumption adjustment

In the previous sections, we calculated the value of the required pool size π_i of CPU for each cluster i that can be allocated by the cIM on the VNFI. Here, we would like to use a pricing mechanism as a signal communicated by the cIM to the VNFM in order to regulate the requested resource demands. Hence, in this section, the problem of deciding the optimal resource allocation among various cNFC clusters is modelled as an optimization problem with the objective of maximizing a hypothetical monetary profit of the cIM.

We assume that the VNFM models the CPU demand of each cluster i, is a function $r_i(p_i)$ of the price signal p_i sent by the cIM. In other words, the value of p_i indicates to the VNFM how scarce or available the CPU resources are at the VNFI. In turn, the VNFM decides on the corresponding demand r_i. For simplicity, we assume a constant elastic relation between the resource demand and the price signal, i.e., $(r_i) = \frac{K_i}{p_i}$, with a constant $K_i > 0$, however, our model can be easily extended to include more complex functions.

The cIM's objective is to regulate the allocated CPU consumption for each cluster once the resource pool π has been calculated. Hence, it maintains a workload ratio ψ_i for each cluster, such that $\psi_i = \frac{\pi_i}{\pi}$. cIM then calculates a basic price signal for a unit of CPU p and an adjusted vector of prices $(p_1, \cdots, p_I)$ for all the I clusters, where $p_i = \frac{p}{\psi_i}$. The problem of calculating these price signals can be formulated as that of maximizing cIM's profit of selling to the VNFM the desired CPU using these prices after deducting the rental cost of the physical CPU from the underlying VNFI while satisfying the constraint that the sum of the actual allocated CPU for all the clusters does not exceed the available pool of resources, i.e., $\sum_{i=1}^{I} \frac{r_i(p_i)}{\psi_i} \leq \pi$. Formally, this problem can be modelled as follows:

$$\max_{(p_1,\cdots,p_I)} = \sum_{i=1}^{I} r_i(p_i) \times p_i - C(\pi) \tag{4.47}$$

$$\text{s.t.} = \sum_{i=1}^{I} \frac{r_i(p_i)}{\psi_i} \leq \pi. \tag{4.48}$$

where $C(\pi)$ is the cost of maintaining the pool of CPUs π. Noting that the Lagrangian for (4.47) can be written as:

$$\mathbf{L}(p_1, \cdots, p_I, \lambda) = \sum_{i=1}^{I} r_i(p_i) \times p_i - C(\pi) + \lambda(\pi - \sum_{i=1}^{I} \frac{r_i(p_i)}{\psi_i}) \tag{4.49}$$

$$= \sum_{i=1}^{I} K_i - C(\pi) + \lambda(\pi - \frac{\sum_{i=1}^{I} K_i}{p}) \tag{4.50}$$

where λ is the lagrangian multiplier. The solution to this problem satisfies the following Karush-Kuhn-Tucker (KKT) conditions:

$$\frac{\sum_{i=1}^{I} K_i}{p} = \pi \tag{4.51}$$

$$\tag{4.52}$$

and we obtain the signal price for every cluster as $p_i = \frac{\sum_{i=1}^{I} K_i}{\pi \times \psi_i}$.

If the VNFM sets the values of K_i such that $\sum_{i=1}^{I} K_i = C(\pi)$, then we have the simple form for the price signals where $p_i = \frac{C(\pi)}{\pi_i}$.

It is worth noting here the price adjustment may take place as a t$\hat{a}$tonnement-like procedure [63] in order to facilitate the convergence of the price signals and the demand between the VNFM and the cIM. During this process, the optimal price signals are calculated based on the monitored means of the demands of each cluster and a price signal is sent to the VNFM. In turn, the demands of the CPU might be adjusted by the VNFM until an equilibrium state is rearched when the prices reflect the actual supply and demand. At each step, the cIM calculates an adjusted price as follows:

$$p_i \leftarrow p_i(1 + \upsilon \times \min\{1, \frac{r_i - \pi_i}{\pi_i}\}) \tag{4.53}$$

where $0 < \upsilon < 1$ is the adjustment rate and r_i here is measured as the average demand during an allocation cycle.

The cIM calculates the price vector that maximizes its revenue during an operating time cycle $[0, T]$. This price vector is updated periodically to reflect any variations that have occurred to the demand pattern. As indicated before, we use the price vector to regulate the workloads: when the prices decrease, cVNFCs will be encouraged to increase the resource consumption [64]. At each period $t \in [0, T]$, the CPU pool size π is reserved to fulfill the total demands of the clusters i, which is calculated based on the current demand. A scaling-up in the CPU demand occurs at time t, if $\delta r_i(t) > 0$, while $\delta r_i(t) < 0$ indicates a scaling-down ($\delta r_i(t) = 0$ means no fluctuation).

4.4 Summary

In this chapter, we raised several optimization problems that may exist as the status of the VNFI resources alters and presented corresponding solution models to adjust the resource demand and meanwhile ensure service providers' profits. We first discussed the VNF profiling process in our experiment, illustrating the performance measurement manner. Then, we divided the optimal allocation problem of different cNFCs clusters into four sub-problems. Specifically, the first two sub-problems of optimal allocation involve homogeneous or heterogeneous loss functions that measure the performance losses between the

actual fluctuation and the reserved resource pool of different cNFCs. And corresponding allocation solutions were proposed using Schur-convexity properties to minimize the losses of these cNFCs clusters. And the subsequent problem considering the predefined losses threshold was solved to maximize the availability of cNFCs. Finally, a pricing scheme with the effect of resource adjustment was introduced to optimally price and regulate the allocated resources. In other words, while the pricing scheme for allocated resources helps maximize the cIM's revenue, it also influences the resource demands and allocation of each cluster in turn. In the next chapter, we will evaluate our proposed schemes and compare their performance against baseline methods.

Chapter 5

Experimental Analysis

In this section, we realize our proposed reservation scheme and various optimization solutions through JAVA programming language. The simulation process is conducted in Docker and Kubernetes. Ksniff and T-shark (the analyzer for capturing and analyzing packet data) are used to monitor the behaviour and performance of VNFs in different clusters, the collected data are further analyzed by corresponding JAVA projects to get final evaluation results.

5.1 Simulation setup

A lot of efforts have been spent in building the simulation environment since existing industrial NFV projects are mainly based on VMs. However, OpenStack (an open source cloud computing platform) as the mainstream of NFV industry offers an excellent branched project Tacker for NFV orchestration service, considering the feasibility of using containerization technology for virtualization. Tacker provides the option of Kubernetes VIM for installation combined with the existing OpenStack NFV platform. Consequently, users can deploy containerized VNF on the top of NFV infrastructure platform using Kubernetes VIM in Tacker. To implement our cIM, we similarly extend containerized Kubernetes VNFs in Kubernetes VIM of Tacker. Based on container topology and orchestration specification for cloud applications (TOSCA) templates offered by Tacker, we create cNFCs

through "openstack vnf" related Linux commands. To test cNFCs are running in Kubernetes environment, we can check their status by running "kubectl" (the command-line tool for managing Kubernetes clusters) related commands. Users also can scale VNF manually by running "openstack vnf scale" Linux command. However, the scaling types offered by OpenStack are mainly limited to scaling in and scaling out, which means fine-grained resizing operations (scaling up/down, resource reservation and allocation) are hard to realize on the side of Tacker or OpenStack NFV platform. One feasible solution for this problem is moving the resizing and allocation operations to the side of Kubernetes and Docker. Since Kubernetes renders *request* and *limit* properties of resources to end-users, we can specify how much CPU a container (cNFC) gets through these two flags. Accordingly, the fine-grained allocation and reservation can be achieved by dynamically changing the *request*/*limit* resources allocated to each cNFC.

Figure 5.1 depicts the architecture of our simulation environment. We use the client application programming interfaces (APIs) offered by Tacker and Kubernetes (e.g., fabric8io/kubernetes client, Kubernetes Java Client) to build Java projects for cNFC deployment, monitoring and resource allocation. Kuryr-kubernetes (a project for integrating Kubernetes with OpenStack networking) will be used as networking between containers and VMs, which increases the scalability.

In our simulation scenarios, each flow was served by multiple connected cNFCs hosted on different Kubernetes pods, the smallest deployable units of compute that can be created and managed in Kubernetes. To chain these cNFCs, we employed configMap (an API object to store separate configuration data) for inter-pod communication. In particular, we need to enter each VNF container using ./bin/sh bash script command and explore its configuration and working principle, so that we can ensure the right configMap can be created and adopted by the container when communicating with other pods.

We compared the performance of the deployed VNFs using our proposed RRM against the default configurations of the system. The optimization solutions are also implemented, testing their effectiveness and efficiency compared with other schemes. To measure the performance of our reservation schemes, we use the following basic metrics: CPU availability

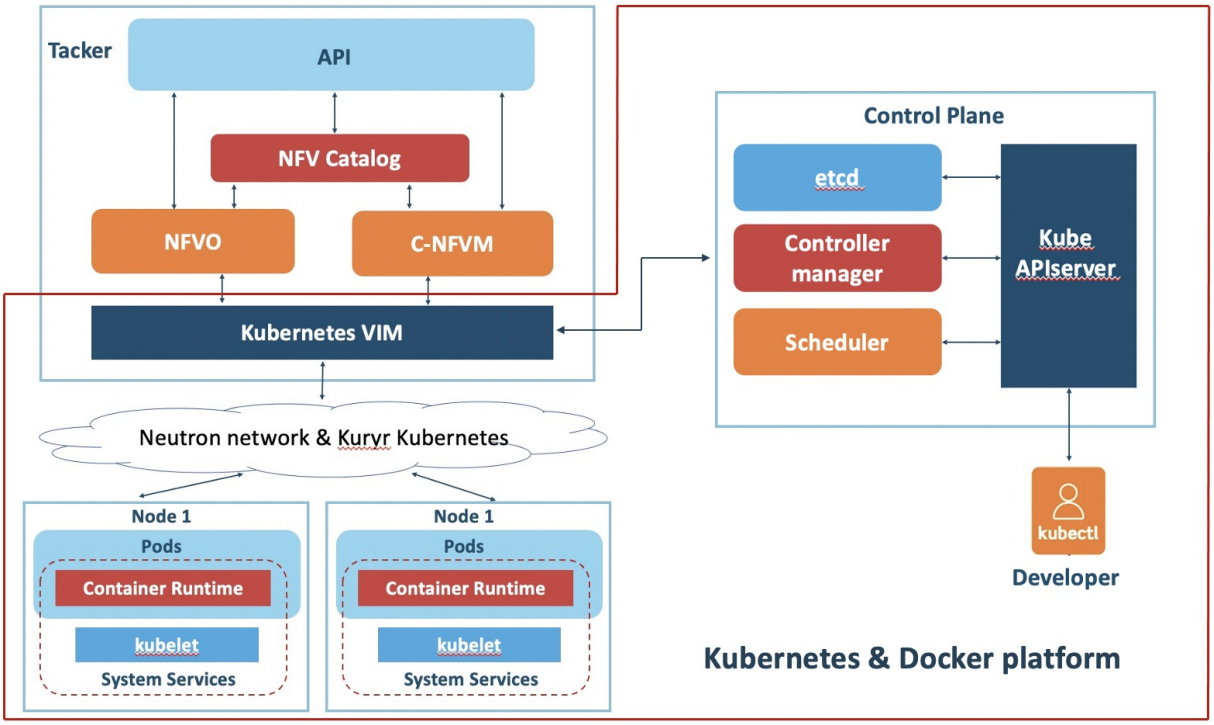


Figure 5.1: The architecture of simulation environment

per a single cNFC, actual CPU consumption and the served flows' performance in terms of the throughput (the number of processed frames per unit time) and delay (the round trip time of these frames). More precisely, Ksniff, a kubectl plugin that utilizes tcpdump and Tshark to start a remote capture on any pod in Kubernetes cluster, is used for the monitoring of workflows of specific cNFCs. The throughput and delay of each cNFC can be recorded in a txt file at the specified speed. As for the monitoring of resource consumption, we choose to implement a special metric "kubectl top" through Java project, which is a special Kubernetes metric collected by the lightweight metrics-server and is exposed via the metrics.k8s.io API. It's worth nothing that we also tested other parameters (e.g., the total revenue) as per different optimization purposes of the proposed schemes. All the experiments were performed on an Ubuntu-based computer with 3.40GHz $\times$ 8 Intel Core i7 processor and 5.7 GB memory. Next, we first evaluate the performance of our RRM.

