UNIVERSITÀ DEGLI STUDI DI GENOVA

DOCTORAL THESIS

Enabling Scalable and Sustainable Softwarized 5G Environments

---

Candidate:

Jane Frances PAJO

Advisers:

Prof. Raffaele BOLLA
Prof. Franco DAVOLI

Tutor:

Dr. Roberto BRUSCHI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Science and Technology for Electronic and Telecommunications Engineering
(Curriculum: Interactive Cognitive Environments)

Department of Electrical, Electronics and Telecommunications Engineering and Naval Architecture (DITEN)
Polytechnic School of Engineering and Architecture

February 2019

"Learn as if you were to live forever."

Mahatma Gandhi

Abstract

Enabling Scalable and Sustainable Softwarized 5G Environments

by Jane Frances PAJO

The fifth generation of telecommunication systems (5G) is foreseen to play a fundamental role in our socio-economic growth by supporting various and radically new vertical applications (such as Industry 4.0, eHealth, Smart Cities/Electrical Grids, to name a few), as a one-fits-all technology that is enabled by emerging softwarization solutions – specifically, the Fog, Multi-access Edge Computing (MEC), Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) paradigms. Notwithstanding the notable potential of the aforementioned technologies, a number of open issues still need to be addressed to ensure their complete rollout. This thesis is particularly developed towards addressing the scalability and sustainability issues in softwarized 5G environments through contributions in three research axes: a) *Infrastructure Modeling and Analytics*, b) *Network Slicing and Mobility Management*, and c) *Network/Services Management and Control*. The main contributions include a model-based analytics approach for real-time workload profiling and estimation of network key performance indicators (KPIs) in NFV infrastructures (NFVIs), as well as a SDN-based multi-clustering approach to scale geo-distributed virtual tenant networks (VTNs) and to support seamless user/service mobility; building on these, solutions to the problems of resource consolidation, service migration, and load balancing are also developed in the context of 5G. All in all, this generally entails the adoption of Stochastic Models, Mathematical Programming, Queueing Theory, Graph Theory and Team Theory principles, in the context of Green Networking, NFV and SDN.

Acknowledgements

I would like to express my heartfelt gratitude to everyone who has contributed in one way or another in the fulfillment of this Doctoral degree.

To my advisers, Prof. Raffaele Bolla and Prof. Franco Davoli, for giving me various opportunities for growth in both technical and social aspects; and to my tutor, Dr. Roberto Bruschi, for the guidance as I embark on my research path; you have been very instrumental in this achievement, I could not thank you enough.

To the Department of Electrical, Electronics and Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa, for the three-year PhD scholarship that allowed me to pursue this dream; and to the Italian National Consortium for Telecommunications (CNIT) for supporting my publications under the frameworks of the European H2020 projects INPUT and MATILDA; I will always be grateful for the many doors that you have and will open for me through the PhD schools/courses, training activities and conferences, both in Italy and abroad.

To Dr. Paolo Lago for following me in the early stages of my PhD and finding time to answer my doubts along the way; to Sergio Mangialardi for the support in setting up testbeds and performing experiments; and to all my colleagues in the Telecommunication Networks and Telematics (TNT) Laboratory – the joint research lab of DITEN and CNIT – for the great company in and out of work; I share this success with you.

To Prof. Dimosthenis Kyriazis for hosting me as a visiting student researcher in the Department of Digital Systems, University of Piraeus, which gave me fresh research perspectives moving forward; and to everyone in Room 207 for the friendship and memorable Greek experience; those four months well spent will always be cherished, and I look forward to our future collaborations and reunions.

To the external reviewers, Prof. José Luis Marzo and Prof. Stefano Secci, for taking time to read and critic this thesis; your kind efforts are deeply appreciated.

To my beloved parents, Leonila and Danilo, for the all-out support and constant encouragement; and to the love of my life, Odnan, for being my rock through everything; your unconditional love have always led me through life's journey, I dedicate this work to you. To the rest of my family and friends: your mere presence fill me with inspiration, thank you.

Most of all, my highest praises to our Almighty Father for all the blessings He showered upon me and my loved ones, as well as for granting me the wisdom and knowledge to accomplish all of these; I would not be here if it were not for Him.

Jane

Contents

List of Figures

List of Tables

Dedicated to Nanay Ning, Tatay Dan and my Lablab Bobi…

Chapter 1

Introduction

In recent years, a multitude of services is being made available at accelerated innovation rates to offer new opportunities for improving our quality of life. To cope with current and future demands, telecommunication networks have been continuously evolving towards convergence; for one, it is imperative to handle the increasing heterogeneity in traffic types, applications, access devices and platforms, among others, while providing consistent – yet differentiated – quality and reliability. Nonetheless, the subsequent increase in the number of devices (of varying capabilities) connecting to the Internet and traffic volumes [1, 2], topped with differing user requirements, entail complex network operations and management that render classical network architectures and paradigms inadequate.

The *fifth generation* of telecommunication systems (5G) is envisioned to bring about a *truly converged* network that will play a fundamental role in both economic and social aspects of our digital world [3, 4]. For instance, Fig. 1.1 shows EU's vision of 5G – a *hyperconnected* environment that offers a wide range of applications dedicated to both end-users and network-connected *things* (e.g., wearables, handhelds, cars, home appliances, sensor networks, etc.). Radically new applications, such as the Tactile Internet (i.e., augmented reality, real-time autonomic control for Industry 4.0, etc.), eHealth, Smart Cities/Electrical Grids use cases, are key design milestones to be supported as *verticals* by the upcoming 5G technologies.

FIGURE 1.1: EU's vision of 5G [4].

Towards this end, the integration of networking paradigms with Information Technology (IT) services (specifically, the virtualization on top of commercial off-the-shelf (COTS) hardware) has become a trend in both industry and research scenes due to their notable potential in overcoming the infrastructure *ossification* dilemma. Moreover, with the onset of Fog [5] and Multi-access Edge Computing (MEC) [6] paradigms, Cloud-like services will soon become – if not already – widely available much closer to end-users and network-connected things, enabling new classes of services with challenging performance/operating requirements that cannot be met by current Cloud solutions running on remote datacenters. The resulting Cloud-Fog-MEC interplay then provides a wide-range of possibilities for augmenting device capabilities and improving network performance [1, 7, 8], granting network service providers (NSPs) more degrees of freedom in improving either the quality of service (QoS) of service components or the quality of experience (QoE) of the end-users, as necessary. In fact, in the 5G scenario, telecommunication infrastructures are emerging into a sort of "geo-distributed datacenter" with advanced virtualization and computing capabilities, able to host both network and application software instances from multiple tenants.

Building on this, emerging network *softwarization* solutions aim at providing flexibility and programmability levels that could future-proof the network. Particularly, Network Functions Virtualization (NFV) [9] and Software-Defined Networking (SDN) [10] are gaining momentum as two core technologies in the current 5G revolution [3]. On one hand, NFV allows for high customizability and improved time to market of network services, at lesser capital expense (CapEx), by the software implementation of networking functionalities that can be (dynamically) placed practically anywhere in the Cloud-Fog-MEC domain. On the other hand, network management complexity is expected to significantly drop with SDN, as it decouples the network intelligence from the forwarding plane, rendering forwarding devices simple and programmable via an open interface like OpenFlow (OF) [11]. Notwithstanding the numerous gains offered by these technologies, the economic impact, environmental sustainability, scalability, and performance levels that they must contest are foreseen to create a bottleneck in their complete rollout.

Within the challenges posed by the Future Internet in general, and particularly by the strong wireless/wired integration [12] in 5G environments, four broad topics, among others, can be seen as interacting and mutually influencing: a) flexibility, programmability and virtualization of network functions and services; b) performance requirements (mapping of users' QoE onto network-level QoS); c) energy efficiency, and; d) network management and control. The first item stems from the evolution of the network towards a multi-purpose softwarized service-aware platform over a heterogeneous infrastructure [13]. Performance issues have to deal with the very strong requirements imposed by 5G key performance indicators (KPIs) [14] and energy-awareness cannot be neglected in view of sustainability, environmental concerns, and operational costs. In this scenario, network management and control strategies are essential to orchestrate all needed functionalities, supervise and optimize the allocation of resources,

to ensure that KPIs are met for network *slices* [15–17] according to the traffic dynamics, multiple tenants, NSPs and infrastructure providers (IPrs).

This thesis is particularly developed towards addressing these open issues through contributions organized in three research axes: a) *Infrastructure Modeling and Analytics*, b) *Network Slicing and Mobility Management*, and c) *Network/Services Management and Control*, with an end goal of enabling scalable and sustainable softwarized 5G environments. The first and second axes cover the main contributions of this thesis, detailing a model-based approach towards real-time analytics in NFV infrastructures (NFVIs), as well as a SDN-based multi-clustering approach to scale geo-distributed virtual tenant networks (VTNs) and to support seamless user/service mobility. The third axis is intended for some applications of the aforementioned approaches, and of a more recent work based on team theory, towards solving the problems of resource consolidation, service migration, and load balancing in the context of 5G. The following sections briefly look into each research axis to further motivate and formally introduce the corresponding contributions.

1.1 Infrastructure Modeling and Analytics

At the infrastructure level, state-of-the-art softwarization solutions are mostly based on the use of COTS hardware, which – contrary to the special-purpose hardware mostly deployed within classical telecommunication infrastructures – have intrinsically lower performance and energy efficiency. While the Advanced Configuration and Power Interface (ACPI) specification [18] equips most – if not all – of it with power management mechanisms (e.g., Low Power Idle (LPI) and Adaptive Rate (AR)), power savings come in trade-off with performance degradation [19]. Moreover, virtualization typically adds extra layer(s) in the networking stack that result in additional processing delays, further lowering the performance. For instance, for a given amount of workload, *virtualized* network functions (VNFs) may consume even more energy than their physical counterparts [20, 21]; this calls for power- and performance-aware resource management solutions, otherwise the resulting operational expenses (OpEx) and carbon footprints will prove to become economically and environmentally unsustainable.

Furthermore, this current shift towards an extremely modular and virtual network architecture, which is foreseen to provide the service *agility* required by the upcoming 5G scenario, entails automated configuration, provisioning and anomaly detection. The ETSI NFV Management and Orchestration (NFV-MANO) framework [22] designates these responsibilities to the virtual infrastructure manager (VIM) of the NFVI. The VIM seeks to obtain performance and anomaly information among virtualized resources based on capacity/usage reports and event notifications, and then to manage them accordingly – yet usually, measurable data do not directly expose network KPIs.

Starting from available and easily measurable performance monitor counters (PMCs) in Linux host servers, we try to bridge this gap through a model-based analytics approach for real-time VNF workload profiling and network KPI estimation (specifically,

power and latency). In particular, the contribution in this aspect is two-fold:

- a complete analytical characterization of the power- and performance-aware virtualized system, considering a fairly general renewal model ($M^X/G/1/SET$ queue [23]) that captures traffic *burstiness* and system *setup times*, and;
- a novel model-based analytics approach for profiling VNF workloads, towards the real-time estimation of the ensuing power consumption and system latency.

This not only augments the capabilities of the VIM, but is also suitable for state-of-the-art dynamic resource and service provisioning approaches, among others, and for devising new ones.

1.2 Network Slicing and Mobility Management

In the 5G scenario, computing resources will be provided by heterogeneous and resource-constrained facilities, placed at various levels of the network, such as micro-, pico- and container-based datacenters [24] in central offices, street cabinets [25], mobile base stations [26], etc. – or collectively referred to as *in-network* datacenters. At the same time, Cloud-native applications have been evolving towards software architectures to boost and embrace modularity. In particular, state-of-the-art applications are composed of a large number of *microservices* [27], often embedded into various execution containers or *virtual objects* (*VOs* – e.g., virtual machines (VMs), Linux containers, etc.), and connected among themselves through complex virtual network topologies (often referred to as *service chains* (*SCs*)). As in the ETSI NFV specifications [28], an application orchestrator is used to dynamically manage the lifecycle of each software instance and of its virtual connectivity to conform with the application-level workload, while meeting the desired performance levels.

Note that such modular applications are the perfect counterpart to be vertically deployed onto 5G-ready infrastructures: microservices/VOs can be placed into the geo-distributed in-network datacenters close to end-users and network-connected things [29], and migrated from one to another as they move, in order to enable seamless user experiences. Moreover, depending on the nature of the application, or on one of the functions fulfilled by a certain microservice, VOs could be differently tolerant to the end-to-end latency, and hence, to their proximity to users and/or things. Thus, it is reasonable to assume that each VO is associated to a Service Level Agreement (SLA), defining its QoS requirements, which can be used to select the most suitable hosting datacenter. Obviously, the tighter the SLA proximity requirement of a VO is, the more frequently it might be migrated among datacenters as the users/things move, causing resource reconfigurations in the network nodes and links. In addition, we can also note how users can have access to multiple Cloud-Fog-MEC applications, which in turn are provided and managed by potentially different tenants.

Owing to the considerations above, the crucial role of SDN technologies (e.g., OF) becomes extremely manifest in this challenging scenario. SDN not only offers effective

and flexible means for isolating VTNs, but also the possibility of supporting SC reconfigurations – especially the live migration of VOs among in-network datacenters in an effective and seamless fashion, since it allows steering traffic flows from/to VOs upon orchestrator requests. In this perspective, a VTN can be viewed as a network slice or a slice instantiation that can be flexibly and dynamically mapped to a geo-distributed telecommunication infrastructure.