5.2 Performance evaluation of resource allocation

We set the number of CNFs clusters, $I = 3$, and allowed $1 - \alpha$ to vary in the range $[0.99, 0.9999]$. Similar to the work in [65], we divide the availability requirements of different clusters into three levels:$\{0.99, 0.999, 0.9999\}$. As shown in Fig. 5.2, the first cluster of cNFCs contains two types, namely, an intrusion detection system (IDS) referred to as a frontend-vnf and an Nginx cNF, which is referred to as the backend-vnf. It should be noted that, for simplicity, we achieved homogeneity of the resource demands for the cNFCs in each cluster by using similar workloads for each component. The second cluster of cNFCs, has its availability set to $1 - \alpha = 0.999$. As shown in Fig. 5.2 it contains three types of cNFCs, namely, firewalls, IDSs and Nginx. Finally, the last cluster is created such that $1 - \alpha = 0.9999$ and is composed of independent firewall instances.

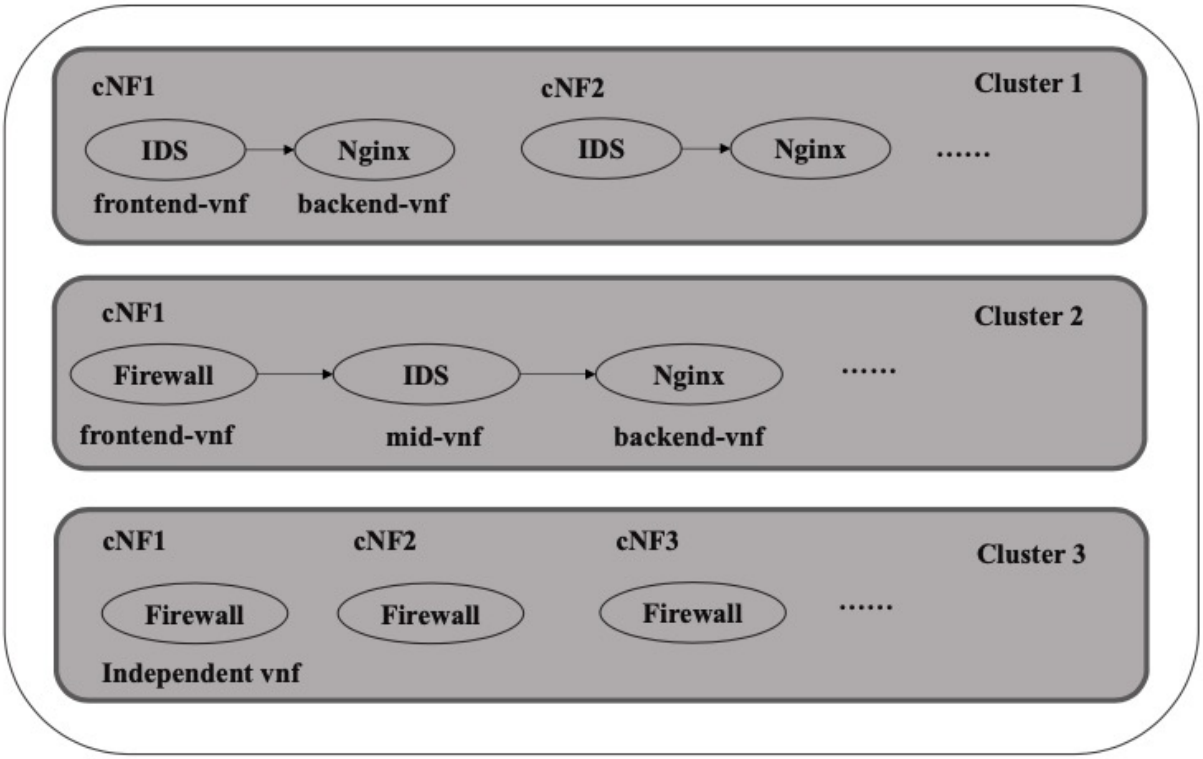


Figure 5.2: cNFCs in different clusters.

We monitored the behaviour of the cNFCs for a duration of thirty minutes and examined how the RRM reservation mechanism ensures the reservation of sufficient CPU resources

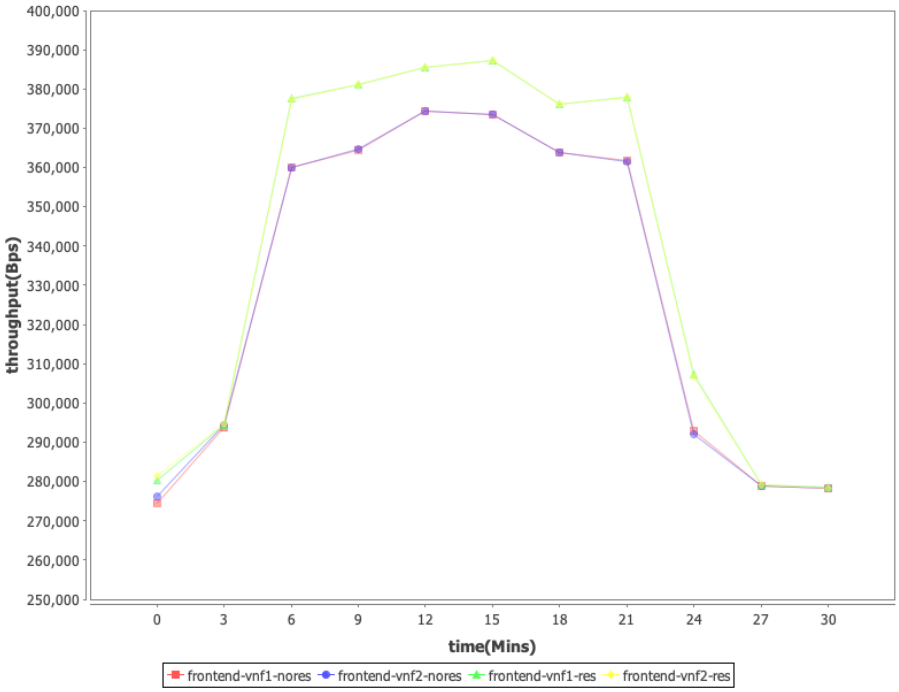
as the workflows of the cNFs fluctuates around their means.

5.2.1 Efficiency of resource allocation and sharing

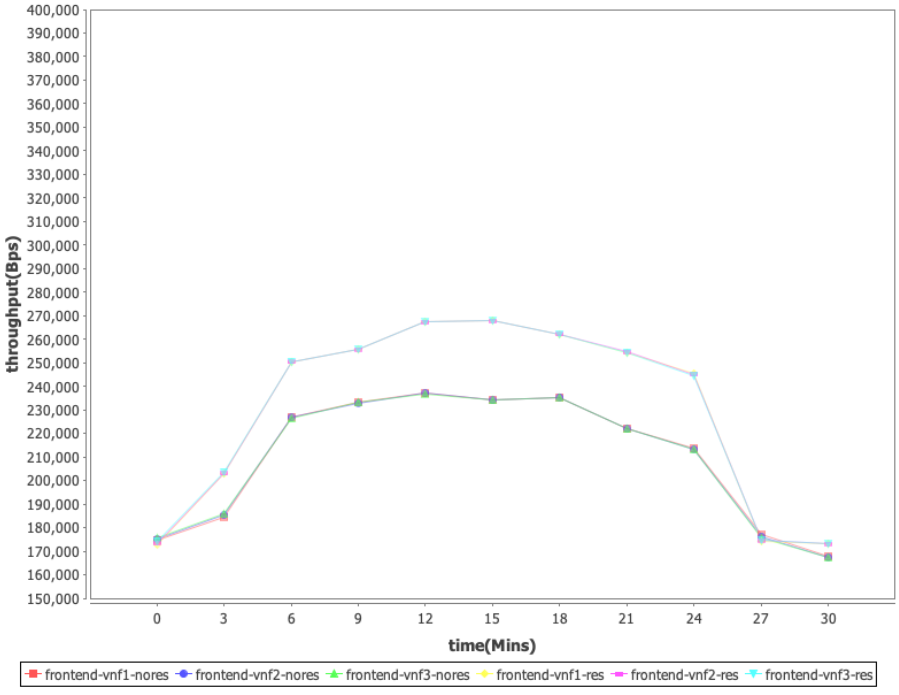
Figs. 5.3 and 5.4, respectively, plot the average experienced throughput and delay by the flows served by one type of cNFCs in each of the clusters. The plots marked with (res) and (nores), respectively, mark the performance of the proposed reservation scheme against that of the default system configuration with no reservation. The corresponding resource consumption for different cNFs is shown in Fig. 5.5. As the number of served workflows increases (e.g., periods 0 mins to 6 mins in Fig. 5.3a), the cNFC instances are allocated sufficient resources by the RRM. Hence, they achieve better performance than those without reservation. The difference in performance increases when the number of cNFCs sharing the resources increases to three in the second cluster. For example, the difference in throughput for two cNFCs per SFC per cluster is between 10,000Bps and 20,000Bps. On the other hand, in the second cluster, the difference increases sharply in the range of 20,000Bps and 30,000Bps. The same trend can be seen with the delay which indicates that the performance of the cNFCs is significantly improved using our reservation mechanism.

Since the mechanism is derived from the law of large numbers, it's necessary to test the performance of the mechanism when the number of cNFCs constituting each cluster is relatively large. To achieve this goal, we build different implementation scenarios where the number of cNFCs composing the cluster increases from 5 to 30. The partition granularity is set to 5, indicating the performance of cNFCs is monitored in 6 experiments as the cluster is comprised of 5, 10, 15, 20, 25 and 30 cNFCs respectively. We test the average throughput, delay of cNFC for a monitoring duration of 30 minutes with the unchanged high workload. For example, Figs. 5.6 and 5.7 respectively plot the throughput in 30 minutes when 5 and 10 cNFCs compose the pool. It's apparent that the fluctuation is pretty slight due to the continuous high workload we assign to these cNFCs.

Based on the simulation results above, we can find the throughput is nearly the same in 30 minutes because of the unchanged high workload. Thus we further calculate the mean

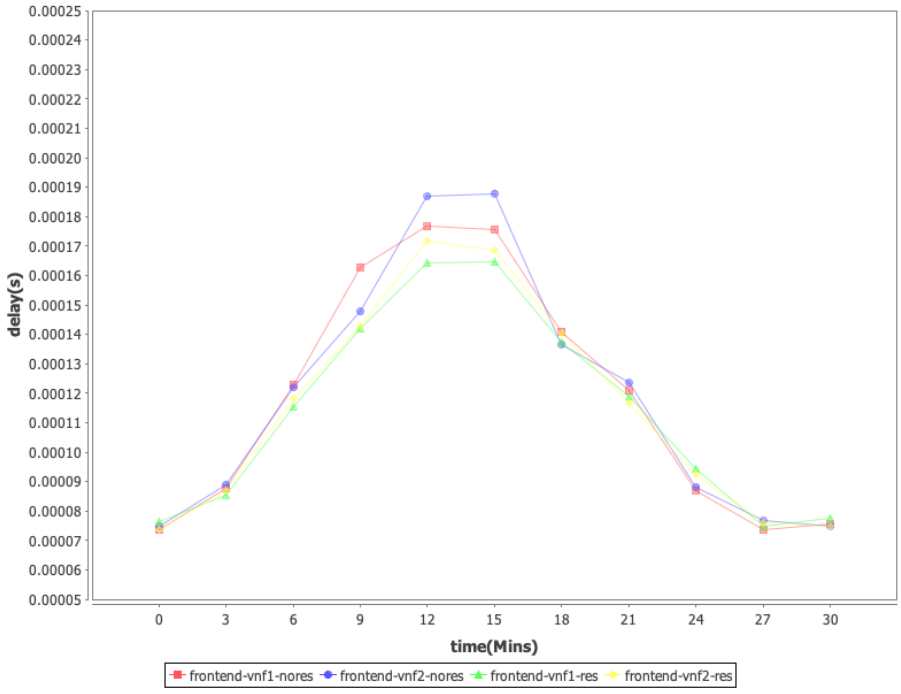


(a) cNFCs in Cluster 1

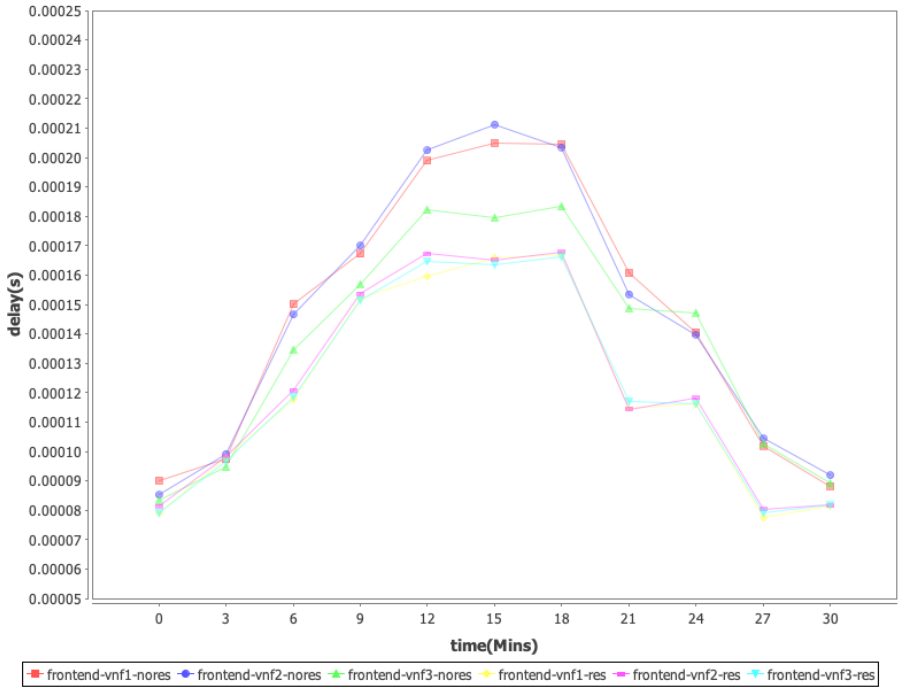


(b) cNFCs in Cluster 2

Figure 5.3: Average flow throughput in cNFCs in the first and second clusters without (nores) and with (res) resource reservation

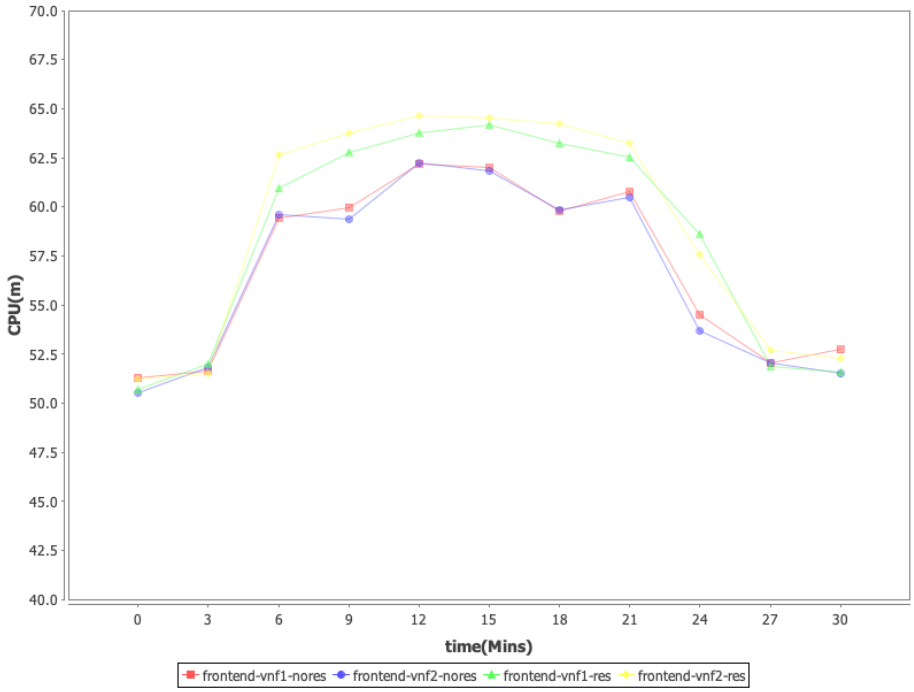


(a) cNFCs in Cluster 1

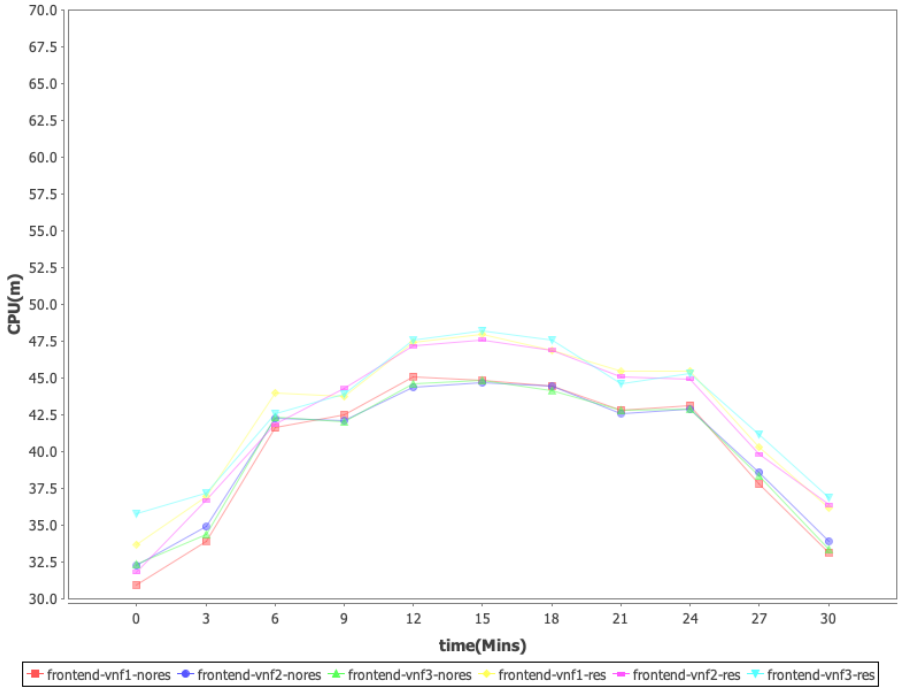


(b) cNFCs in Cluster 2

Figure 5.4: Average flow delay in cNFCs in the first and second clusters



(a) cNFCs in Cluster 1



(b) cNFCs in Cluster 2

Figure 5.5: Average resource consumption by cNFCs in the first and second clusters without (nores) and with (res) resource reservation

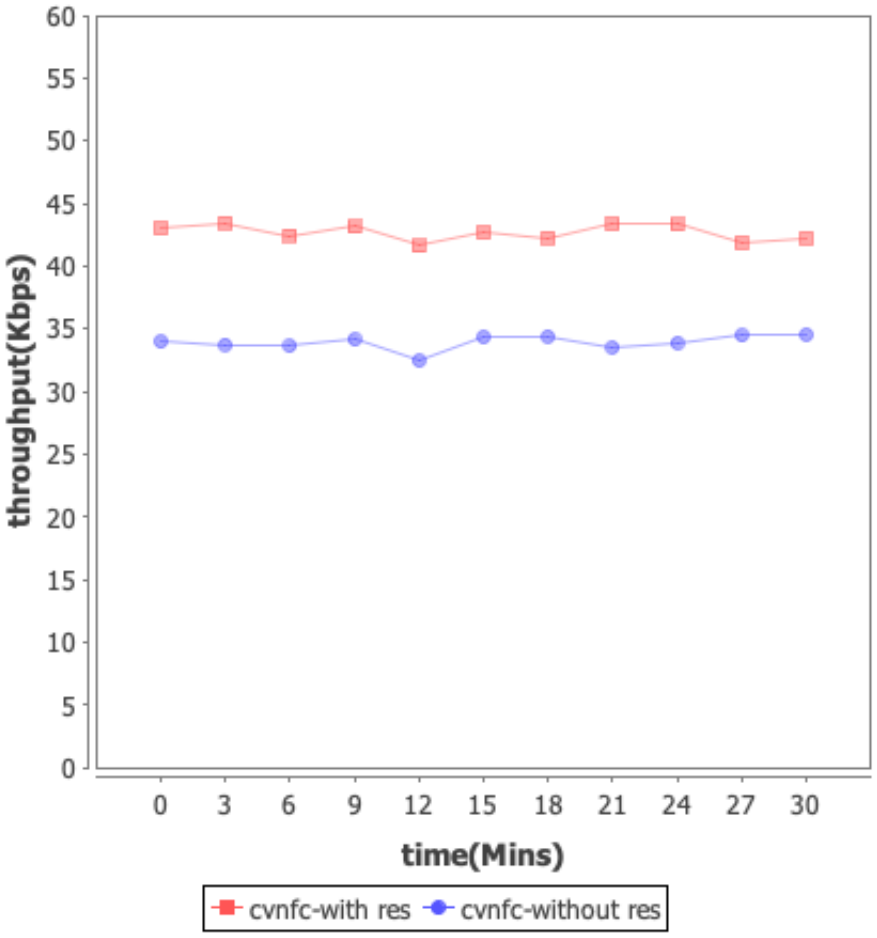


Figure 5.6: Average throughput of 5 cNFCs composing the cluster in 30 minutes without and with resource reservation

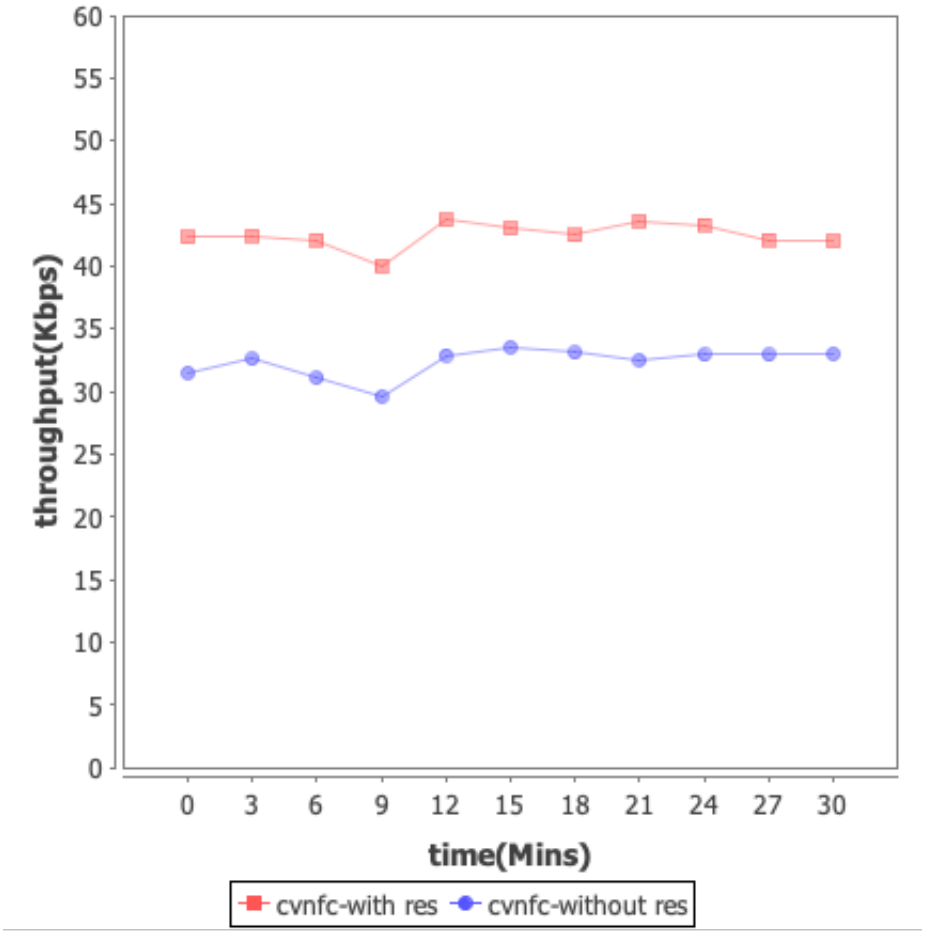


Figure 5.7: Average throughput of 10 cNFCs composing the cluster in 30 minutes without and with resource reservation

of throughput in 30 minutes, which returns a scalar value of average throughput. Then the average throughput/delay/resource consumption as the number of cNFCs increases can be derived. And the results are shown in Figs. 5.8, 5.9 and 5.10. It should be noted that each point represents the average throughput/delay/resource consumption of each cNFC in 30 minutes. When the number of cNFCs of the cluster increases, the difference of performance between cNFCs with reservation and without reservation increases correspondingly. And with reservation, even when the number of cNFCs increases, the performance can still be guaranteed (nearly unchanged) while the performance degrades proportionally without reservation. The incremental difference demonstrates that our scheme has superiority in large-scale deployment, outperforming the resource allocation without reservation.