With this in mind, we exploit SDN's capabilities and develop a novel multi-clustering approach to scale geo-distributed VTNs, especially in the presence of a massive number of Cloud-Fog-MEC applications and VOs. The contribution in this aspect can be summarized through the following main advantages that the proposed approach provides with respect to state-of-the-art SDN mechanisms:

- overlay isolation through *tunnel-less* communications [30] (i.e., non-overlapping OF rules among different overlays without the use of resource-hungry tunneling protocols like GRE [31] or VXLAN [32]);
- intrinsic support for distributed computing facilities (i.e., overlay connectivity is provided inside and among datacenters at the network edge);
- more abstracted and agile overlay network control (i.e., clusters of VOs with similar SLA requirements are considered as aggregate entities in the wide-area);
- efficient network support for single and bulk VO live migrations (i.e., no network-induced packet loss);
- low infrastructure-level requirements (i.e., simple and mandatory OF filters and actions, as well as layer-2 (L2) addressing criteria);
- high performance (i.e., optimal/close-to-optimal traffic paths, low computational overhead); and
- high scalability (i.e., the number of OF rules installed in the overlay implementation and of rule updates during wide-area (bulk) migrations are significantly reduced).

1.3 Network/Services Management and Control

The ETSI NFV standard [33] defines an architectural framework where the operational domains of IPrs and NSPs (e.g., vertical industries, over-the-top network providers, etc.) are fully split. In more detail, NSPs can define their own services and instantiate their components over infrastructure resources (in terms of computing, storage and networking) acquired "as-a-Service" from different IPrs. At the same time, IPrs can simultaneously host multiple NSPs in their respective (in-network) datacenters – also referred to as NFVI *Points of Presence* (*PoPs*).

The scenario can be viewed from a sort of "Cloud" perspective, in which the NFV framework can be defined as a *multi-domain* and *multi-tenant* architecture. NSPs are

envisioned to compose and orchestrate the lifecycle of their services, instantiating their components (i.e., execution containers or VOs) over the resources acquired as-a-Service from PoPs. In their turn, IPrs are expected to handle the service components running in their infrastructure in an efficient fashion – for instance, by consolidating execution containers or VOs in a subset of servers and/or by applying energy-aware local control policies (LCPs) on the underlying hardware in order to reduce the power consumption of PoPs, while taking into account the QoS/QoE requirements.

With this in mind, we devise network/services management and control solutions by building on the real-time analytics, network slicing and mobility management mechanisms developed in the first two research axes. In addition, a more recent work based on team theory is presented.

Joint power scaling and consolidation: While numerous studies have addressed the VM consolidation problem in datacenters, none of them has (completely) considered the AR and LPI capabilities of underlying resources in today's telecommunication infrastructures. In this respect, we seek to optimize the trade-off between power and performance, by considering the aforementioned capabilities, as well as the workload characteristics and variations, in the dynamic consolidation of VNFs. The solution is developed under the notion that the model-based analytics approach in Section 1.1 is already available for real-time workload profiling and network KPI estimation.

User-centric service migration: An open issue with the upcoming Cloud-Fog-MEC interplay, brought by recent advances in mobile Internet, regards user mobility support – specifically, the Fog-MEC counterparts of services that require close proximity may call for migration(s) to meet the SLA requirements as users move. At the same time, correlation/chaining among VOs (or, inter-VO *affinity*) must be taken into account in making such migration decisions, which may initiate (bulk) VO migration(s) with user mobility. Supposing that the multi-clustering approach in Section 1.2 is already in place for managing geo-distributed VTNs, we look at a user-centric perspective and jointly consider the user proximity and inter-VO affinity requirements, as well as a subscription-based parameter, in a scalable and differentiated approach for service migration.

Decentralized load balancing: In a multi-domain and multi-tenant scenario such as the upcoming 5G environments, the management and control of dynamic network slices pose a set of challenges that are still largely unsolved. For instance, a certain network service can generally be offered by multiple NSPs (i.e., with different SC implementations), and depending on the service demands, each NSP can activate multiple instances of its SCs. This manifests the need for load balancing techniques able to consider the entire end-to-end SC deployment and to optimally steer traffic among multiple service instantiations. We scalably address this problem with a team-theoretic approach for finding informationally decentralized (per-PoP) load balancing strategies, which takes into account a power- and performance-aware cost function.

The remainder of this thesis is structured as follows. Firstly, some technological background and related literature are presented in Chapter 2. Chapters 3, 4 and 5 delve into the three research axes, respectively, discussing the details of the proposed solutions, as well as the obtained results.

Chapter 2

Background and Related Work

This chapter provides an overview of the technological scenario – encompassing the fundamental concepts and the related literature – on which we build the problems considered in this thesis and the proposed contributions in their respect.

First, 5G is briefly outlined to get acquainted with its use cases and their corresponding requirements. Then, we look into some key concepts and technologies foreseen to facilitate the roll out of softwarized 5G environments – specifically, the Cloud-Fog-MEC interplay, NFV and SDN. As previously noted, the resulting scenario still admits a number of open issues at the infrastructure and service levels, which will be described – along with some related work – in the context of network/services management and control. Furthermore, as a prelude to one of the main contributions of this thesis, a run-through of related system modeling and analytics approaches is also presented.

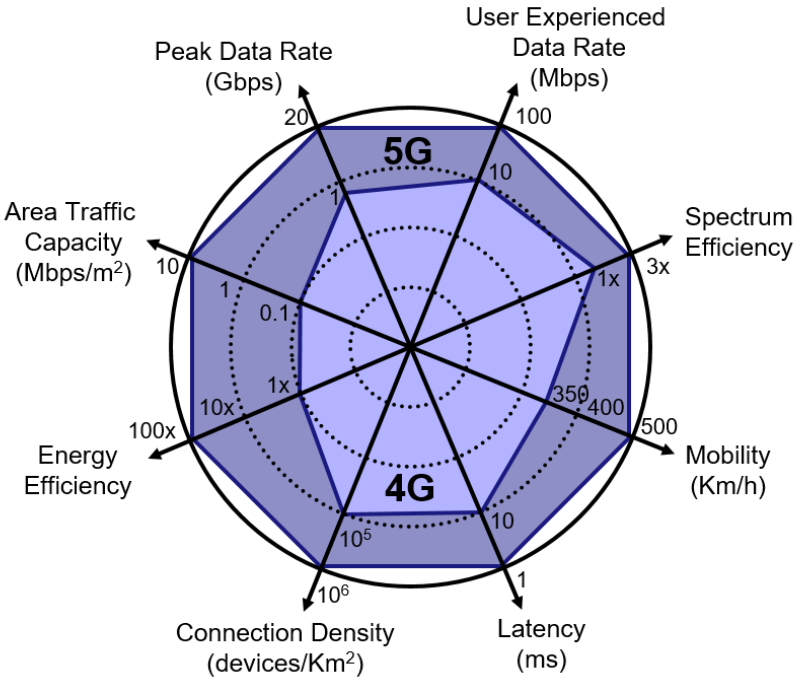
2.1 5G in a Nutshell

5G goes beyond the next generation of mobile broadband but rather an evolution of telecommunication technologies in their entirety [3] – embracing the heterogeneity in access technologies (i.e., wireless/wired integration), underlying resources (i.e, Cloud-Fog-MEC interplay), QoS/QoE requirements, etc. – towards a unified infrastructure for enabling next-generation network services and applications.

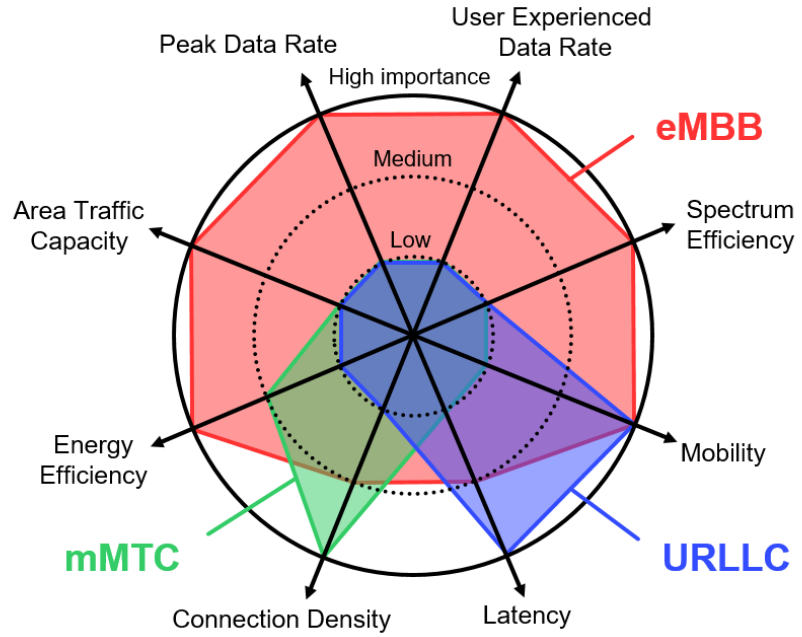
The ITU-R has defined the following three main categories of 5G use cases [34]:

- Enhanced mobile broadband (eMBB) (i.e., enhanced indoor/outdoor broadband, augmented and virtual reality, etc.);
- Massive machine-type communications (mMTC) (i.e, Internet of Things (IoT), Smart Cities, Domotics, etc.), and;
- Ultra-reliable and low-latency communications (URLLC) (i.e., autonomous vehicles, Smart Electrical Grids, eHealth, Industry 4.0, etc.);

based on application-specific requirements that takes into account the prioritization among 5G KPIs. Considering eight main KPIs [35]: a) *peak data rate*, b) *user experienced data rate*, c) *latency*, d) *mobility*, e) *connection density*, f) *energy efficiency*, g) *spectrum efficiency*, and h) *area traffic capacity*, Fig. 2.1a illustrates the target improvements for 5G



(A) 4G vs 5G



(B) eMBB vs mMTC vs URLLC

FIGURE 2.1: 5G KPIs and targets [35].

with respect to the current 4G technologies, while Fig. 2.1b indicates the prioritization among these KPIs for eMBB, mMTC and URLLC applications.

As a *one-fits-all* technology, 5G will have to overcome the challenges of simultaneously handling the great diversity of requirements. While the need for different Radio Access Networks (RANs) has been anticipated in this respect, the adoption of softwarization solutions on the core network is foreseen to enable efficient and tailored

management of resources on a per-slice perspective [3].

The NGMN and 3GPP recently introduced the network slicing concept into 5G ecosystem specifications [16, 17], where a network slice can be roughly summarized as a virtual projection of a 5G network with all the functionalities, isolation level, and capabilities customized according to the needs of the vertical applications. Moreover, each network slice can be defined by one or more interconnected logical subnetworks which might provide different functionalities or can be even shared with other slices.

2.2 Network Softwarization and Key Enablers

As pointed out by the 5GPPP SN WG, network softwarization generally refers to the telecommunication architecture paradigm shift: "from *boxes* to *functions*, and from *protocols* to *application programming interfaces* (*APIs*)" [3]. In this section, we look into the key enabling concepts and technologies for this softwarization revolution.

2.2.1 Cloud-Fog-MEC Interplay

In a broad sense, virtualization basically describes the separation between a service and its physical implementation [36], on which the Cloud, Fog and MEC paradigms are built upon. With the hardware and software separation brought by state-of-the-art virtualization technologies, Fig. 2.2 illustrates the resulting pooling of computing, storage and networking resources in a virtual infrastructure – that could be of a single Cloud/Fog/MEC (in-network) datacenter, or of an entire geo-distributed telecommunication infrastructure. Moreover, the execution containers or VOs (e.g., VMs, Linux containers, etc.) running the software applications are seen as isolated entities that can be dynamically instantiated and (live) migrated across the shared infrastructure.

As previously anticipated, the Fog and MEC paradigms are relatively recent developments that aim at bringing Cloud-like services (i.e., from providing intelligence to dumb devices, to offloading smart ones) much closer to end-users and network-connected things. It is interesting to note that the former is a sort of generalization

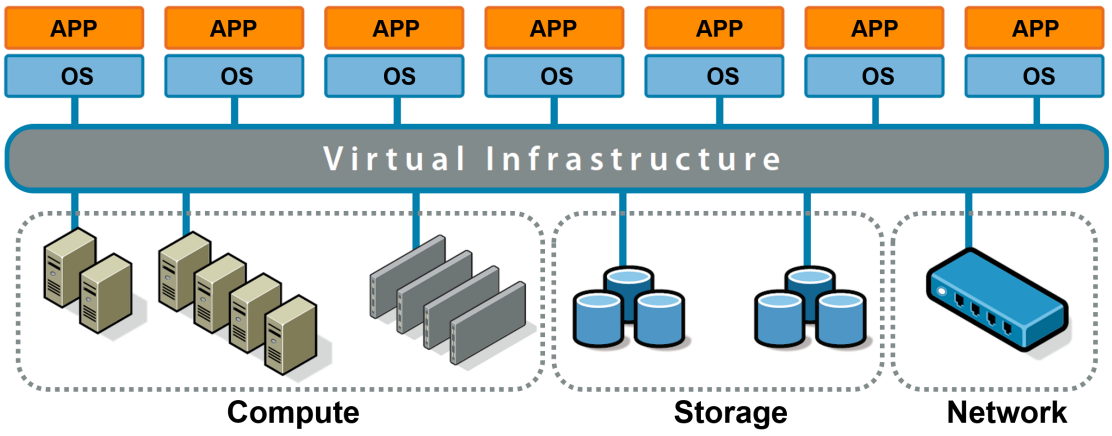


FIGURE 2.2: Pooling of resources in a virtual infrastructure [36].