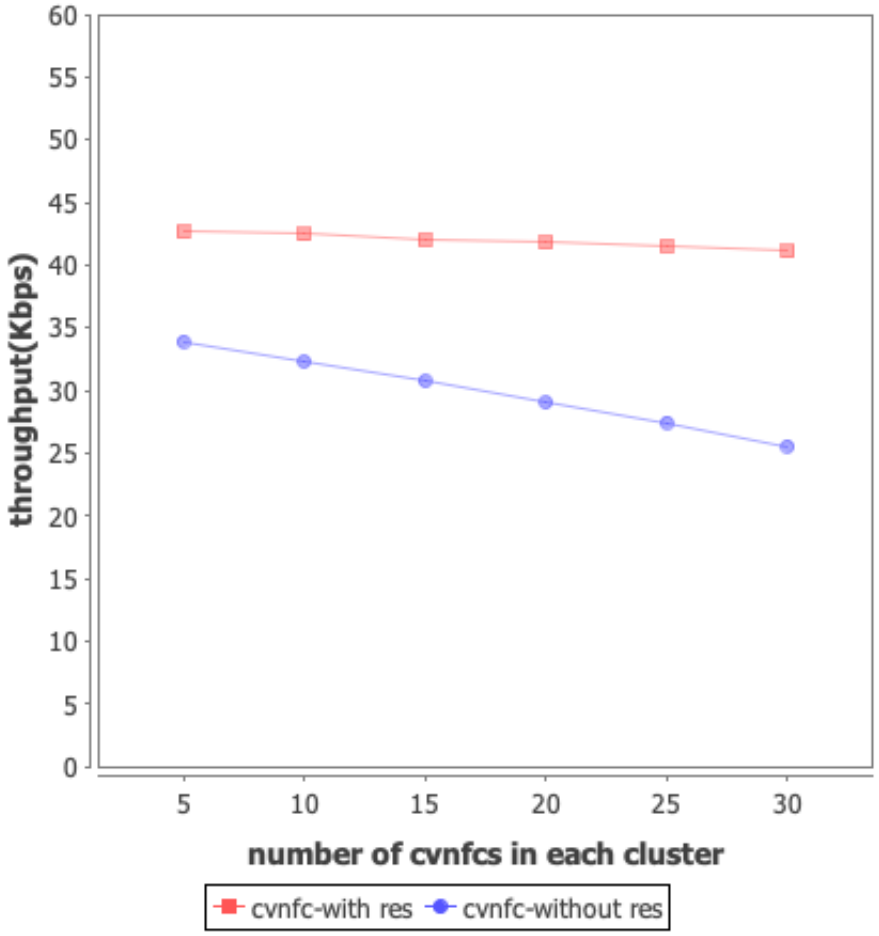


Figure 5.8: Average throughput of cNFCs in 30 minutes as the number of cNFCs composing cluster increases

Next, we examine the efficiency of our scheme when multiple clusters share the reserved

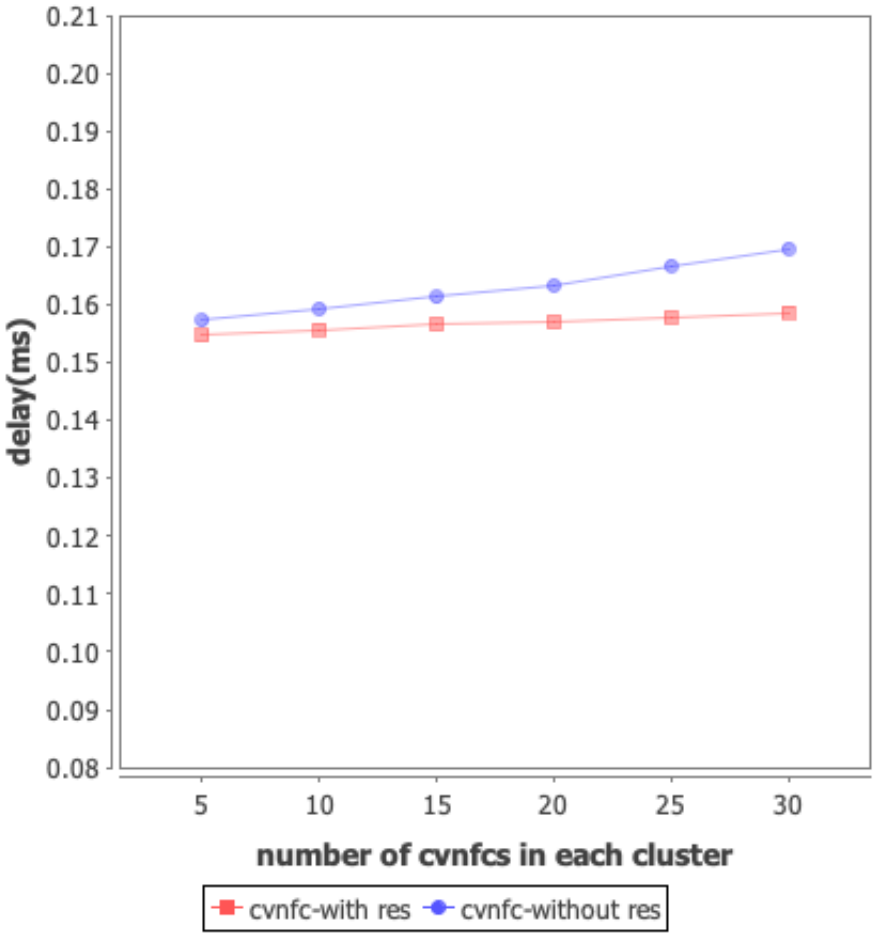


Figure 5.9: Average delay of cNFCs in 30 minutes as the number of cNFCs composing cluster increases

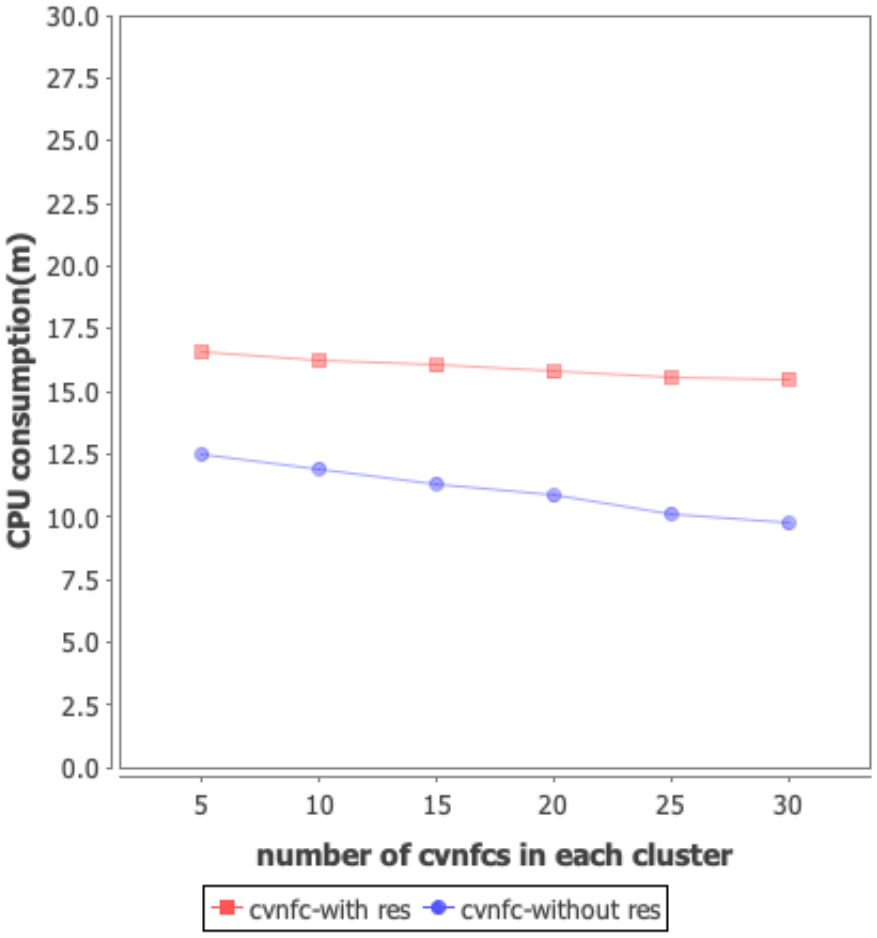


Figure 5.10: Average resource consumption of cNFCs in 30 minutes as the number of cNFCs composing cluster increases

resource pool and experience different workload patterns. Figs. 5.11, 5.12, respectively, depict the average throughput and delay of sampled cNFC in different clusters. Fig. 5.13 plots the corresponding CPU consumption of these cNFCs. It is worth nothing that the workload fluctuations are quite different due to the heterogeneity of clusters. However, as the workloads increase, cNFCs in clusters sharing the resource pools achieve a much better performance than the case without resource reservation. The CPU utilization of the infrastructure is shown in Fig. 5.14. It's obvious that while the performance of every cluster is improved significantly, our reservation mechanism consumes only a small amount of total CPU resources as depicts by the green areas in the pie chart.