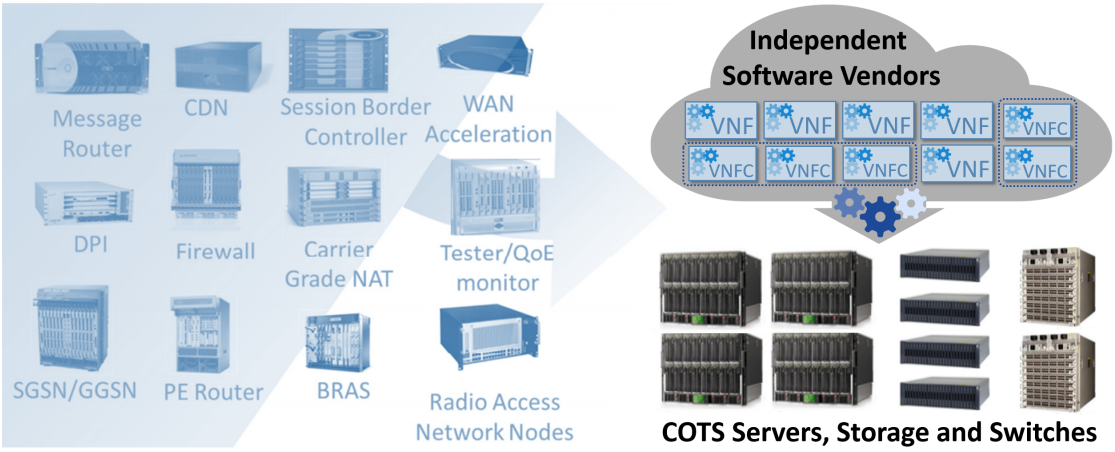


FIGURE 2.3: Classical network appliance approach vs NFV [9, 37].

of the latter, as it conceives the possibility of having different levels of proximity that can span from end-user devices, through the edge and core networks, and up to the Cloud. In fact, Fog nodes are expected to be deployed practically anywhere – wherever there are available computing/storage resources and of course, network connectivity [5]. Intuitively, there is an inherent trade-off between the achievable end-to-end latency and the available computing/storage/networking capacities, as you move amidst the Cloud, Fog and MEC domains, that must be taken into account in the dynamic network/services management and orchestration.

2.2.2 Network Functions Virtualization

With NFV, various types of network appliances are being consolidated as VNFs onto COTS servers, storage and switches, as shown in Fig. 2.3, that are expected to populate the NFVI PoPs across the Cloud-Fog-MEC domain [9]. From this perspective, the ETSI NFV WG generalizes the definition of a network service as a graph of network functions (either physical or virtual) connecting network end-points. In its turn, each network function can be hierarchically composed of further functions or by components, such that VNF components (VNFCs) represent the lowest decomposition level and might correspond to the execution containers or VOs (e.g., VMs, containers, etc.) [37]. Moreover, a set of VNFs and/or VNFCs chained together to provide specific service functionality is referred to as a VNF Forwarding Graph (VNF-FG) [33] – an oriented graph, where each node corresponds to a particular VNF or VNFC, and each edge describes the operational flow exchanged between a pair of VNFs and/or VNFCs.

Basically, the NFVI can employ various virtualization layer solutions towards the deployment of VNFs and VNFCs. This involves selection among (or mixing of) different virtualization techonologies, as well as their corresponding platforms and I/O technologies, which govern the overall performance and flexibility of the network service implementation [38–40].

In more detail, the performance yardstick in NFV is a set of network KPIs (e.g., power consumption, latency, response time, maximum throughput, isolation, mobility

management complexity, instantiation time, etc.); this set varies (or at least the weight of each KPI) with the application. Nonetheless, the overall performance is closely linked to the level of abstraction, and hence, to the virtualization overhead introduced in the chosen implementation. For instance, as described in [38], typical *hypervisor-based* solutions create isolated VMs that are highly abstracted and flexible but with relatively high overhead, while *container-based* solutions create isolated guests (referred to as containers) that directly share the host OS, thus avoiding much of the overhead but with a number of flexibility limitations (e.g., consolidation of heterogeneous VNFs, mobility support, etc.).

Other ways for reducing the virtualization overhead regard the handling of network I/O. A number of works in the scientific literature (e.g., [41–43], among others) consider technologies like Single Root I/O Virtualization (SR-IOV) and Intel's Data Plane Development Kit (DPDK), which bypass the OS network stack. This, however, entails building a specialized network stack on applications that require one, and the device cannot be shared with other applications [40].

The SDN paradigm is recognized to be a highly complementary to NFV, as it enables the programmable sequencing of network functions (both physical and virtual) for network services deployment within the Cloud-Fog-MEC domain [3].

2.2.3 Software-Defined Networking

As detailed in [10], the SDN network architecture can be characterized by four design viewpoints:

- *decoupling of the control and data planes* – forwarding devices (either physical or virtual) become simple and programmable via an open southbound API;
- *logically centralized control function* – the SDN controller becomes a sort of network OS with a global view of the network state, which provides the necessary resources and abstractions for adapting network behavior;
- *network programmability* – the interaction between network applications and the underlying forwarding devices can be programmed with unprecedented flexibility (i.e., leveraging an open northbound API to implement the network control and operation logic, which in turn are translated to southbound-specific instruction sets), and;
- *flow-based forwarding* – forwarding devices perform the same instruction set on all packets matching a flow's criteria, which are executed by their corresponding flow table implementations.

Fig. 2.4 illustrates the resulting shift in the networking approach – from infrastructure ossified to highly programmable network behavior. On the downside, due to the ensuing costs, the Ternary Content-Addressable Memory (TCAM) essential to accommodate fine-grained forwarding rules in SDN devices have smaller capacities than the binary

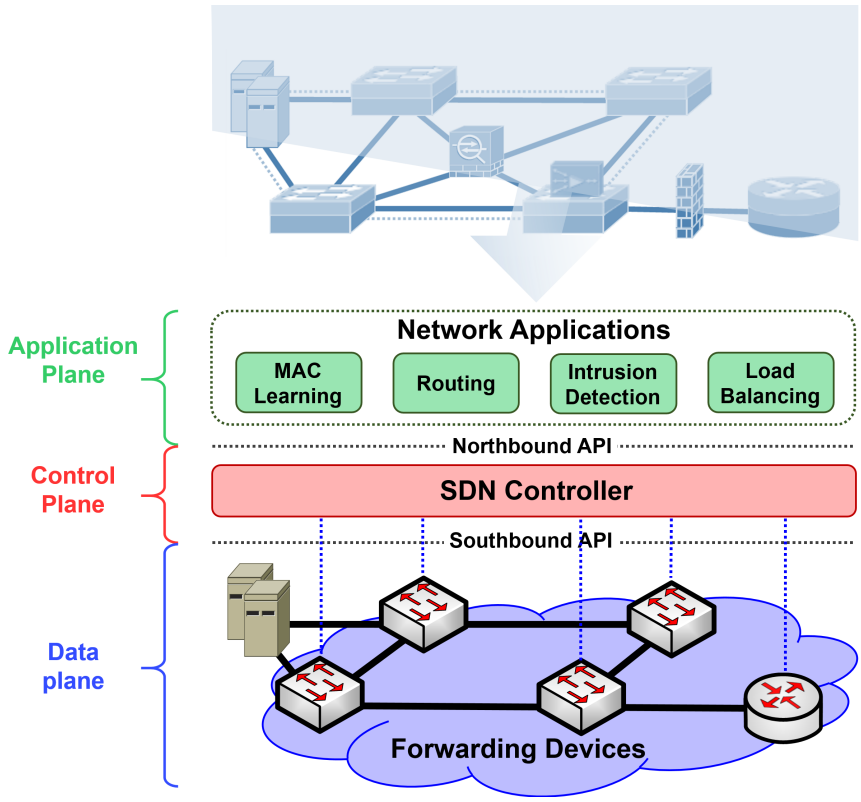


FIGURE 2.4: Classical networking vs SDN [10].

CAMs in legacy routers [44]. In light of this, numerous works in the scientific literature have sought for solutions that could support customized traffic handling, while minimizing the size of forwarding tables (e.g., Media Access Control (MAC) addresses as *light-weight* universal labels [45] and MAC rewriting at the network edge [46], embedding a packet's path in the address fields [47] and forwarding table compression [44, 48, 49], among others).

In the collective context of SDN/NFV, the network applications can correspond to the VNFs or VNFCs, which can be (re-)instantiated and/or migrated practically anywhere in the Cloud-Fog-MEC domain during network (re)configurations. From this perspective, SDN is well-known to fit re-routing mechanisms that would avoid any network-induced performance drawbacks (e.g., encapsulation overhead [32], packet losses [50], delays due to reactive rebuilding of forwarding tables [51]), which typically occur in legacy L2/L3 networks [52].

Furthermore, within the multi-tenant and multi-service 5G scenario, SDN can be essentially useful in implementing virtual networks (as overlays over the shared infrastructure) while efficiently ensuring address space isolation among overlays (e.g., using host-based VLAN tags for tunnel-less overlays [30]). A large part of the related work on this aspect focuses on the efficient management of a single datacenter, rather than the complete resource pool of multiple, geo-distributed datacenters.

2.3 Network/Services Management and Control

With the use of COTS hardware, reduction in CapEx is a straightforward advantage brought forth by softwarization solutions, while a similar reduction in OpEx necessitates the adoption of specific management and control solutions. In this section, we look at both infrastructure- and service-level approaches towards meeting the power and performance requirements of 5G networks.

2.3.1 Power Management in COTS Hardware

The massive introduction of COTS hardware with the adoption of NFV would tend *per se* to increase power requirements with respect to specialized hardware solutions [20, 21], in the absence of specific control actions. Among the various techniques that can be used to implement control policies in network processing devices, dynamic adaptation ones consist of modulating the processing rate (i.e., AR) or of exploiting low power consumption states in idle periods (i.e., LPI).

In the context of softwarized networks, where a collection of network service components must be allocated on physical network nodes, the latter may apply LCPs implementing such dynamic adaptation techniques. Moreover, note that a network service request can be allocated either on dedicated hardware, on virtualized resources deployed in the Cloud-Fog-MEC domain, or their combination. The overall energy cost and performance of a network service then depends on the LCPs applied on each node involved in the chain.

The ACPI specification

The ACPI exposes the LPI and AR functionalities at the software level through the *power* (C_x) and *performance* (P_y) states, respectively. The former comprise the active state C_0 and the sleeping states $\{C_1, \ldots, C_X\}$, while the latter correspond to different processing performances $\{P_0, \ldots, P_y\}$ at C_0. Higher values of x and y indexes indicate deeper sleeping states and lower working frequencies and/or voltages, respectively. Sleeping states, although resulting in lower power consumptions, incur performance degradation due to the wakeup times, whereas reduced processing capacity increases the service times.

It can be noted in Fig. 2.5 how LPI and AR have opposite effects on the burstiness of the processor's workload (i.e., the former clusters packets into bursts, while the latter smoothens the traffic profile). Joint adoption of both mechanisms does not guarantee greater savings [53, 54] – in fact, negative savings may even result with the naïve use of the ACPI [55]. Furthermore, the optimum configuration largely depends on the inherent burstiness of the incoming traffic.

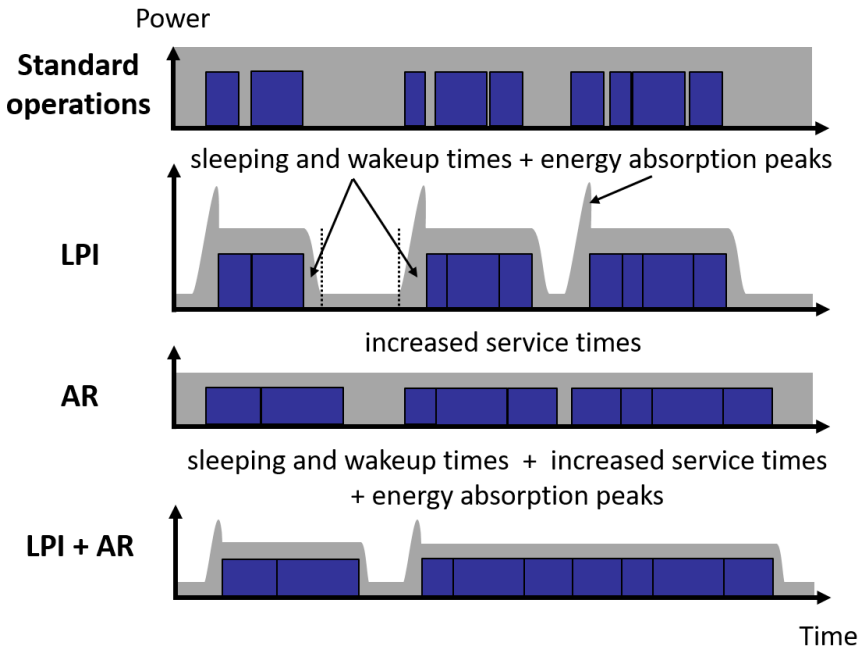


FIGURE 2.5: Power/performance comparison between standard packet processing and with ACPI enabled.

Optimizing the power-performance trade-off

Numerous works in the literature looked into the power-performance trade-off optimization according to different problem formulations. For instance, the multi-objective problem can be transformed into a single-objective problem through linear scalarization (e.g, power and delay are weighted based on a trade-off parameter [53, 54]), contraint method (e.g., delay and packet loss are upper bounded [56]), among others.

In addition, the product between the power consumption and processing delay has been widely used in recent years for modeling and optimizing the design of modern (virtualization-ready) computing systems and components, like multi-core processors and systems-on-chip, according to the performance of software applications [57].

2.3.2 Dynamic Management of Cloud-Fog-MEC Resources

One of the main objectives of the ETSI NFV-MANO orchestrator [22], along with the help of VNF managers, is to automatically and dynamically manage SCs and the configuration of any single component to cope with the offered load and performance requirements. This aspect is directly inherited from the *elasticity* capability in today's Cloud computing technologies [58], which allows tenants to dynamically acquire and release resources as-a-Service depending on their needs. In this respect, two base techniques, namely *vertical* and *horizontal scaling* can be used in a non-exclusive fashion [59]. The former is clearly upper limited by the resources available in the hosting servers, while the latter removes such limitation by enabling the creation/removal of copies of the same VNF/VNFC on-the-fly and balancing the load among these copies accordingly.