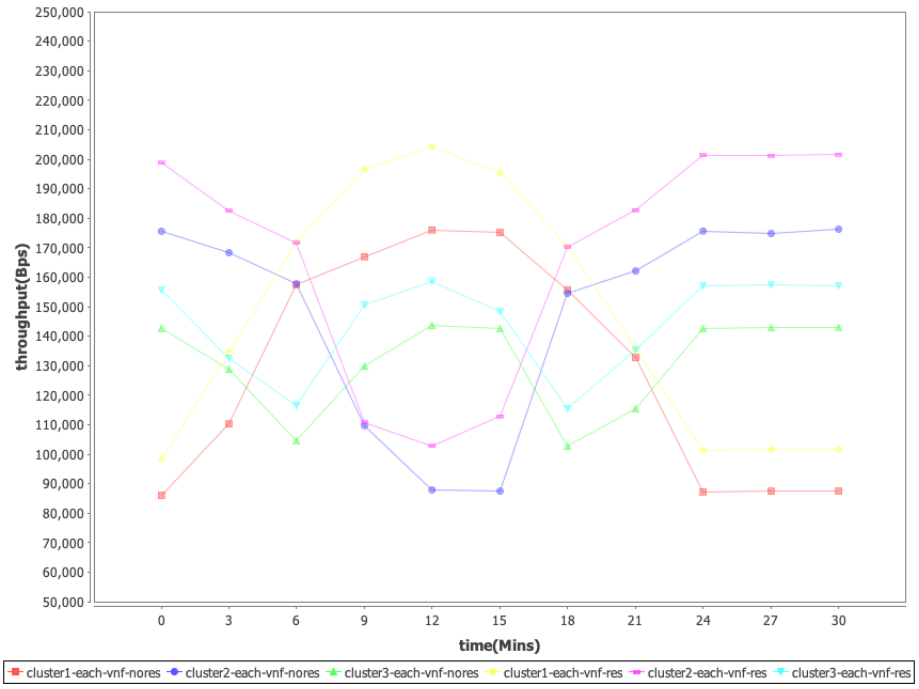


Figure 5.11: Average throughput of different cNFCs in multiple clusters with (res) and without (nores) shared resource pools

5.2.2 Accuracy of resource allocation by the RRM

The previous experiments demonstrated that our reservation scheme can effectively provide sufficient resources to different cNFCs sharing the same resource pool. Next, we examine whether any additional resources could have been saved. In other words, we investigate

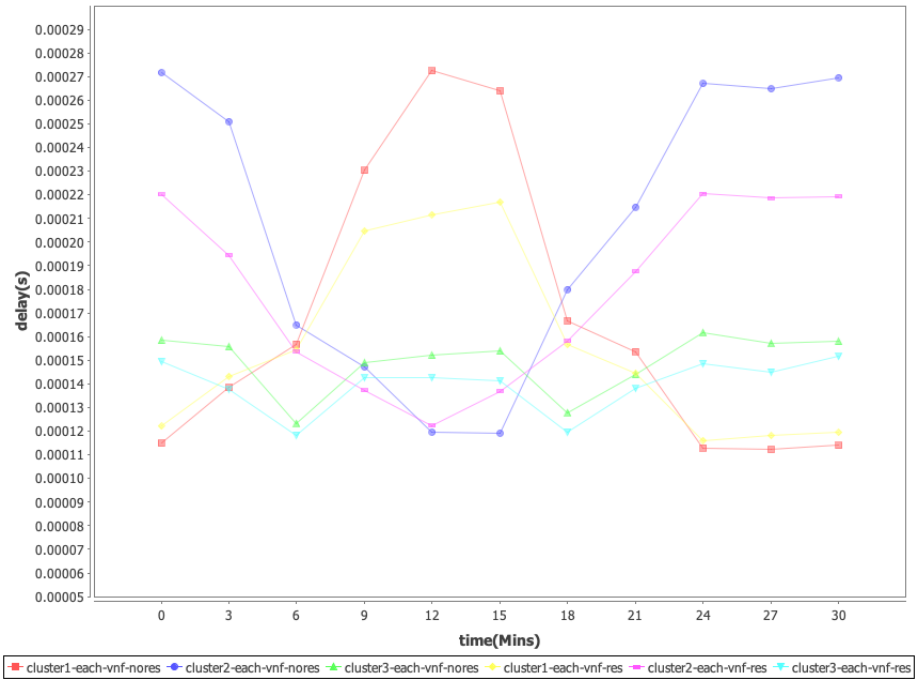


Figure 5.12: Average delay of different cNFCs in multiple clusters with (res) and without (nores) shared resource pools

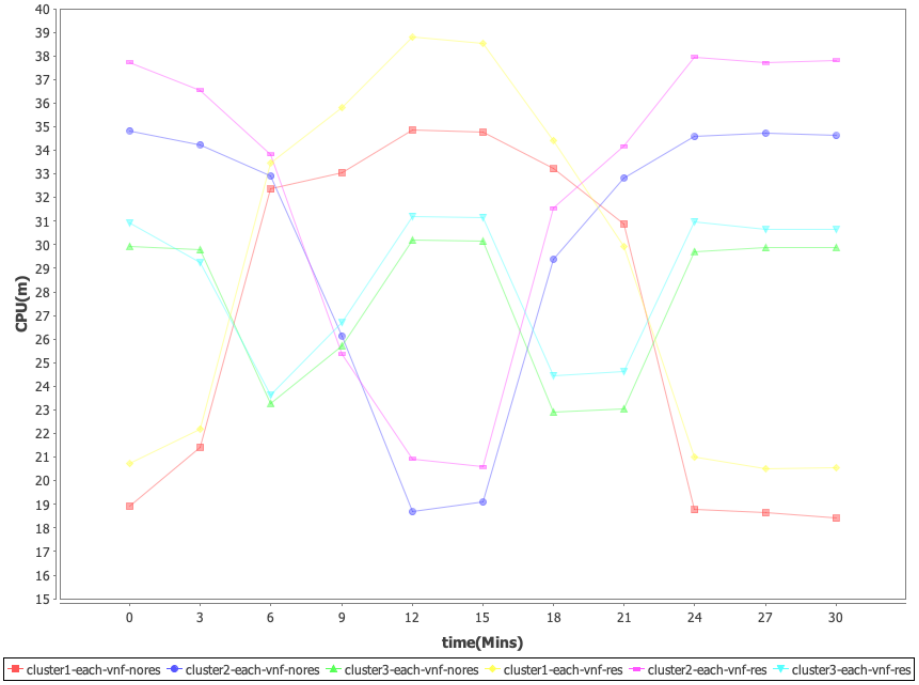


Figure 5.13: CPU consumption by different cNFCs in multiple clusters with (res) and without (nores) shared resource pools

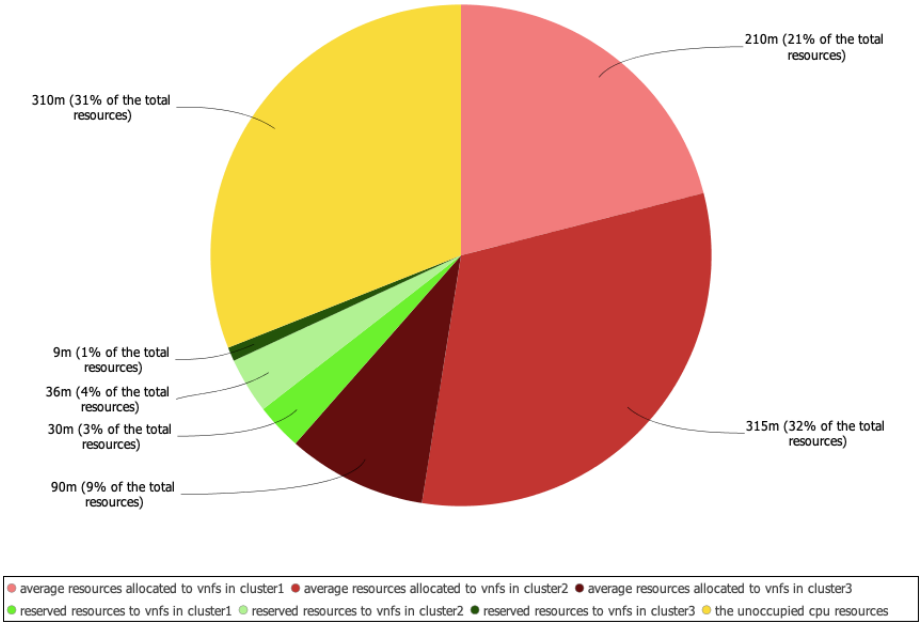


Figure 5.14: Resource utilization pie chart of the whole CPU

whether our reservation scheme tends to over-provision resources or not. To achieve that, we measure the performance of the cNFCs as they are allocated different amounts of resources and then compare that performance against that when the cNFCs' resources are allocated via our RRM. We examine the performance of two cNFCs, denoted, frontend-vnf1 and frontend-vnf2, when they are allocated different CPUs. More precisely, we evaluated their performance when they are allocated 68 milli (68m) CPU, each, as calculated by our proposed RRM against their performance when they are allocated 60, 65, 70, 75 and 100m CPUs, each. Figs. 5.15 and 5.16, respectively, depict the throughput and delay of the two cNFCs with varying allocated CPUs as the workloads increase gradually, up to 12mins and then start to decrease again. The corresponding resource consumption for these cNFCs is shown in Fig. 5.17. When the allocated CPUs are smaller than the RRM's reserved 68m CPUs, the throughput (and delay) decreases (increases) significantly as the workload increases when compared to the performance of the instances with 68mCPUs. On the other hand, the figures demonstrate that allocating any CPUs exceeding the 68mCPUs selected by the RRM does not result in any significant performance enhancement. In other words, any additional resources allocated to the instances are wasted.

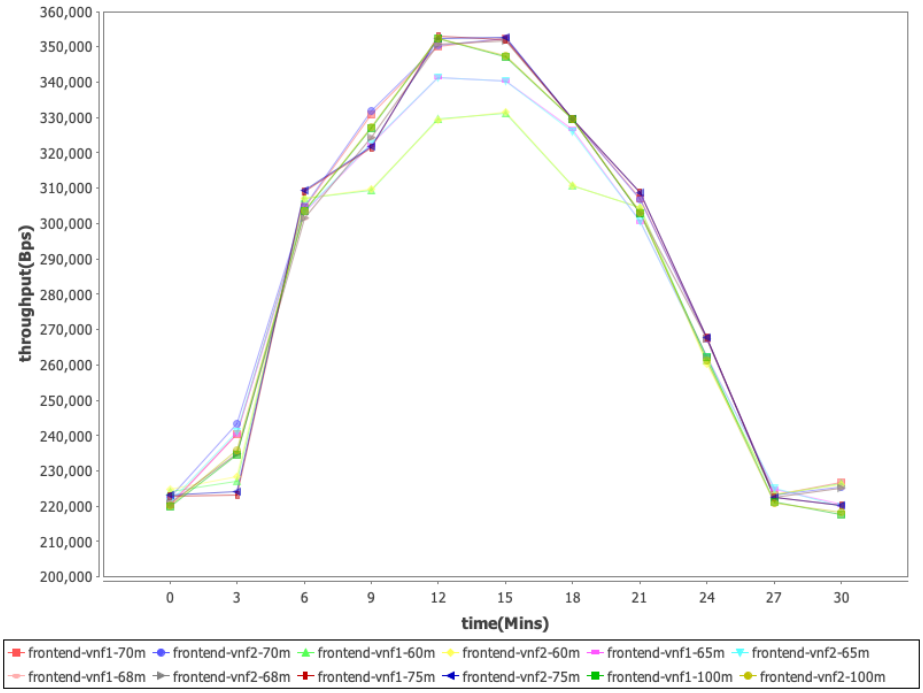


Figure 5.15: Average Throughput of different cNFCs when allocated different CPUs

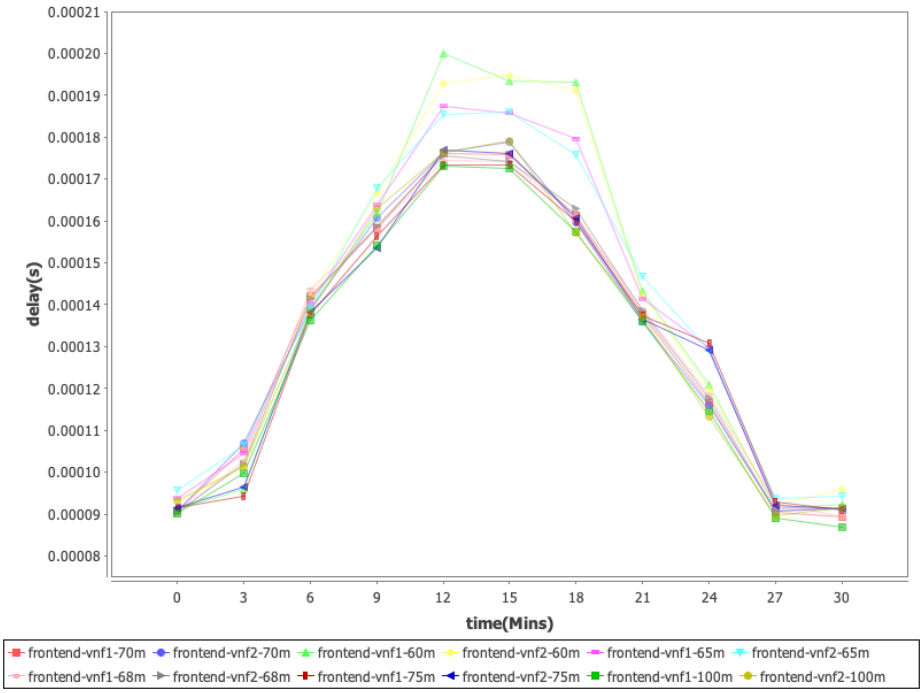


Figure 5.16: Average delay of different cNFCs when allocated different CPUs

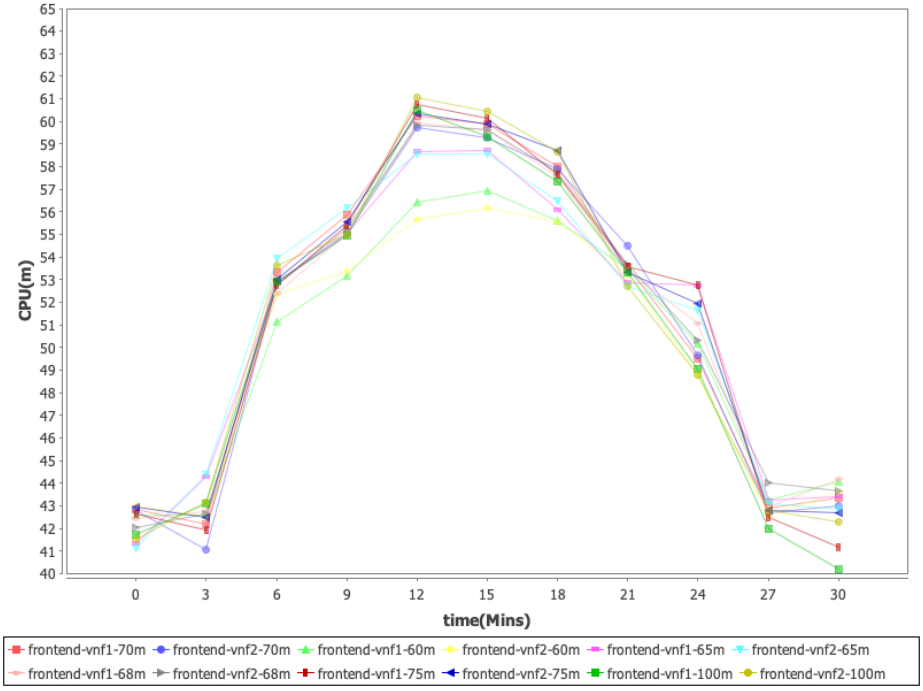


Figure 5.17: CPU consumption by different cNFCs

5.3 Performance evaluation of different optimization problems

After demonstrating the efficiency and accuracy of our reservation scheme in different simulation scenarios, we move on to the next part, evaluating the performance of VNF clusters which apply different optimization solutions for corresponding optimization problems in chapter 4. To verify our solutions are effective to these optimization problems, we generally use three VNF clusters in the implementation of each optimization scenario except for the pricing model as one cluster is enough to demonstrate its effectiveness. Similar to the aforementioned experiments, we use Java to build several projects, realizing these optimization solutions. We monitored the workload traffic and resource consumption of different clusters for a long period, collecting the data for statistical analyses. In the following subsections, we illustrate and evaluate our simulation results of different optimization solutions in detail.

5.3.1 Performance evaluation of optimal resource allocation given the availability

This simulation is conducted in a situation where a specific level of availability is given. In other words, cNFC clusters are assigned with a pre-determined small probability α that the total resource demands exceed the total reserved pool. The availability $1 - \alpha$ of the clusters, in turn, is also a pre-determined number. Similar to the experiments in section 5.2, we specify the overall availability demand of the three clusters as $1 - \alpha = 0.99$.

Figure 5.18 depicts the results of our solution to the minimization problem of weighted distances, compared to other representative allocation schemes. More precisely, allocation scheme 2 is calculated by our optimized solution. Scheme 1 and scheme 3 are representative counterexamples. In particular, allocation scheme 1 represents equal distribution. The allocation scheme 3 is in close proximity to scheme 2 but gets a slightly larger amount of weighted distance (3.64) than scheme 2 (3.61). It's apparent that the sum of weighted distance is minimum in scheme 2 as 3.61, indicating that our solution reaches the goal of minimizing the sum of the distance between actual resource demand and reserved resource pool size of each cluster while guaranteeing the required availability level. In general, the results demonstrate that our allocation scheme is in the closest proximity to the actual resource demands, avoiding unnecessary energy waste. It uses the minimum amount of resources for reservation while ensuring the specific availability.

5.3.2 Performance evaluation of optimal resource allocation given the expected fluctuation

On the contrary, this simulation is conducted in a situation where the sum of the expected weighted distance is given. Accordingly, the whole resource pool consisting of three clusters is specified with a pre-determined overall weighted distance ψ between the real resource consumption and the reserved pool size of each cluster. Different allocation schemes result in different availability or elasticity level of each cluster. We set ψ as 1 CPU (1000m CPU) and calculate the elasticity level/availability of the whole resource pool in accordance with

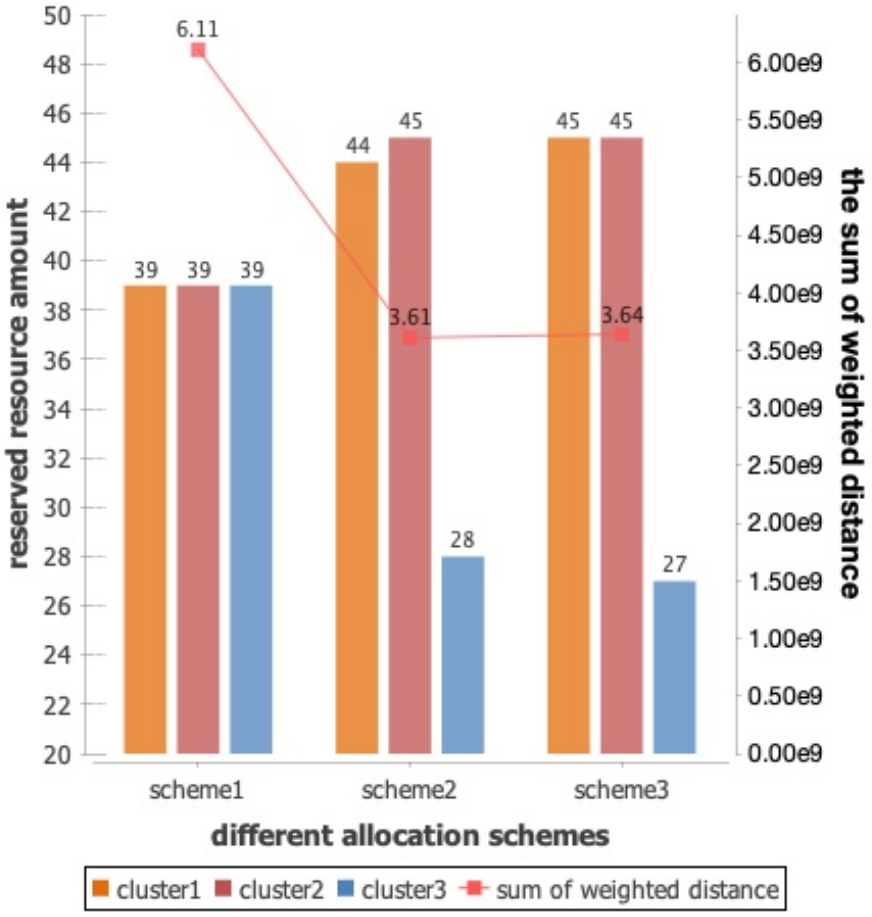


Figure 5.18: Different resource allocation schemes of clusters and corresponding sum of weighted distances

different allocation schemes.

Figure 5.19 depicts the testing results of our solution to the minimization (maximization) problem of the probability α $(1-\alpha)$, compared to other allocation schemes. The allocation scheme 2 is derived from our optimization solution, while scheme 1 and scheme 3 are representative counterexamples. It is worth nothing that all the listed schemes satisfy the precondition that the overall squared distance ψ is equal to 1000m CPU. As shown in the figure, the overall availability of three clusters is maximum in scheme 2 as 0.902, indicating that our solution reaches the goal of maximizing the availability while guaranteeing the required sum of the weighted squared distance. In other words, it proves that the optimized allocation scheme reaches the maximum elasticity level when the expected deviation between the actual fluctuation and the reserved pool is pre-defined, further indicating that our solution utilizes the available resources optimally for the acquisition of higher QoS.

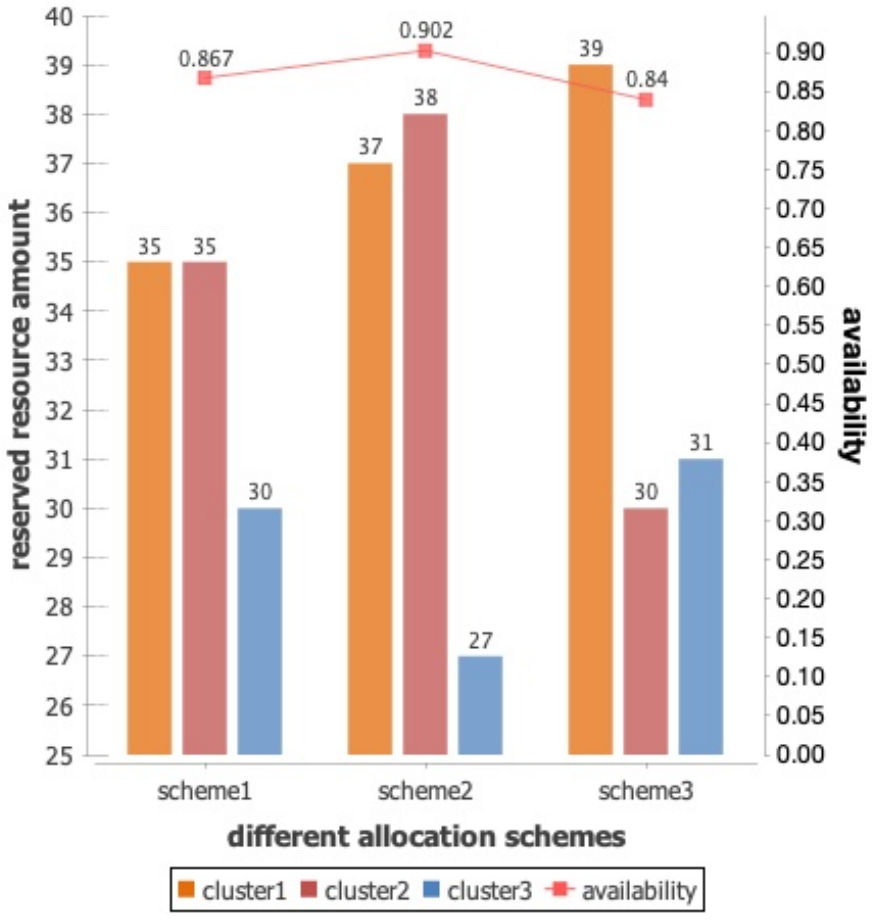


Figure 5.19: CPU consumption by different cNFCs

5.3.3 Performance evaluation of available resource allocation given penalties

In this simulation, we consider the available resources are scarce, which may not satisfy the required amount of resources for reservation. Accordingly, we set the amount of available resources as a pre-determined value: 110 milli CPU. The price scheme of each cluster is also pre-defined as $ 0.8 per n CPU (1n=0.000001m) for cluster1, $ 1.0 per n CPU for cluster2 and $ 0.5 per n CPU for cluster3. The penalty function $\wp(.)$ used in the simulation is $(x_e^{c,i}(t) - y_e^{c,i}(t)) * p_e^{c,i}(t)$, where $x_e^{c,i}(t)$ denotes the required resource pool size of cluster i derived from our reservation algorithm, while $y_e^{c,i}(t)$ is the actual available resources allocated to cluster i, and $p_e^{c,i}(t)$ is the price function of cluster i.