Service composition and placement

Recall that a network service can be implemented as one or a chain of physical and/or virtual components, and various NSPs can offer the same services but orchestrated differently. In this respect, numerous works have proposed different approaches to the resulting service composition and placement problems, which are mostly based on either state-of-the-art solutions to the Virtual Network Embedding (VNE) problem [60], or sub-optimal centralized solutions that require a full knowledge on the network state and system parameters, as pointed out in [61].

The complexity of these problems further increase when aspects like underlying hardware capabilities, resource availability and affinities are taken into account [62], as well as service access patterns and their geographical distribution [63, 64], among others. Distributed and informationally decentralized approaches (e.g., [61] and [65]) may come in handy in maximally capturing the variety of problem conditions, while achieving the desired scalability in the solution.

Resource allocation and consolidation

Generally, the dynamic allocation of various types of resources (i.e., network links' bandwidth, nodes' CPU, memory and network I/O) and consolidation of workloads among them must be considered when managing SCs. With softwarization solutions, 5G networks is expected to be able to vertically and horizontally scale service components on-demand, and dynamically manage their mapping to the underlying infrastructure in order to meet application-specific KPIs.

Within a Cloud/Fog/MEC facility, state-of-the-art datacenter management solutions look into dynamically optimizing the number of active resources according to the datacenter load [66–68], while also minimizing unnecessary migrations [69, 70]. More recent developments, such as [71], also started to consider service-level constraints (e.g., component sequencing, end-to-end delay).

Most works on energy-aware management are basically based on simple on-off capabilities – for instance, making underutilized nodes/links idle (i.e. via live migration and re-routing operations) and possibly turning them off; although there are also a few (e.g., [72] and [73]) that attempt to capture the AR and/or LPI capabilities of underlying devices in their solutions.

Load balancing

Path diversity generally enhances the robustness of a network. In this respect, datacenter topologies typically provide multiple paths between two servers and/or server racks, which can be exploited for load balancing [74–76], among others.

As we move towards softwarized 5G environments, the load balancing problem is being extended in the context of multiple service instantiations that can span multiple NFVI PoPs. Application of state-of-the-art solutions that are originally designed

for the "centralized" nature of Cloud computing scenarios might lead to highly inefficient network configurations [77], where traffic may bounce between datacenters at any "load-balanced" component in the SC.

From this observation, the need becomes evident of load balancing techniques (perhaps closer to control-plane run-time decisions than to the MANO framework) able to consider the entire end-to-end SC deployment, and to optimally steer traffic where horizontally scaled copies of the same VNF/VNFC, or even other implementations of equivalent functionalities, can be exploited. A potential problem associated to such end-to-end load balancing techniques is the quantity of information to be maintained and synchronized among all the distributed elements concurring to the service implementation. Distributed optimization strategies (capable of mapping local information into dynamic control actions) can reduce the quantity of signaling to be exchanged for control purposes and scale better as a viable solution for large-scale systems like the upcoming 5G networks.

2.3.3 User and Service Mobility Management

With the current advancements in mobile Internet, users will be accessing services while on the move, and supporting user mobility becomes a concern among NSPs. While Cloud-Fog-MEC services have been embracing modular software design, VOs hosted in the Fog-MEC domain will have heterogeneous user proximity requirements (e.g., home environment and Content Delivery Network (CDN) virtualization use cases [78], mobile augmented reality applications [79], among others), and may have to "follow" the user at different paces.

A number of recent works gave different contributions to this user-service mobility problem. For instance, the *Follow Me Cloud* framework [80] proposed full/partial service migration by initiating/replicating individual components based on migration costs vs QoS/QoE trade-off, while [81] considered live migration, taking into account the dynamic user access patterns and migration amortization in the decision. As regards bulk live migration optimization, [82] and [83] focused on migration bandwidths and remapping of correlated VOs, respectively. However, they either consider only a single VO (or a small set of VOs) or neglect the inter-VO affinity, and more importantly, they did not study the scalability aspect of wide-area network reconfigurations during the migrations.

2.4 System Modeling and Analytics

In this section, we look at two most widely used system modeling and analytics approaches in the scientific literature – namely, machine learning (ML) and queueing theory.

2.4.1 Machine Learning

A large part of the state-of-the-art software-level modeling approaches use ML techniques on measurable PMCs. For instance, numerous PMC-based power models have already been proposed at the VM and core levels [84, 85], while [86] explores correlations between application-level QoS parameters like throughput and response time with the power readings from Intel's Running Average Power Limit (RAPL) interface [87]. Note that high levels of accuracies can be obtained with ML-based approaches, provided that the appropriate set of PMCs is considered and an extensive dataset is available for training.

2.4.2 Queueing Theory

Queueing theory has, over the years, proved to be a useful tool for modeling and analyzing telecommunication systems, and more recently, it is being adopted in the context of NFV as well. Most of the works in the literature regard estimating the system or queueing latencies towards efficient (QoS-aware) network service provisioning by considering network of queues to model interactions among virtual systems, SCs, VNFs and/or VNFCs, with each entity modeled as a (unique) single-arrival queueing system, with exponentially distributed inter-arrival times – for instance, using M/M/1 (or multi-class M/M/1), M/M/m and M/M/1/K queueing models [88–91].

With a more generalized arrival process, [92] considers an arbitrary distribution for the inter-arrival times; although packets still arrive singly on the network interface card (NIC), the authors delved deeper into the infrastructure level by trying to capture the *interrupt coalescing* (*IC*) operation on the NIC, resulting in batch arrivals at the CPU running the VNF.

The M^X/G/1/SET queue

Among the multitude of queueing models out there, a particularly interesting one is the M^X/G/1/SET queue [23], as it captures both the burstiness in the workload and the system setup times (e.g., due to IC, ACPI configuration, etc.). While rarely found in the literature, this model is not new yet among the published works (e.g., [23, 93–96]), some discrepancies are found among multiple expressions for the mean busy period and waiting time, calling for further analysis.

2.5 Summary

In this chapter, 5G is described as the next generation of telecommunication technologies as a whole, which is expected to support a wide-range of vertical applications (e.g., eMBB, mMTC and URLLC use cases) as network slices (or slice instantiations) on top of a unified infrastructure. The need to simultaneously handle such great diversity of requirements has further driven the current network softwarization revolution; hence,

it is only fitting to say that softwarization will be characteristic of upcoming 5G environments.

The key enablers for network softwarization that are considered in this thesis are the upcoming Cloud-Fog-MEC interplay, NFV and SDN. The first two are basically built on the concept of virtualization, where network functions/services can be implemented in software with high customizability and portability across the Cloud-Fog-MEC domain; the latter, on the other hand, deals with the programmable connectivity among (physical and/or virtual) network endpoints. Of course, despite the notable potential of the resulting softwarized environment, it still admits a number of open issues at the infrastructure and service levels, requiring the adoption of specific management and control solutions.

With an end goal of enabling scalable and sustainable softwarized 5G environments, this thesis engages in the following network/services management and control topics: a) power management in COTS hardware, b) dynamic management of Cloud-Fog-MEC resources, and c) user and service mobility management, through novel contributions in NFVI modeling and analytics, SDN-based network slicing and mobility management, as well as a more recent work in decentralized load balancing among multiple service instantiations. A review of related literature is presented to better position these contributions on the current technological scenario.

Chapter 3

NFV Infrastructure Modeling and Analytics

As previously described, NFV will play an important role in the current 5G revolution as it is foreseen to support the accelerated network service innovation rates with the software implementation of networking functionalities. However, some operational issues that stem from the underlying COTS hardware and virtualization solutions adopted need to be handled effectively and efficiently. On top of that, the highly modular and customizable nature of the virtualized network architecture manifests the need for automated management of NFVIs.

This chapter details a novel model-based analytics approach for NFVIs based on queueing theory; the $M^X/G/1/SET$ model is specifically considered to take into account the inherent workload burstiness and the system setup times (caused by interrupt coalescing and power management actions). The proposed approach seeks to augment the capabilities of a NFVI's VIM by enabling real-time VNF workload profiling and estimation of network KPIs (specifically, power and latency) using – and adding value to – available hardware/software PMCs.

In the following sections, a complete analytical characterization of the $M^X/G/1/SET$ queueing model is first presented, clarifying the discrepancies found in the scientific literature. Then, starting from available and easily measurable PMCs in Linux host servers, key model parameters are exposed for real-time VNF workload profiling, and ultimately, towards the estimation of the considered network KPIs.

3.1 Analytical Model

An energy-aware core hosting a VNF (or VNFC) is modeled as an $M^X/G/1/SET$ queue, which is a generalization of the well-known $M^X/G/1$ queue [97] for burst arrivals that also covers the cases in which an additional setup period SET is necessary before service can be resumed.

In more detail, batches of packets arrive at the system at exponentially distributed inter-arrival times with a random batch size X. If the system is empty at the arrival instant, SET is initiated; service only begins after the completion of SET. Packets are queued as they arrive and served individually with generally-distributed service times

TABLE 3.1: $M^X/G/1/SET$ model notation.

Symbol	Description
λ	batch arrival rate
β_j	probability that an incoming batch is composed of j packets
$X(z)$	PGF of the batch size, $X(z) = \sum_{j=1}^{\infty} \beta_j z^j$
$\beta_{(k)}$	k-th factorial moment of the batch size, $\beta_{(k)} = E\{\frac{X!}{(X-k)!}\} = \lim_{z\to 1} \frac{d^{(k)}X(z)}{dz^{(k)}}$
$\tau(t)$	probability density of the setup time
$\tau^*(\theta)$	Laplace transform of $\tau(t)$
$\tau_{(i)}$	i-th moment of the setup time, $\tau_{(i)} = (-1)^i \frac{d^{(i)}\tau^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
$s(t)$	probability density of the packet service time
$s^*(\theta)$	Laplace transform of $s(t)$
$s_{(i)}$	i-th moment of the packet service time, $s_{(i)} = (-1)^i \frac{d^{(i)}s^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
μ	average packet service rate, $\mu = 1/s_{(1)}$
ρ	server utilization, $\rho = \frac{\lambda\beta_{(1)}}{\mu}$
$B^*(\theta)$	Laplace transform of the busy period density
$B_{(i)}$	i-th moment of the busy period, $B_{(i)} = (-1)^i \frac{d^{(i)}B^*(\theta)}{d\theta^{(i)}}\big\vert_{\theta=0}$
$P(z)$	PGF of the number of packets in the system at a random epoch
W	average waiting time in the queue

S. Moreover, we approximate the loss probability in a queue with finite buffer N by the stationary probability that the number n of customers in the infinite-buffer queue at a generic time t be greater than N ($Pr\{n > N\}$). Therefore, hereinafter, we will consider the infinite buffer case.

More details on the different model components are presented in this section – from the arrival, setup and service processes, to the characterization of network KPIs. The model notation is given in Table 3.1.

3.1.1 Traffic Model

In telecommunication networks, where burst packet arrivals are more representative of the traffic behavior rather than single arrivals, effectively capturing the traffic burstiness is essential. The Batch Markov Arrival Process (BMAP) has long been established in this respect [98, 99]; BMAP allows for dependent and non-exponentially distributed packet inter-arrival times, while keeping the tractability of the Poisson process [100].

We suppose that packets arrive according to a BMAP with batch arrival rate λ. Then, to characterize the random batch size X, let β_j be the probability that an incoming batch

is composed of j packets ($j = 1, 2, \ldots$). Then, the probability generating function (PGF) of X is given by

$$X(z) = \sum_{j=1}^{\infty} \beta_j z^j \tag{3.1}$$

from which the first and second factorial moments of the batch size are obtained as $\beta_{(1)} = \sum_{j=1}^{\infty} j\beta_j$ and $\beta_{(2)} = \sum_{j=1}^{\infty} j^2\beta_j - j\beta_j$, respectively. The offered load in packets per second (pps) is then obtained as $OL = \lambda\beta_{(1)}$.

Given λ, $\beta_{(1)}$ and $\beta_{(2)}$, the burstiness of the traffic can already be well estimated. However, a common difficulty stems from the fact that the discrete probability distribution $\{\beta_j, j = 1, 2, \ldots\}$ may not be given, and typically requires detailed analysis of packet-level traces [53, 54]. As an alternative approach, we propose to estimate the factorial moments from easily measurable parameters (e.g., VNF workload, idle and busy times), which will be discussed in Section 3.2.

3.1.2 Setup Model

With the deterministic nature of the setup period due to core wakeup transitions and reconfigurations considered in this work, the Laplace transform of the probability density $\tau(t)$ can be reduced to

$$\tau^*(\theta) = e^{-\tau\theta} \tag{3.2}$$

From this, the first and second moments of the setup time are simply given by $\tau_{(1)} = \tau$ and $\tau_{(2)} = \tau^2$, respectively, with

$$\tau = \begin{cases} \tau_p & \text{in the context of power consumption, and} \\ \tau_l & \text{in the context of latency.} \end{cases}$$

3.1.3 Service Model

Once setup is completed, the core starts to serve backlogged packets and remains busy until the system becomes empty. We suppose that the VNF (or VNFC) hosted on the core has deterministic service times.

Service process

Generally, multiple execution containers (e.g., VMs, Linux containers, etc.) may be consolidated on the same core, and the service process can be captured by a discrete set of service rates $\{\mu_m,\ m \in M\}$ with corresponding probabilities $\{\pi_m,\ m \in M\}$, where M is the set of execution containers sharing the core. The Laplace transform of the probability density $s(t)$ is then obtained as

$$s^*(\theta) = \sum_{m \in M} \pi_m e^{-\frac{\theta}{\mu_m}} \tag{3.3}$$

However, in the special case of one-to-one correspondence between cores and execution containers, Eq. (3.3) reduces to $s^*(\theta) = e^{-\frac{\theta}{\mu}}$, giving the first and second moments of the service time as $s_{(1)} = 1/\mu$ and $s_{(2)} = 1/\mu^2$, respectively. Note that the core utilization due to actual packet processing is obtained as $\rho = OL/\mu$ (< 1 for system stability).