Figure 5.20 depicts the results of our allocation solution with penalty function compared to other representative counterexamples of allocation scheme. The allocation scheme 2 is calculated by our optimization solution for total revenue. Scheme 1 and scheme 3 are the counterexamples. It should be noted that the sum of allocated resources of these three clusters in each scheme is equal to 110m CPU due to the limited available resources. More precisely, resources are nearly equally allocated to each cluster in scheme 3, while allocation scheme 1 is similar to scheme 2. However, it's obvious that the total revenue of the three clusters is maximum in scheme 2 as $ 173.71, while other schemes obtain less revenue ($ 165.42 for scheme 1 and $ 134.56 for scheme 3). This result strongly proves that our allocation solution maximizes the profits of Cloud Service Providers (CSPs) with the existence of penalties which is caused by the failure of meeting the required elasticity service level. Besides, our scheme successfully prioritize clusters with higher availability level under the constraint of the scarce available resources of compute node.

5.3.4 Performance evaluation of pricing model

To test the performance of our pricing model, we categorize three resource demand states that the clusters may experience based on historical monitoring and calculation. More specifically, $state1$ namely low-demand state with $\mu = 18.83$ mCPU, $state2$ namely mid-

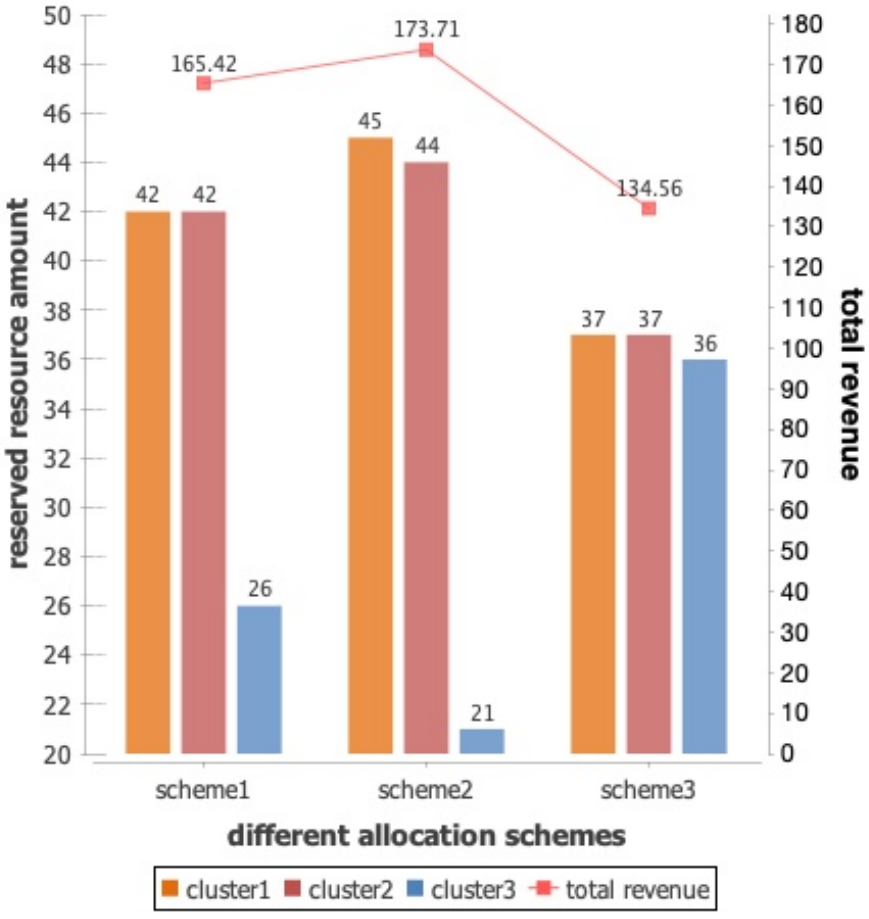


Figure 5.20: CPU consumption by different cNFCs

demand state with $\mu = 28.77$ mCPU and $state3$ namely high-demand state with $\mu = 33.81$ mCPU. As mentioned in chapter 4, the prices should be bound by maximum and minimum values. However, in the simulation, we specify the possible price range as a set of discrete scalar values $P = \{1.5, 1.6, 1.7\}$ for the sake of simplicity. Thus $p_e^{c,i}(s_t) \in \{1.5, 1.6, 1.7\}$, where $s_t \in \{state1, state2, state3\}$. The operating time cycle $[0, T]$ is defined as $[0, 12mins]$, where each period t is equal to 3mins and $t \in [0, 12mins]$. The Markov state transition matrix between states is deduced from historical monitoring and collected resource consumption data. Besides, the transition matrix also takes the fact that price influences the workload to the cloud into account. It's worth nothing that we list and evaluate the pricing process for one cluster in the first 12 minutes here since the principle and conclusion can be generalized to other clusters for every 12 minutes recursively and similarly.

Figure 5.21 plots the results of our optimized pricing scheme in the first 12 minutes compared to other representative pricing schemes. Similarly, pricing scheme 2 is derived from our pricing solution. Scheme 1 and scheme 3 are representative counterexamples. To be more specific, pricing scheme 1 is similar to scheme 2 with a slightly different price for state1. As for scheme 3, each state is priced at \$1.7 per mCPU uniformly. The simulation results show that our pricing scheme obtains the maximum total revenue (211.71) than all the other schemes, which further demonstrates that our dynamic pricing model provides the optimal price vector for possible states, maximizing the CSPs' expected long-term profits based on workload prediction while guaranteeing availability.

5.4 Summary

In this chapter, we first describe the simulation setup process, depicting the experiment environment of Tacker combined with Kubernetes and Docker. The measurement methods for workflow and resource consumption are also listed. Then, we illustrate the structure and composition of our cNF clusters. The experiment results are presented in the two subsections respectively, proving the efficiency and accuracy of our RRM. In particular,

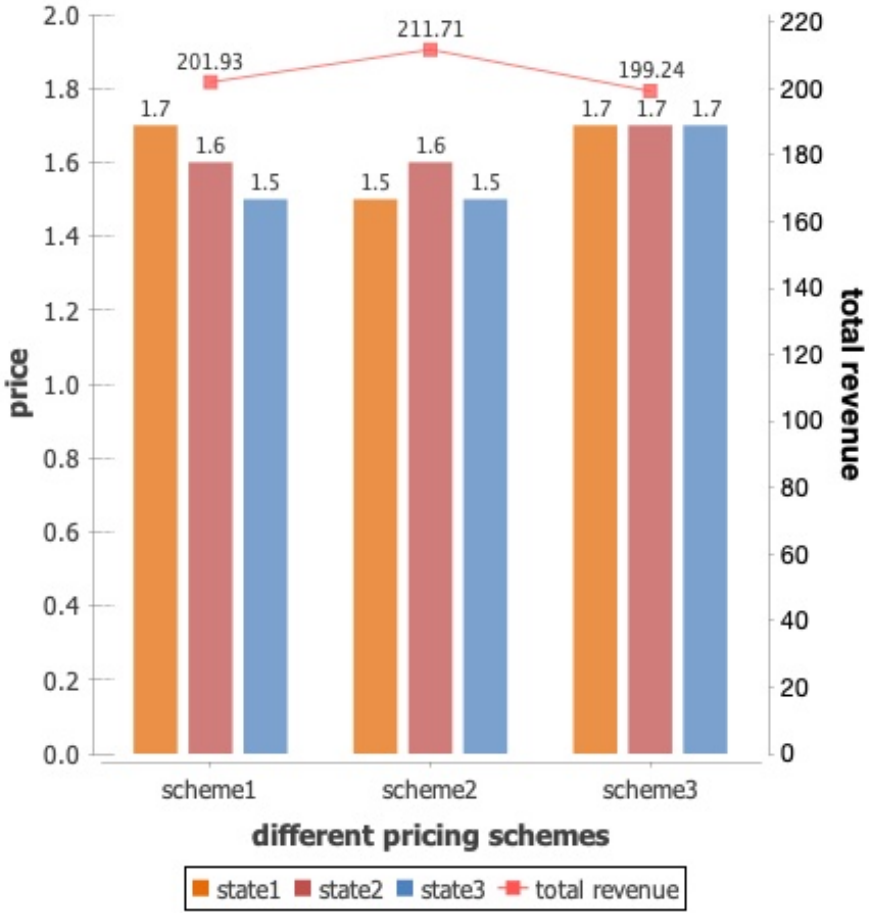


Figure 5.21: CPU consumption by different cNFCs

we demonstrate the outstanding performance of our scheme in large-scale deployment. The simulation results of different optimization solutions to corresponding problems are evaluated in the following section. We conduct four experiments as per the respective goals of these optimization problems. The results show that all of our optimization solutions are effective and efficient, making full use of the available resources to obtain high performance and profits while considering various limitations and obstacles that may exist in resource reservation.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a novel design of a containerized infrastructure manager(cIM) as one of the main building blocks within the ETSI VNF architecture. In contrast to existing schemes, we focused on the problems of resource scarcity and resource availability during workload fluctuations of the network edge (NE). The main component of the proposed cIM can accurately calculate the needed buffer resources for hosted containerized virtual network function components (cVNFCs) using concepts of risk pooling. More precisely, given a desired resource availability guarantee at the individual component, or at the level of a single cluster or a set of clusters of cNFCs, the proposed scheme allocates a sufficient quota that ensures resource demands are met during an increase in the served workloads. For the situation where multiple cNFC clusters composing the resource pool, we also illustrate different weighted resource allocation schemes in particular. Besides, we take the availability of cVNFC clusters, the expected deviation between the actual resource consumption and the reserved pool, the penalties due to limited available resources, the pricing influence on the total revenue into consideration. Based on all these factors, we describe different problems that may arise in the reservation with the corresponding solutions optimizing the performance and/or profits. It's noted that to formulate the pricing scheme, we apply workload prediction in our model based on Markov chain theory, maximizing the profits

of Cloud Service Providers (CSPs). The first part of the simulation results demonstrates that our proposed reserving resource approach achieves better performance compared to other baseline solutions in terms of resource availability and achieved workflow delay and throughput while satisfying an economical and optimal use of resources. The corresponding simulation results demonstrate that all the presented various solutions achieve their respective optimization purposes in different hypothetical scenarios.

6.2 Future work

This thesis is based on specific scenarios and related assumptions, and there are many more that can be studied in-depth and optimized. In addition, with the continuous development of network edge computing technology, network function virtualization technology faces many new scenarios and new challenges, which are worthy of further study. The shortcomings of the thesis and the future research work directions are summarized as follows.

Compatibility problem

The basic usage scenario presented in this thesis is containerized NFV, neglecting the existence of VM-based NFs that are also widely deployed for commercial uses. Thus the compatible problems between our model and existing NFV projects should be emphasized and solved to guarantee practicability. And one of the promising research directions is integrating our model with VM-based VNFs.

Resource limitation problem

In our reservation scheme, the availability of resources is discussed when we propose the penalties for those cVNFCs failing to meet reservation demands. However, most of our research is on the assumption that the available resources of compute nodes are sufficient, which may be violated in ultra-dense networks with enormous VNFs and service requests. Consequently, it is critical to assess the availability of resources for reservation in large-scale networks. And we can survey the severity of resource scarcity in the future.

Scaling problem

Most existing NFV projects achieve elasticity through horizontal scaling and vertical scaling. Our RRM as the complement to these scaling methods can be further studied and integrated with them to achieve VNF's efficiency. For example, a scheme can be proposed and improved, such that the temporary need of excess resources can be immediately met through our RRM in most cases, without horizontal scaling or vertical scaling. But when the demand increases to an extreme value that exceeds the threshold for reservation, or the high resource demand is durable, the scheme will switch to horizontal or vertical scaling based on different requirements. Specifically, to achieve this, it's promising to add neural network perceptrons into the resource allocation scheme, by which the cIM can judge and make the right decision about whether reserving or scaling.

References

[1] What's the Diff: VMs vs Containers. [Online]. Available: https://www.backblaze.com/blog/vm-vs-containers/.

[2] Docker overview. [Online]. Available: https://docs.docker.com/get-started/overview/.

[3] Kubernetes vs. Docker: A Primer. [Online]. Available: https://containerjournal.com/topics/container-ecosystems/kubernetes-vs-docker-a-primer/.

[4] CILLONI Marco. *Design and implementation of an ETSI Network Function Virtualization compliant container orchestrator*. PhD thesis, 2016.

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

[6] ETSI Network Function Virtualization. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv.

[7] N. Slamnik-Kriještorac and J. M. Marquez-Barja. Demo Abstract: Assessing MANO Performance based on VIM Platforms within MEC Context. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1338–1339, 2020.

[8] Richard Cziva and Dimitrios P Pezaros. Container Network Functions: Bringing NFV to the Network Edge. *IEEE Communications Magazine*, 55(6):24–31, 2017.

[9] Sooeun Song, Changsung Lee, Hyoungjun Cho, Goeun Lim, and Jong-Moon Chung. Clustered Virtualized Network Functions Resource Allocation based on Context-Aware Grouping in 5G Edge Networks. *IEEE Transactions on Mobile Computing*, 19(5):1072–1083, 2019.

[10] Giuseppe Bianchi et al. Superfluidity: a Flexible Functional Architecture for 5G Networks. *Trans. Emerg. Telecommun. Technol.*, 27(9):1178–1186, 2016.

[11] P. L. Ventre, P. Lungaroni, G. Siracusano, C. Pisa, F. Schmidt, F. Lombardo, and S. Salsano. On the Fly Orchestration of Unikernels: Tuning and Performance Evaluation of Virtual Infrastructure Managers. *IEEE Transactions on Cloud Computing*, 2018, early access.

[12] A. Laghrissi and T. Taleb. A Survey on the Placement of Virtual Resources and Virtual Network Functions. *IEEE Communications Surveys Tutorials*, 21(2):1409–1434, 2019.

[13] Industry Specification Group, Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability, V1, document GS NFV-REL 003, ETSI, Sophia Antipolis, France, 2016.

[14] Zenan Wang, Jiao Zhang, Tao Huang, and Yunjie Liu. A clustering-based approach for Virtual Network Function Mapping and Assigning. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–5, 2017.

[15] J David Cummins. Statistical and Financial Models of Insurance Pricing and the Insurance Firm. *The Journal of Risk and Insurance*, 58(2):261–302, 1991.

[16] Yaniv Zaks, Esther Frostig, and Benny Levikson. Optimal Pricing of a Heterogeneous Portfolio for a Given Risk Level. *ASTIN Bulletin: The Journal of the IAA*, 36(1):161–185, 2006.

[17] Zhuonan Huang, Nancy Samaan, and Ahmed Karmouch. A Novel Resource Reliability-Aware Infrastructure Manager for Containerized Network Functions. In *2021 IEEE International Conference on Communications (ICC): Communication*

QoS, Reliability and Modeling Symposium (IEEE ICC'21 - CQRM Symposium), Montreal, Canada, June 2021.

[18] B. P. Rimal, D. Pham Van, and M. Maier. Mobile-Edge Computing Versus Centralized Cloud Computing Over a Converged FiWi Access Network. *IEEE Transactions on Network and Service Management*, 14(3):498–513, 2017.

[19] R. Cziva and D. P. Pezaros. On the Latency Benefits of Edge NFV. In *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 105–106, 2017.

[20] J. Fink. Docker: a Software as a Service, Operating System-Level Virtualization Framework. *Code4Lib Journal*, 2014.

[21] Deval Bhamare, Raj Jain, Mohammed Samaka, and Aiman Erbad. A Survey on Service Function Chaining. *Journal of Network and Computer Applications*, 75:138–155, 2016.

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

[23] Alok Shrivastwa, Sunil Sarat, Kevin Jackson, Cody Bunch, Egle Sigler, and Tony Campbell. *OpenStack: Building a Cloud Environment*. Packt Publishing Ltd, 2016.

[24] Giuseppe A Carella, Michael Pauls, Thomas Magedanz, Marco Cilloni, Paolo Bellavista, and Luca Foschini. Prototyping nfv-based multi-access edge computing in 5G ready networks with open baton. In *2017 IEEE Conference on Network Softwarization (NetSoft)*, pages 1–4. IEEE, 2017.

[25] OPNFV, NOMAD, accessed: 2019-02-11. [Online]. Available: https://www.nomadproject.io.

[26] T. Sechkova, M. Paolino, and D. Raho. Virtualized Infrastructure Managers for Edge Computing: OpenVIM and OpenStack Comparison. In *2018 IEEE International*

Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pages 1–6, 2018.

[27] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running: Dive into the Future of Infrastructure*. O'Reilly Media, Inc., 2017.

[28] S. Clayman, F. Tusa, and A. Galis. Extending Slices into Data Centers: the VIM on-demand model. In *2018 9th International Conference on the Network of the Future (NOF)*, pages 31–38, 2018.

[29] B. Y. Lee and B. C. Lee. Analysis the architecture of VNFM (Virtual network function manager). In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 336–340, 2015.

[30] CloudBand Management System. [Online]. Available: http://www.alcatel-lucent.com/.

[31] Cloud Manager. [Online]. Available: http://www.ericsson.com/.

[32] NFV Director. [Online]. Available: http://www.hp.com/.

[33] N. Gao and H. Jiang. A Resource Reservation Algorithm with Muti-parameters. In *2011 Sixth Annual Chinagrid Conference*, pages 211–214, 2011.

[34] N. Dezhabad, S. Ganti, and G. Shoja. Cloud Workload Characterization and Profiling for Resource Allocation. In *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, pages 1–4, 2019.

[35] Mansaf Alam, Kashish Shakil, and Shuchi Sethi. Analysis and Clustering of Workload in Google Cluster Trace based on Resource Usage. 01 2015.

[36] Sheng Di, Derrick Kondo, and Franck Cappello. Characterizing and modeling cloud applications/jobs on a Google data center. *The Journal of Supercomputing*, 69:139–160, 07 2014.

[37] Asit K. Mishra, Joseph L. Hellerstein, Walfredo Cirne, and Chita R. Das. Towards Characterizing Cloud Backend Workloads: Insights from Google Compute Clusters. *SIGMETRICS Perform. Eval. Rev.*, 37(4):34–41, March 2010.

[38] Naresh Sehgal, Pramod Bhatt, and John Acken. *Cloud Workload Characterization*, pages 75–97. 01 2020.

[39] Nilabja Roy, Abhishek Dubey, and Aniruddha Gokhale. Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting. In *2011 IEEE 4th International Conference on Cloud Computing*, pages 500–507. IEEE, 2011.

[40] Jia Rao, Yudi Wei, Jiayu Gong, and Cheng-Zhong Xu. QoS Guarantees and Service Differentiation for Dynamic Cloud Applications. *IEEE Transactions on Network and Service Management*, 10(1):43–55, 2012.

[41] Qi Zhang, Quanyan Zhu, Mohamed Faten Zhani, Raouf Boutaba, and Joseph L Hellerstein. Dynamic Service Placement in Geographically Distributed Clouds. *IEEE Journal on Selected Areas in Communications*, 31(12):762–772, 2013.

[42] Milad Ghaznavi, Aimal Khan, Nashid Shahriar, Khalid Alsubhi, Reaz Ahmed, and Raouf Boutaba. Elastic virtual network function placement. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 255–260. IEEE, 2015.

[43] H. Tang, D. Zhou, and D. Chen. Dynamic Network Function Instance Scaling Based on Traffic Forecasting and VNF Placement in Operator Data Centers. *IEEE Transactions on Parallel and Distributed Systems*, 30(3):530–543, 2019.

[44] Chunlin Li, Jianhang Tang, and Youlong Luo. Elastic edge cloud resource management based on horizontal and vertical scaling. *The Journal of Supercomputing*, pages 1–26, 2020.

[45] Windhya Rankothge, Helena Ramalhinho, and Jorge Lobo. On the Scaling of Virtualized Network Functions. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 125–133. IEEE, 2019.

[46] Hui Yu, Jiahai Yang, Carol Fung, Raouf Boutaba, and Yi Zhuang. ENSC: Multi-Resource Hybrid Scaling for Elastic Network Service Chain in Clouds. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 34–41. IEEE, 2018.

[47] Hui Yu, Jiahai Yang, and Carol Fung. Fine-grained Cloud Resource Provisioning for Virtual Network Function. *IEEE Transactions on Network and Service Management*, 2020.

[48] Meng Wang, Bo Cheng, Shuai Zhao, Biyi Li, Wendi Feng, and Junliang Chen. Availability-Aware Service Chain Composition and Mapping in NFV-Enabled Networks. In *2019 IEEE International Conference on Web Services (ICWS)*, pages 107–115. IEEE, 2019.

[49] Jingyuan Fan, Chaowen Guan, Yangming Zhao, and Chunming Qiao. Availability-aware mapping of service function chains. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[50] Defang Li, Peilin Hong, Kaiping Xue, and Jianing Pei. Availability Aware VNF Deployment in Datacenter Through Shared Redundancy and Multi-Tenancy. *IEEE Transactions on Network and Service Management*, 16(4):1651–1664, 2019.

[51] S. Yow and M. Sherris. Enterprise Risk Management, Insurance Pricing and Capital Allocation http://citeseerx.ist.psu.edu/viewdoc/similar?doi=10.1.1.587.4343type=cc.

[52] B. He, D. Zhang, S. Liu, H. Liu, D. Han, and L. M. Ni. Profiling Driver Behavior for Personalized Insurance Pricing and Maximal Profit. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1387–1396, 2018.

[53] Doron Kliger and Benny Levikson. Pricing insurance contracts — an economic viewpoint. *Insurance: Mathematics and Economics*, 22(3):243 – 249, 1998.

[54] H. Mao and K. Ostaszewski. Pricing insurance contracts and determining optimal capital of insurers. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 1–5, 2010.

[55] B. Wu, J. Zeng, L. Ge, S. Shao, Y. Tang, and X. Su. Resource Allocation Optimization in the NFV-Enabled MEC Network Based on Game Theory. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–7, 2019.

[56] Saqib Rasool Chaudhry, Andrei Palade, Aqeel Kazmi, and Siobhán Clarke. Improved QoS at the Edge Using Serverless Computing to Deploy Virtual Network Functions. *IEEE Internet of Things Journal*, 7(10):10673–10683, 2020.

[57] M. Peuster and H. Karl. Understand Your Chains: Towards Performance Profile-Based Network Service Management. In *2016 Fifth European Workshop on Software-Defined Networks (EWSDN)*, pages 7–12, 2016.

[58] B. K. Ghosh. Probability Inequalities Related to Markov's Theorem. *The American Statistician*, 56(3):186–190, 2002.

[59] Luca Abeni, Alessio Balsini, and Tommaso Cucinotta. Container-Based Real-Time Scheduling in the Linux Kernel. *ACM SIGBED Review*, 16(3):33–38, 2019.

[60] Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: Theory of Majorization and Its Applications*, volume 143. Springer, 1979.

[61] Steven Van Rossem, Wouter Tavernier, Didier Colle, Mario Pickavet, and Piet Demeester. VNF Performance modelling: From stand-alone to chained topologies. *Computer Networks*, 181:107428, 2020.

[62] A. Abdelsalam, P. L. Ventre, C. Scarpitta, A. Mayer, S. Salsano, P. Camarillo, F. Clad, and C. Filsfils. SRPerf: a Performance Evaluation Framework for IPv6 Segment Routing. *IEEE Transactions on Network and Service Management*, pages 1–1, 2020.

[63] Donald A Walker. Walras's Theories of Tatonnement. *Journal of Political Economy*, 95(4):758–774, 1987.

[64] B. Wanis, N. Samaan, and A. Karmouch. Modeling and pricing cloud service elasticity for geographically distributed applications. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 559–565, 2015.

[65] J. Kong, I. Kim, X. Wang, Q. Zhang, H. C. Cankaya, W. Xie, T. Ikeuchi, and J. P. Jue. Guaranteed-Availability Network Function Virtualization with Network Protection and VNF Replication. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.