Busy period analysis

Adopting the approach presented in [101], the busy period B of an $M^X/G/1/SET$ queue is decomposed into two components: (a) the initial busy period B_τ – in which all the packets that arrived during the setup SET are served, and (b) the ordinary busy period B_X that corresponds to the busy period of an $M^X/G/1$ queue – in which the batch initiating the setup and the rest that arrived while the core is busy are served. Considering that the busy period density $B(t)$ is given by the convolution of the probability densities $B_\tau(t)$ and $B_X(t)$, then its Laplace transform is simply obtained as the product $B^*(\theta) = B^*_\tau(\theta)B^*_X(\theta)$.

Let the random variable η_τ denote the number of batch arrivals during SET. Given that $SET = t$ and $\eta_\tau = m$, then B_τ is distributed as the sum of the lengths of m independent ordinary busy periods $B_{X1}, \ldots, B_{Xm}$ [97]. By first conditioning on SET and η_τ, and then averaging out, we obtain the Laplace transform of $B_\tau(t)$ as

$$\begin{aligned} B^*_\tau(\theta) &= \int_0^\infty \sum_{m=1}^{\infty} e^{-\lambda t}\frac{(\lambda t)^m}{m!} E\left\{e^{-\theta[B_{X1}+\ldots+B_{Xm}]}\right\}\tau(t)\,dt \\ &= \tau^*\left(\lambda - \lambda B^*_X(\theta)\right) \end{aligned} \tag{3.4}$$

Similarly, let the random variables S_{X_1}, X_1 and η_1 denote the service time of the initiating batch, the number of packets in this batch, and the number of batch arrivals during S_{X_1}, respectively. Given that $S_{X_1} = t$, $X_1 = j$ and $\eta_1 = n$, then in the same way as before, B_X is distributed as the sum of the lengths of t and n independent ordinary busy periods. By conditioning on S_{X_1}, X_1 and η_1, and proceeding as before, we obtain the Laplace transform of $B_X(t)$ as

$$\begin{aligned} B^*_X(\theta) &= \sum_{j=1}^{\infty}\beta_j \int_0^\infty \sum_{n=1}^{\infty} e^{-\lambda t}\frac{(\lambda t)^n}{n!} E\left\{e^{-\theta[t+B_{X1}+\ldots+B_{Xn}]}\right\} s_1(t) * \cdots * s_j(t)\,dt \\ &= \sum_{j=1}^{\infty}\beta_j s^*\left(\theta+\lambda-\lambda B^*_X(\theta)\right)^j \\ &= X\left(s^*\left(\theta+\lambda-\lambda B^*_X(\theta)\right)\right) \end{aligned} \tag{3.5}$$

Specializing to the case of deterministic setup and service times, the Laplace transform of the busy period density $B(t)$ is given by

$$B^*(\theta) = e^{-\lambda\tau(1-B^*_X(\theta))} X\left(e^{-\frac{\theta+\lambda-\lambda B^*_X(\theta)}{\mu}}\right) \tag{3.6}$$

obtaining the first and second moments of the busy period as $B_{(1)} = \frac{(\beta_{(1)}/\mu)+\rho\tau}{1-\rho}$ and $B_{(2)} = \frac{(\beta_{(1)}+\beta_{(2)})(1+\lambda\tau)}{\mu^2(1-\rho)^3} + \frac{\rho\beta_{(1)}(2\tau+\lambda\tau^2)}{\mu(1-\rho)^2}$, respectively. It is important to note that these expressions are particularly useful for estimating the power consumption and system latency, as we shall see further on.

System state probabilities

The PGF $P(z)$ of the number of packets in an $M^X/G/1/SET$ queueing system at a random epoch can be expressed as the product $P(z) = P_\tau(z)P_X(z)$, with $P_\tau(z)$ being the PGF of the number of individual packet arrivals during the residual life of the vacation period (i.e., idle period I plus the setup SET) and

$$P_X(z) = \frac{(1-\rho)(1-z)s^*\big(\lambda-\lambda X(z)\big)}{s^*\big(\lambda-\lambda X(z)\big)-z} \tag{3.7}$$

the well-known PGF of the number of packets at a random epoch in the ordinary $M^X/G/1$ system.

Let $\varphi(z) = X(z)\tau^*\big(\lambda-\lambda X(z)\big)$ be the PGF of the number of packet arrivals during a generic vacation period, from which we obtain the average number of packets by the end of SET as $\varphi_{(1)} = \frac{d\varphi(z)}{dz}|_{z=1} = \beta_{(1)}(1+\lambda\tau_{(1)})$. By simplifying the general expression found in [102], we arrive to

$$P_\tau(z) = \frac{1-X(z)\tau^*\big(\lambda-\lambda X(z)\big)}{(1+\lambda\tau_{(1)})\big(1-X(z)\big)} \tag{3.8}$$

Again, specializing the resulting general expression of $P(z)$ to the case of deterministic setup and service times, we obtain

$$P(z) = \frac{1-X(z)e^{-\lambda\tau(1-X(z))}}{(1+\lambda\tau)\big(1-X(z)\big)} \frac{(1-\rho)(1-z)e^{-\frac{\lambda(1-X(z))}{\mu}}}{e^{-\frac{\lambda(1-X(z))}{\mu}}-z} \tag{3.9}$$

3.1.4 Power Model

The power model works according to a renewal process, where the idle (I) and delay busy ($SET + B$) periods constitute independent and identically distributed (iid) "cycles" (R), as illustrated in Fig. 3.1. A delay busy period, as defined in [23], starts with the arrival of the batch initiating the setup, and ends with the depature of the last packet in the system.

Based on classical renewal theory principles, the steady-state behavior of the stochastic process can be studied by looking at a representative cycle [97]. With this in mind, the average power consumption of the core is expressed as a sum of the average contributions incurred during the idle, setup (i.e., due to wakeups) and busy periods, $\Phi = \frac{I_{(1)}}{R_{(1)}}\Phi_i + \frac{\tau_{(1)}}{R_{(1)}}\Phi_w + \frac{B_{(1)}}{R_{(1)}}\Phi_a$, where $R_{(1)} = I_{(1)} + \tau_{(1)} + B_{(1)}$ is the average length of a renewal cycle, and $I_{(1)}$ is the average length of an idle period. Specializing these

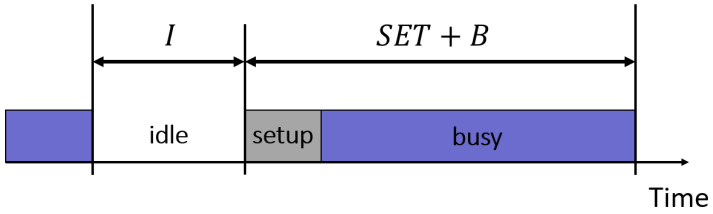


FIGURE 3.1: A generic renewal cycle in a $M^X/G/1/SET$ system.

to the case of BMAP arrivals (i.e., $I_{(1)} = \frac{1}{\lambda}$), and deterministic setup and service times (i.e., $\tau_{(1)} = \tau$ and $B_{(1)} = \frac{(\beta_{(1)}/\mu)+\rho\tau}{1-\rho}$), we obtain $R_{(1)} = \frac{1+\lambda\tau}{\lambda(1-\rho)}$; then

$$\Phi = \frac{1-\rho}{1+\lambda\tau}\Phi_i + \frac{\lambda\tau(1-\rho)}{1+\lambda\tau}\Phi_w + \rho\Phi_a \tag{3.10}$$

3.1.5 Latency Model

The system latency D is derived as the sum of the average waiting time W of a packet in the queue and its average service time (i.e., $s_{(1)} = 1/\mu$).

The well-known Little's law defines the former as $W = L/\lambda\beta_{(1)}$, where L is the average length of the queue that can be derived from $P(z)$ (see Sub-section 3.1.3) as $L = \frac{dP(z)}{dz}\big|_{z=1} - \rho$; from this we obtain

$$W = \frac{\rho\beta_{(1)} + \beta_{(2)}}{2\beta_{(1)}\mu(1-\rho)} + \frac{2\tau + \lambda\tau^2}{2\beta_{(1)}(1+\lambda\tau)} \tag{3.11}$$

and hence,

$$D = \frac{\rho\beta_{(1)} + \beta_{(2)}}{2\beta_{(1)}\mu(1-\rho)} + \frac{2\tau + \lambda\tau^2}{2\beta_{(1)}(1+\lambda\tau)} + \frac{1}{\mu} \tag{3.12}$$

It is interesting to note that Eq. (3.11) can also be derived starting from the Laplace transform of the waiting time density, as in [96].

3.2 Model-based Analytics

With the $M^X/G/1/SET$ core model as the basis, we seek to facilitate scalable and sustainable network/service management and orchestration mechanisms, through a novel model-based analytics approach for profiling VNF workloads and estimating network KPIs, which is developed in an experimental setting. In this section, we describe the system under test (SUT) and expose the key model parameters from available and easily measurable PMCs.

3.2.1 System Description

Considering that the system behavior highly depends on the ACPI configuration of the host, as well as the virtualization and I/O technologies used in the VNF implementation, more details on these aspects are provided in the following sub-sections.

ACPI configuration

For a given (C_x, P_y) pair, a number of power- and performance-aware parameters can be defined.

Firstly, the instantaneous power requirements vary with the core's state. Specific values of idle (Φ_i) and active (Φ_a) power consumptions are associated with each available power and performance state, respectively. Moreover, transitions between C_0 and C_x are not instantaneous; hence, the power consumed in these periods must also be taken into account. Since the average power consumption during sleeping transitions ($C_0 \rightarrow C_x$) approximates Φ_i, we only consider the wakeup transitions ($C_x \rightarrow C_0$) in this work. Particularly, the power spike in the latter is associated with a wakeup power consumption Φ_w that is approximately $2.5\Phi_a$, as pointed out in [103].

Meanwhile, the total delay experienced by packets can be broken down into contributions of different system operations. As packets arrive at the RX queue, NICs may wait either for some time interval (i.e., time-based IC) or number of arrival events (i.e., frame-based IC) before raising interrupts to notify the core of pending work; we consider the time-based IC at the NIC, for which we define the period τ_{ic}. Generally, such service requests can occur while the core is in idle or active mode; in the former, there is an additional setup period due to the wakeup and reconfiguration operations before the actual packet processing begins. At the core level, we consider two setup contributions (i.e., due to wakeup and due to reconfiguration), for which we define the periods τ_p and τ_r, respectively.

Note that for the Sandy Bridge EP platform, core wakeup latencies are in the order of nano/microseconds [104], and the value of τ_p depends on the core's power state C_x. Once in active mode, the core then performs some reconfiguration operations (i.e., supposed to include context switching); nonetheless, the value of τ_r depends on the core's performance state P_y. In the context of power and latency modeling, τ_p and $\tau_l = \tau_{ic} + \tau_p + \tau_r$ will be considered, respectively.

After the completion of the setup period, backlogged packets are suppose to be served exhaustively (considering that packets are already transferred in the main memory via standard Direct Memory Access (DMA)) with an average processing capacity μ, which corresponds to the operating energy point of the performance state P_y.

VNF implementation

In order to minimize the dependence of the proposed approach on the virtualization and I/O technologies, we consider a traditional VM-based VNF implementation in this development.

Particularly, as virtualization extensions are made available in x86 hardware, Kernel-based Virtual Machine (KVM) [105] – a full virtualization solution for Linux – has gained much popularity for its simplicity and ability to run VMs with unmodified guest OSs. KVM uses the Linux kernel as bare-metal hypervisor and has been integrated in

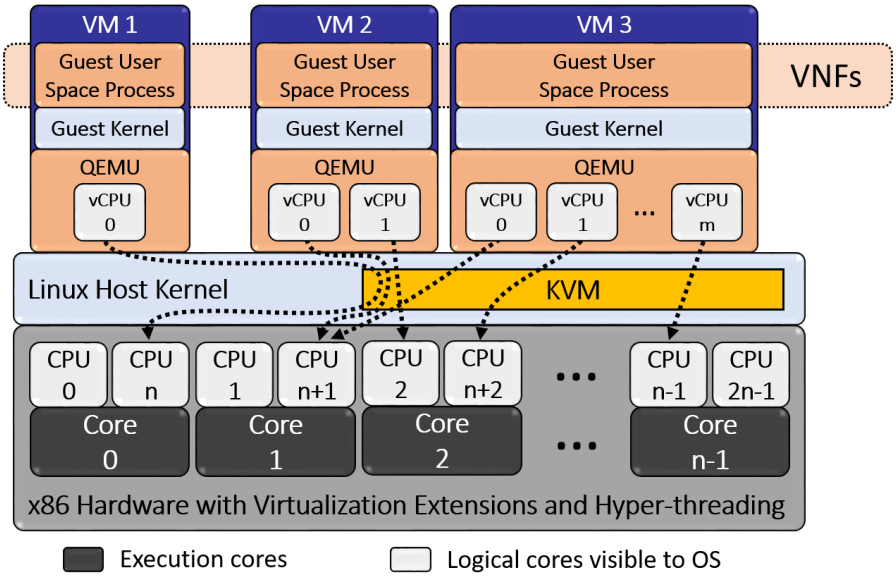


FIGURE 3.2: Traditional KVM-based VNF implementation.

the kernel since the 2.6.20 release, making it the default virtualization mechanism recommended for most Linux distributions [106]. In this respect, KVM virtualization is the basis of this work, but the approach can be also applied to (or easily adaptable for) other platforms.

Fig. 3.2 illustrates the traditional KVM architecture for VNF implementation. Guest networking is implemented by the user space process Quick Emulator (QEMU) – a detailed description of the network I/O path can be found in [107]. VM processes are allocated a certain number of vCPUs, each one seen as a physical CPU by the guest OS. Then, VNFs run as guest user space processes in the corresponding VMs.

For simplicity, but without loss of generality, we suppose a one-to-one correspondence between the VM and the core to match the core workload, utilization, power consumption and latency with those of the VNF. More complex VNFs may consist multiple VMs – each one running a VNFC, but the overall performance can be derived from the individual performances of the components and their chaining.

Note that with such a traditional VM-based implementation, switching between VNF and interrupt handler codes can be rather costly. To reduce this overhead, the interrupt and VM process affinities are set to different cores, such that the core tasked with interrupt handling notifies the one running the VM process via an inter-processor interrupt (IPI) for backlogged packets. Moreover, as also pointed out in [108], setting affinities or core pinning in such fashion improves the energy efficiency of the system.

A number of parameters can be tuned in the NIC using the `ethtool` command [109]. In order to preserve as much as possible the shapes of the incoming/outgoing traffic of the VNF, we look into the IC and RX/TX ring parameter settings. For the former, we decided to keep the default settings since they are more or less equivalent with respect to the generated input traffic – specifically, options for adaptive IC (i.e., `adaptive-rx` and `adaptive-tx`) are off, the parameters for frame-based IC (i.e.,

`rx-frames` and `tx-frames`) are set to 0, and the parameters for time-based IC (i.e., `rx-usecs` and `tx-usecs`) are set to 3 μs and 0, respectively. On the other hand, the RX/TX ring buffer sizes are set to the pre-set maximums (i.e., 4096) in order to maximize the NIC's ability to handle burst arrivals.

3.2.2 Exposing Model Parameters

Linux has different utilities for performance monitoring – among them, we consider the PMCs described in the following. Note that in the syntax of these utilities, the term 'CPU' refers to a core (or logical core, in the case of hyperthreading).

Idlestat: As a tool for CPU power/performance state analysis, the `idlestat` command [110] in trace mode is able to monitor and capture the $C-$ and $P-$ state transitions of CPUs over a user-defined interval. To run it in trace mode, the `-trace` option is used together with the `<filename>` and `<time>` parameters to specify the trace output filename and the capture interval in seconds, respectively. With the `-c` and `-p` options, $C-$ (including the *POLL* state, in which the CPU is idle but did not yet enter a power state) and $P-$ states statistics are reported in terms of the time spent in each state per CPU.

VnStat: As a network traffic monitoring tool, the `vnstat` command [111] is able to report how much traffic (in terms of average rates) goes through a specific interface over a user-defined interval. This is done by using the `-tr` option together with the `<time>` parameter to specify the monitoring interval in seconds, and the `-i` option together with the `<interface>` parameter to specify the interface.

Starting from the PMCs provided by these tools, we seek to expose key model parameters for a given (C_x, P_y) pair, in a sort of black-box measurement/estimation approach, as detailed in the following sub-sections, and in effect profiling the VNF workloads. Starting from this, the corresponding power consumption and system latency can then be readily derived using Eqs. (3.10) and (3.12), respectively.

Offered load and utilization

Measuring OL with the `vnstat` command is quite straightforward, as it corresponds to the rate of incoming traffic on the network interface bound to the VM process.

On the other hand, the utilization measurable with the `idlestat` command as

$$\tilde{\rho} = \frac{\tilde{T}_{P_y}}{\tilde{T}_{P_y} + \tilde{T}_{C_x} + \tilde{T}_{POLL}} \tag{3.13}$$

where $\tilde{T}_{P_y}$, $\tilde{T}_{C_x}$ and $\tilde{T}_{POLL}$ are the measured average times spent in the corresponding states, encompassing all operations (including reconfigurations, context switching, sleep transition, etc.) performed by the active core. While this gives indications on the utilization overhead incurred with this VNF implementation, we are also interested in

estimating the utilization due to actual packet processing, for which we consider

$$\hat{\rho} = \tilde{OL}/\mu \tag{3.14}$$

where $\tilde{OL}$ is the offered load measured with the `vnstat` command.

Batch arrival rate

With exponentially distributed inter-arrival times, λ can be estimated from the average idle times measurable using the `idlestat` command. Theoretically, $\tilde{I}_{(1)} \approx \tilde{T}_{C_x} + \tilde{T}_{POLL}$; however, with the high variance observed on $\tilde{T}_{POLL}$, we propose to consider only T_{C_x} for stable estimates. Linear regression is then used to compensate for the discrepancy, giving

$$\hat{\lambda} = \frac{\alpha_1}{T_{C_x}} + \alpha_0 \tag{3.15}$$

where α_1 and α_0 are the computed regression coefficients.

Factorial moments of the batch size

Given $\tilde{OL}$ and $\hat{\lambda}$, $\beta_{(1)}$ can be directly estimated, by definition, as

$$\hat{\beta}_{(1)} = \frac{\tilde{OL}}{\hat{\lambda}} \tag{3.16}$$

Although estimating $\beta_{(2)}$ is a bit more involved, it is essential for estimating the latency. In this regard, we propose to start from the expression of the second moment of the busy times $B_{(2)}$ (see Sub-section 3.1.3), and consider

$$\hat{\beta}_{(2)} = max\left(0, \left(\frac{\mu^2}{1+\hat{\lambda}\tau_l}\right)\left(\hat{B}_{(2)}(1-\hat{\rho})^3 - \frac{\hat{\beta}_{(1)}(1+\hat{\lambda}\tau_l)}{\mu^2} - \frac{\hat{\rho}(1-\hat{\rho})\hat{\beta}_{(1)}(2\tau_l+\hat{\lambda}\tau_l^2)}{\mu}\right)\right) \tag{3.17}$$

with the *max* function ensuring $\hat{\beta}_{(2)} \geq 0$. Now we are left with how to obtain $\hat{B}_{(2)}$, for which we adopt the well-known theorems in statistics regarding the mean and variance of sample means.

In more detail, let $\{B^1_{(1)}, B^2_{(1)}, \ldots, B^\eta_{(1)}\}$ be a random sample of size η obtained from a busy period distribution with mean $B_{(1)} = E\{B\}$ and variance $var(B) = B_{(2)} - (B_{(1)})^2$. Supposing that $B^n_{(1)}$, $n = 1, \ldots, \eta$, are iid, we consider the standard estimators $\hat{B}_{(1)} = \frac{1}{\eta}\sum_{n=1}^{\eta} B^n_{(1)}$ and $\hat{var}(B_{(1)}) = \frac{1}{\eta-1}\sum_{n=1}^{\eta}\left(B^n_{(1)} - \hat{B}_{(1)}\right)^2$, and estimate the variance of the busy times as $\hat{var}(B) = \Delta t \hat{var}(B_{(1)})$ starting from sample means [112], where Δt is the observation period over which each sample of $B^n_{(1)}$ is obtained. Then, we can estimate the second moment of the busy times as

$$\hat{B}_{(2)} = \Delta t \hat{var}(B_{(1)}) + (\hat{B}_{(1)})^2 \tag{3.18}$$

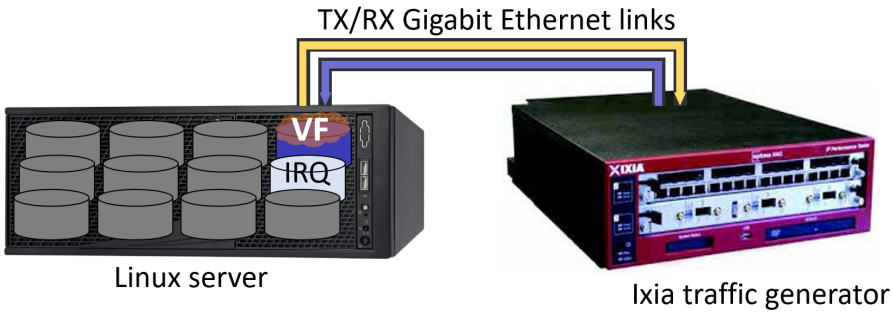


FIGURE 3.3: Analytics experimental testbed.

In this work, we consider $\tilde{B}_{(1)} \approx \tilde{T}_{P_y} - \Delta T$, where ΔT is the busy overhead due to operations other than actual packet processing (which includes τ_r, context switching, sleep transitions, etc.). Eq. 3.18 is then applied on the set of samples $\{\tilde{B}^n_{(1)}, n = 1, \ldots, \eta\}$ to obtain an estimate of $B_{(2)}$.

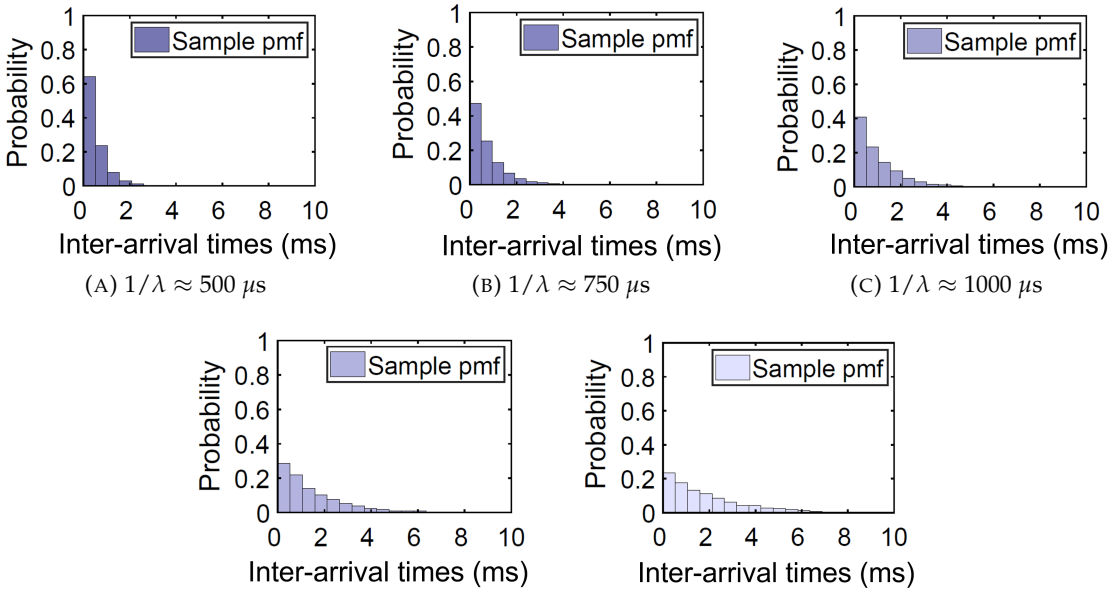
3.3 Experimental Results

The proposed model-based analytics approach is evaluated considering a SUT equipped with two Intel® Xeon® E5-2643 v3 3.40GHz processors, running an OpenWrt [113] virtual firewall (VF). The latter is pinned (or with affinity set) to a single core and the interrupt request (IRQ) handling to another one. The SUT is connected via RX/TX Gigabit Ethernet links to an Ixia NX2 traffic generator, as shown in Fig. 3.3. The setup creates a controlled environment that allows monitoring of the SUT's PMCs, power consumption and system latency, as the burstiness of incoming traffic is varied.

The ACPI configuration of the pinned core is set to (C_{1E}, P_{0T}) to maximize the system throughput, where the power state C_{1E} corresponds to the *Enhanced Halt* – the lightest sleeping state with improved power requirements, and the performance state P_{0T} to the maximum turbo frequency; while the rest of the cores are put to deep power saving state C_6. Under this configuration, we approximate the values of the following model parameters as: $\mu \approx 199628$ pps, $\tau_{ic} \approx 3$ μs, $\tau_p \approx 10$ μs, $\tau_r \approx 10$ μs, $\Phi_i \approx 8.33$ W, $\Phi_a \approx 53.38$ W and $\Phi_w \approx 133.45$ W.

In more detail, the processing capacity is estimated as the maximum system throughput measured with the `vnstat` command. Setup components are derived from the IC configuration, wake-up latencies specified in the kernel's `cpuidle sysfs` and the results of [114]. Power related parameters are estimated starting from actual package-level (i.e., core part) measurements obtained with the `turbostat` command [115] – one of the many tools that expose power measurements from Intel's RAPL interfaces; [116] confirms from extensive tests that RAPL exposes true averages that are updated at fine-grained intervals.

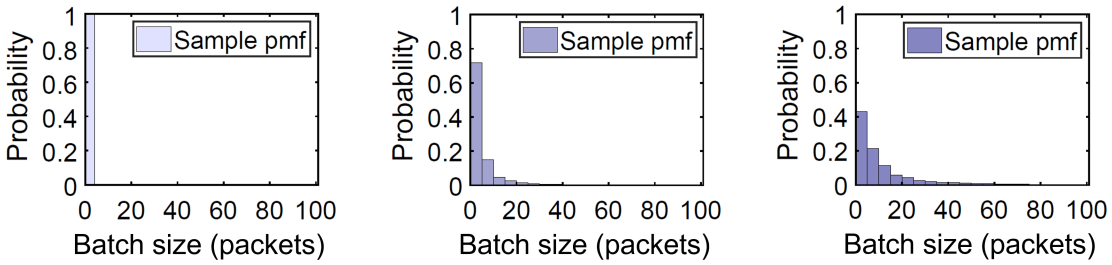
Input traffic (composed of 64-byte Ethernet frames) is generated from the Ixia traffic generator by considering $1/\lambda \in \{500, 750, 1000, 1500, 2000\}$ μs and $\beta_{(1)} \in \{1, 6, 12\}$



(A) $1/\lambda \approx 500\ \mu s$ (B) $1/\lambda \approx 750\ \mu s$ (C) $1/\lambda \approx 1000\ \mu s$

(D) $1/\lambda \approx 1500\ \mu s$ (E) $1/\lambda \approx 2000\ \mu s$

FIGURE 3.4: Distributions of the input batch inter-arrival times.



(A) $\beta_{(1)} \approx 1$ packet (B) $\beta_{(1)} \approx 6$ packets (C) $\beta_{(1)} \approx 12$ packets

FIGURE 3.5: Distributions of the input batch sizes.

packets; details on the BMAP emulation approach are described in the following subsection.

Lastly, the results obtained with the model-based approach are validated with respect to the inputs (for the workload profiling) and actual measurements (for the network KPI estimation); for each test point, 100 samples are collected, from which the 99% confidence intervals are obtained and indicated with error bars.

3.3.1 Generating BMAP Arrivals

The input traffic is generated to emulate BMAP arrivals by using `Tcl` scripts to specify inter-arrival times and batch sizes. Realizations of the inter-arrival times are drawn from exponential distributions (setting the desired mean value), while those of batch sizes from truncated generalized Pareto distributions (varying the *scale* and *location* parameters to approximate the desired mean value). The probability mass functions (pmfs) of the resulting inter-arrival times and batch sizes are illustrated in Figs. 3.4 and 3.5, respectively.

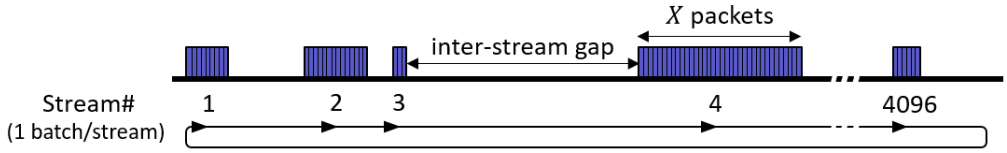


FIGURE 3.6: Emulating BMAP arrivals.

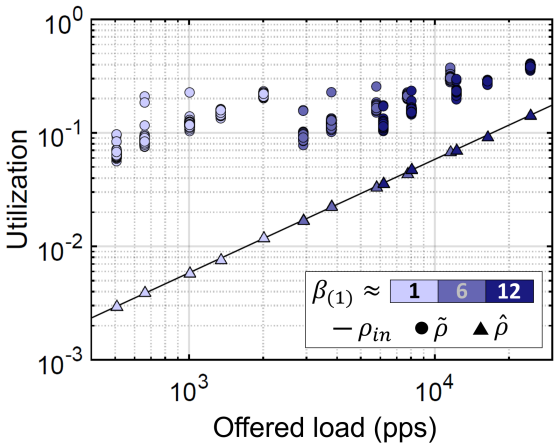


FIGURE 3.7: Core utilization for varying VNF workload burstiness.

In the `Tcl` scripts, each batch (of X packets) is assigned to a stream, as well as an inter-stream gap (ISG) that approximates the inter-arrival time in the model. Starting from the first stream/batch, the next one is generated after the specified ISG, and so on. Then, the system loops back to the first stream/batch after the ISG of the last one, as shown in Fig. 3.6. Since the traffic generator allows up to 4096 streams when using Gigabit ports, and only 512 streams when using 10 Gigabit ports, we use the former to achieve better approximations of the batch size and inter-arrival time distributions.

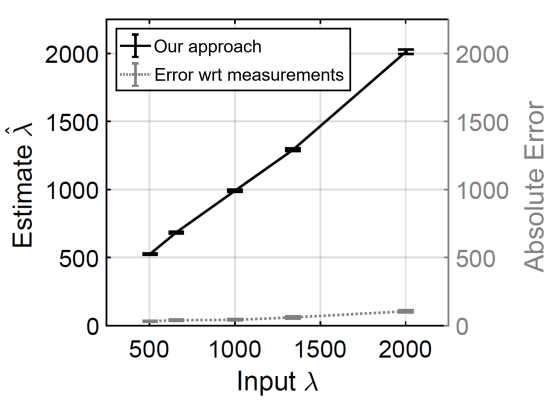
3.3.2 Validation of Workload Profiling

As initially motivated, the proposed approach seeks to profile VNF workloads beyond offered loads and utilization – specifically, to capture the workload burstiness with the model parameters λ, $\beta_{(1)}$ and $\beta_{(2)}$.

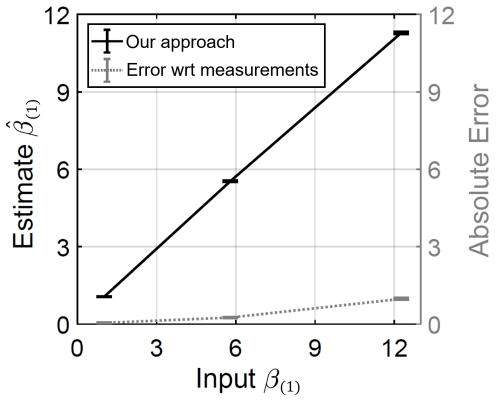
Offered load and utilizaion

Using the `vnstat` command, we obtain $\tilde{OL}$ with maximum and mean absolute percent error of 5.69% and 2.15%, respectively. Looking at Eq. (3.14), the same accuracy is expected for $\hat{\rho}$ with a constant value for μ.

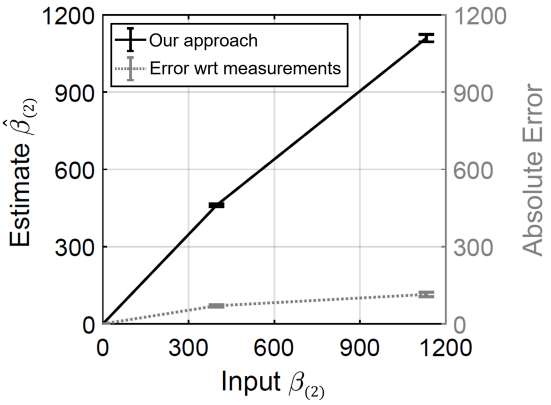
Fig. 3.7 shows a comparison between the measured ($\tilde{\rho}$) and estimated ($\hat{\rho}$) utilization values, with values computed from the input model parameters (ρ_{in} – the utilization due to actual packet processing). It can be observed how $\hat{\rho}$ matches with the model, while $\tilde{\rho}$ exhibits an overhead that is correlated with the batch size; this further motivates the need to capture the workload burstiness.



(A) batch arrival rate



(B) 1st factorial moment of X



(C) 2nd factorial moment of X

FIGURE 3.8: Estimating key statistical features of the workload burstiness.

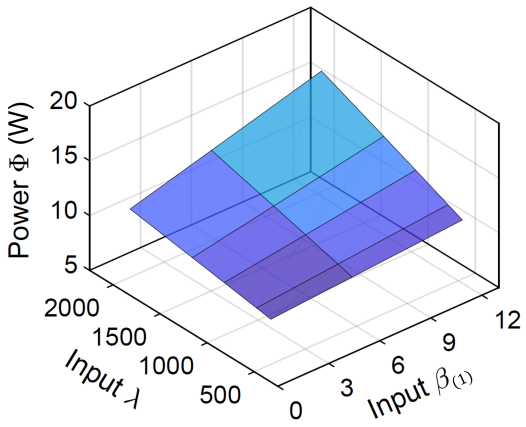
Burstiness

With BMAP emulation, the `idlestat` command gave reliable estimates for λ, which also comply with the theory of Poisson processes, while the software PMCs initially considered in [117] failed. Fig. 3.8a shows the estimates obtained based on Eq. (3.15), with $\alpha_1 = 0.910651$ and $\alpha_0 = -60.418339$. Tight confidence intervals and low absolute errors are achieved, even with varying values of $\beta_{(1)}$ aggregated in each test point.

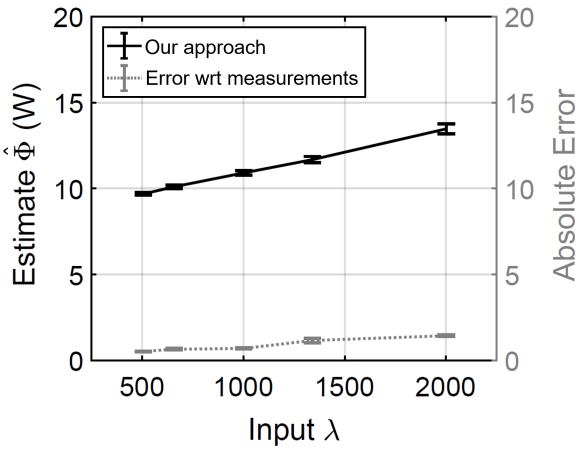
A similar level of accuracy is expected with $\beta_{(1)}$ estimates as they are solely based on $\tilde{OL}$ and $\hat{\lambda}$, as indicated in Eq. (3.16); the obtained estimates are shown in Fig. 3.8b. On the other hand, Fig. 3.8c shows the $\beta_{(2)}$ estimates obtained based on Eq. (3.17). Although the input and estimated values follow the same trend, a relatively higher variance (and hence, errors) are observed in $\hat{\beta}_{(2)}$ stemming from the fact that the starting point was the busy times (i.e., Eq. (3.18)), which also depend on other model parameters.

3.3.3 Validation of Network KPI Estimation

In this sub-section, we apply the results obtained from the workload profiling to the real-time estimation of network KPIs – specifically, the VNF power consumption and



(A) Power model



(B) Power estimates vs measurements

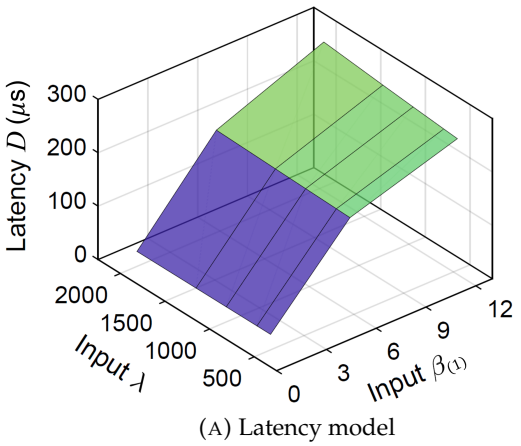
FIGURE 3.9: Estimating the power consumption.

incurred latency. Then, the estimates obtained from the power (i.e., Eq. (3.10)) and latency (i.e., Eq. (3.12)) models are compared with actual measurements.

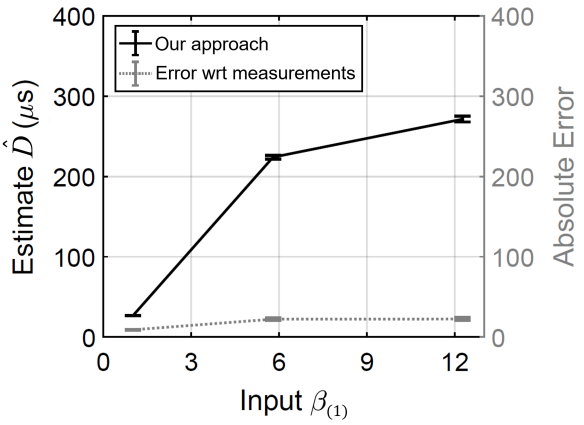
Power

We suppose that the core power consumption due to the VNF can be obtained as $\tilde{\Phi} = \tilde{\Phi}_{cpkg} - \Delta\Phi$, where $\tilde{\Phi}_{cpkg}$ is the power consumed by the core part of the package (measurable with the `turbostat` command), and $\Delta\Phi$ is the overhead due to the other cores in the package. Recalling that the VNF is pinned to a core under (C_{1E}, P_{0T}), and the rest of the cores are under (C_6, P_{0T}), we consider $\Delta\Phi \approx 15.93$ W in this work.

Fig. 3.9a illustrates the behavior of the power model for varying traffic burstiness. Intuitively, by looking at Eq. (3.10), the average core power consumption is linearly dependent to both λ and $\beta_{(1)}$ (embedded in ρ), although a stronger correlation is observed with the former. Results on the model-based power estimation, and its comparison with the actual measurements (in terms of absolute error) are shown in Fig. 3.9b.



(A) Latency model



(B) Latency estimates vs measurements

FIGURE 3.10: Estimating system latency.

Latency

We suppose that the system latency can be obtained as $\tilde{D} = \tilde{D}_{ixia} - \Delta D$, where $\tilde{D}_{ixia}$ is the store-and-forward latency measurable from the user interface of the traffic generator, and ΔD is the overhead due to the 2-way transmission on a Gigabit link (i.e., $\approx 2\beta/1488095$) plus the busy overhead $\Delta T \approx 90$ μs that includes τ_r, context switching (i.e., for which [92] proposed a rule of thumb of 30 μs), sleep transitions, etc.

Fig. 3.10a illustrates the behavior of the latency model for varying traffic burstiness. Contrary to the power consumption, the average VNF latency incurred is more strongly linked to $\beta_{(1)}$ (and $\beta_{(2)}$) than to λ. Results on the model-based latency estimation, and its comparison with the actual measurements (in terms of absolute error) are shown in Fig. 3.10b.

3.4 Summary

In this chapter, the $M^X/G/1/SET$ queue is considered to model an energy-aware core hosting a VNF (or VNFC), as it captures both the inherent workload burstiness of

telecommunication applications, and the system setup times caused by interrupt coalescing and ACPI configuration on the underlying physical resources. A complete analytical characterization of the model is presented in order to clarify the discrepancies found in the scientific literature.

With the $M^X/G/1/SET$ model as the basis, a novel analytics approach is proposed for the real-time VNF workload profiling, and ultimately, towards the estimation of network KPIs (specifically, power and latency). In particular, the following model parameters are exposed from available and easily measurable PMCs in Linux host servers: a) offered load, b) utilization, c) batch arrival rate, and d) the factorial moments of the batch size, which are then used to derive the power consumption and system latency of the VNF.

Experimental evaluations are performed on a SUT equipped with Intel® Xeon® E5-2643 v3 3.40GHz processors, with input traffic generated to emulate BMAP arrivals, through scripting in an Ixia NX2 traffic generator. Results show good estimation accuracies for both VNF workload profiling and network KPI estimation, with respect to the input traffic and actual measurements, respectively. This demonstrates how the proposed approach can be a powerful tool, not only for augmenting the capabilities of a NFVI's VIM, but also in the development of next-generation resource/service provisioning solutions, among others.

Chapter 4

SDN-based Network Slicing and Mobility Management

We have seen that network softwarization will be characteristic of upcoming 5G environments: with the Cloud-Fog-MEC interplay, NFV and SDN, a unified, multi-domain and multi-tenant infrastructure will emerge. In this scenario, SDN could (potentially) provide the network programmability levels required to enable network/service "agility". However, as previously anticipated, an open issue with the latter regards scalability, which stems from the smaller capacities of TCAMs (i.e., finite-sized rule tables) in SDN devices with respect to the binary CAMs in legacy routers.

This chapter presents the *Multi-Cluster Overlay (MCO)*, a novel SDN-based mechanism specifically designed to realize geo-distributed VTNs and effectively support dense deployments of mobile VOs at the network edge, as well as seamless user/services mobility through bulk inter-datacenter VO live migrations, in a highly scalable fashion, while relaxing any (resource/functional) requirements at the OF switches in the network infrastructure. The proposed approach seeks to enable network/service agility with significantly reduced number of OF rules and low computational overhead.

In the following sections, we start with the description of the considered scenario to establish the bases on which the subsequent OF rules for frame forwarding and seamless migration support are built upon. For completeness, a simple VO clustering policy is also included as an example of how VOs with similar QoS/QoE requirements and/or closely linked with each other can be bundled together and then considered as an aggregate entity in the wide-area.

4.1 Anatomy of MCO Networks

We consider a scenario where users own/manage/have access to physical/virtual objects through geo-distributed VTNs and the set of Cloud/Fog/MEC (in-network) datacenters D. Objects are accessed through their physical (fixed/mobile) and virtual endpoints. A tenant $\delta \in \Delta$ is associated to a VTN, implemented as an MCO network Q_δ – a sort of private overlay specifically designed to provide L2 interconnection among its endpoints, including the user premises physical equipment deployed in the home ($\hbar_u$ – which can be understood as the gateway to a fixed home/enterprise network), and/or

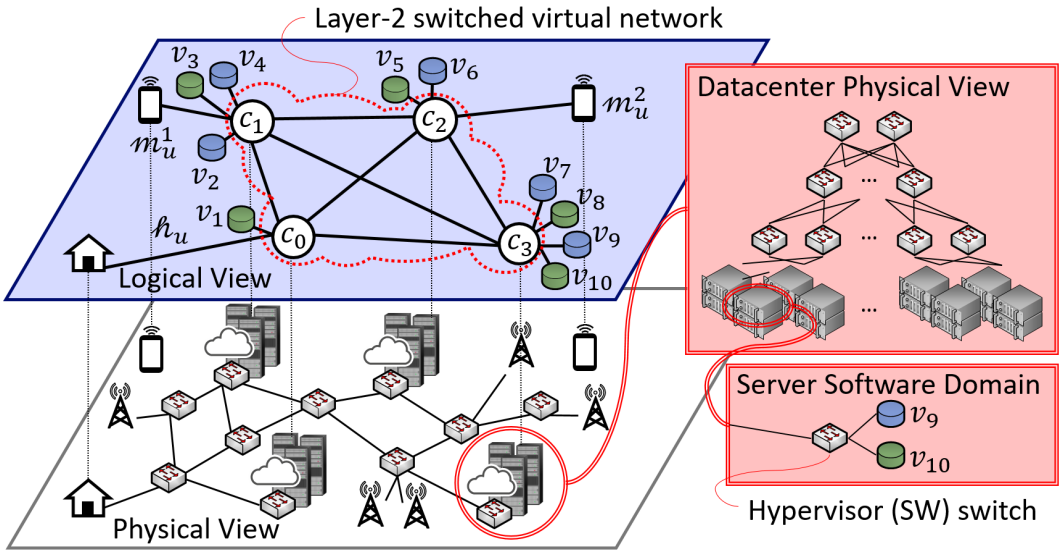


FIGURE 4.1: Physical and logical view of an MCO network owned by a single user.

the set of devices connected through mobile access ($\mathcal{M}_\delta$), as well as the set of VOs (V_δ) hosted within a subset of datacenters ($D_\delta \subseteq D$).

In more detail, endpoints in Q_δ are given by $F(Q_\delta) \triangleq \{\varphi(v), \forall v \in \hbar_u \cup \mathcal{M}_\delta \cup V_\delta\}$, where $\varphi(v)$ is a function providing the current position of the (physical/virtual) object v as a switch-port pair. It is worth noting that the endpoints are terminations both towards the network edge and towards execution containers in D. Generally, a VO v can be hosted by any datacenter $d \in D$, and VOs must be able to migrate across servers of the same datacenter, or across different datacenters, in a seamless and dynamic fashion.

To reflect the peculiarities above, Q_δ is organized into N clusters of objects that are identified with *cluster centers*, $\{c_0, \ldots, c_{N-1}\}$ – each one mapped to a datacenter gateway switch in the telecommunication infrastructure. For instance, Fig. 4.1 illustrates the physical and logical view of an MCO network owned by user u, having 4 cluster centers and 13 endpoints (i.e., 10 VOs, home $\hbar_u$ and mobile access $\mathcal{M}_\delta = \{m_u^1, m_u^2\}$ terminations). The terms *cluster center* and *center* will be used interchangeably hereinafter.

The set $V_{\delta,n} \subseteq V_\delta$ corresponds to the VO cluster bound to the center c_n. If $V_{\delta,n}$ is hosted in the datacenter d, then c_n maps to its gateway switch i_d (i.e., $i_{d_n} \equiv i_d$, $d \in D_\delta : c_n \mapsto i_d$). Without loss of generality, we will assume henceforth that each datacenter has a single gateway switch i_d, $\forall d \in D$, on which (multiple) centers can be mapped, in order to simplify the representation.

Although each VO v belongs to a single cluster in Q_δ, it is important to note that other VO interfaces may be associated with cluster centers of *back-end networks (BNs)*, which are essential for some data handling operations. Again, without loss of generality, we will focus on Q_δ's connectivity in the discussion for easy presentation, but PN-BN interactions will be covered in the experiments.

4.1.1 Overlay Connectivity

We build the L2 connectivity among endpoints in Q_δ on the basis of paths and shortest-path trees, where each SDN device (e.g., physical/virtual OF switch) along a path constitutes a hop. The terms *hop* and *switch* will be used interchangeably in this paper.

By definition, the shortest-path tree $SPT(r, L)$ is the union of the (shortest) paths $\mathcal{P}(l, r)$ from each leaf $l \in L$ to the root r, where $\mathcal{P}(l, r)$ is the optimal sequence of edges and hops from l to r. For a given $\mathcal{P}(l, r)$, two edges e_i, e_o are defined on each of its hops h, and collectively as $\xi \triangleq \{e_i, e_o\}$. The direction of the flow (i.e., towards the leaf/root) is indicated by i/o, respectively.

Considering that edges are mapped to distinct ports on a switch, the terms *ports* and *edges* will be used interchangeably hereinafter. From the perspective of paths, however, an edge e can also be defined by the pair of vertexes at its endpoints, which is given by the function $vertex(e)$.

Suppose that the datacenter $d \in D_\delta$ hosting the VO cluster bound to the center c_n houses the sets $\mathcal{S}_d$ of servers and I_d of OF switches. Inside d, VOs in the set $V_{\delta,n}$ are interconnected according to the shortest-path tree $SPT(i_{d_n}, V_{\delta,n})$, with the gateway switch i_{d_n} being the root and the VOs $v \in V_{\delta,n}$ being the leaves. The servers and intermediate switches involved are given by the subsets $\mathcal{S}^\delta_{d_n} \subseteq \mathcal{S}_d$ and $I^\delta_{d_n} \subseteq I_d$, respectively.

$$SPT(i_{d_n}, V_{\delta,n}) \triangleq \cup_{\forall v \in V_{\delta,n}} \{\mathcal{P}(v, i_{d_n})\} \tag{4.1}$$

Note that over time, multiple cluster centers of the same overlay Q_δ may be mapped on the same gateway switch i_d. Consequently, for each datacenter d, the internal connectivity for Q_δ will then involve all VOs $v \in V^d_\delta$, where $V^d_\delta \triangleq \cup_{\forall c_n \mapsto i_d} V_{\delta,n}$, realized by the shortest-path tree $SPT(i_d, V^d_\delta)$. The servers and intermediate switches involved are then given by $\mathcal{S}^\delta_d \triangleq \cup_{\forall c_n \mapsto i_d} \mathcal{S}^\delta_{d_n}$ and $I^\delta_d \triangleq \cup_{\forall c_n \mapsto i_d} I^\delta_{d_n}$, respectively.

Conversely, the interconnection among the centers $c_0, \ldots, c_{N-1}$ of Q_δ is given by the shortest-path trees $Q^{c_n}_\delta$, $\forall n \in \{0, \ldots, N-1\}$, with the root being i_{d_n} and the leaves being i_{d_m}, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$.

$$Q^{c_n}_\delta \triangleq SPT(i_{d_n}, \{i_{d_m}\}) \tag{4.2}$$

4.1.2 Layer 2 Addressing

The majority of today's IT virtualization platforms/hypervisors allows the association of (customized) locally administered MAC addresses to virtual network interfaces (i.e., of VMs/containers, or those managed by the guest OS). We exploit this capability to configure MAC addresses in a form convenient for flow identification inside/among overlays, as well as for high-speed rule matching in OF hardware and software switches.

As illustrated in Fig. 4.2, the MCO forwarding rules are designed based on the matching of Ethernet 48-bit MAC addresses [118] (and optionally on IEEE 802.1Q VLAN tags [119]). In the former case, the MAC address is partitioned into three fields – from

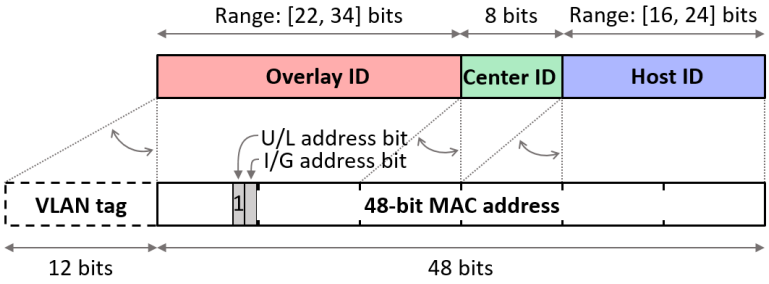


FIGURE 4.2: Overlay network addressing scheme and possible mapping on Ethernet 48-bit MAC addresses and IEEE 802.1Q VLAN tags.

the most significant bit, there is the 22-bit Overlay ID (i.e., 3 Bytes minus the 2 flag bits for universal/local (U/L) and individual/group (I/G) addresses), the 8-bit Center ID, and the 16-bit Host ID. When a VLAN tag is also used, the Overlay and Host IDs can have lengths within the ranges [22,34] and [16,24] bits, respectively. Such configurations are particularly convenient for simple OF matches, considering that OF defined matching of 48-bit MAC addresses with 1 to 6 Bytes masks at a step of 1 Byte, and only supports precise matching of VLAN tags.

4.2 Frame Forwarding Rules

Consider the subset of datacenters $D_\delta \subseteq D$ on whose gateway switches a cluster center in Q_δ has been mapped (i.e., $D_\delta \triangleq \{\forall d \in D : \exists c_n \mapsto i_d\}$). The virtual topologies inside and among $d \in D_\delta$ that define the overlay connectivity are given by a number of OF rules installed on every crossed hardware/software switch, for each VO in V_δ.

In more detail, the OF rules are defined according to two complementary algorithms – one acting inside each datacenter, and the other on the wide-area infrastructure. The rationale behind this division is to isolate as much as possible the number of network reconfigurations during VO migrations or changes in the mapping between overlay and underlay resources. Independently of such implementation, the rules will be described according to their forwarding type (i.e., unicast, broadcast/multicast), for the sake of readability. Additionally, minor rule adaptations for home and mobile access terminations are also presented later on.

4.2.1 OpenFlow Notation

The MCO is designed to use simple and mandatory primitives defined in the OF 1.3.1 protocol [11] and widely supported by commercial switches. All the rules are meant to be placed in Table 0 of OF switches (i.e., first flow table in the OF pipeline).

In this work, the OF rules are expressed in the following form:

$$\mathbf{p} \rightarrow l_x, \ \mathbf{if}\ match_1 \ \&\&\ match_2 \ \&\&\ \cdots \&\&\ match_y \Rightarrow \quad \{action_1, action_2, \ldots, action_z\}$$

TABLE 4.1: MCO key notations.

Parameter	Description
$\mathcal{P}(l,r)$	optimal sequence of edges and hops from the leaf l to the root r
e_i (e_o)	edge at a generic hop $h \in \mathcal{P}(l,r)$ towards l (r)
ξ	set $\{e_i, e_o\}$ of edges at a generic hop $h \in \mathcal{P}(l,r)$
$vertex(e)$	function that returns the pair of vertexes at the endpoints of an edge e
$SPT(r,L)$	shortest-path tree with r being the root and L being the set of leaves, $SPT(r,L) \triangleq \cup_{\forall l \in L}\{\mathcal{P}(l,r)\}$
$\mathcal{S}_d^\delta \subseteq \mathcal{S}_d$	subset of servers in d that are involved in the MCO network Q_δ
$I_d^\delta \subseteq I_d$	subset of OF switches in d that are involved in Q_δ
i_{d_n}	gateway switch i_d of $d \in D_\delta$ on which the cluster center c_n is mapped
$\varphi(\upsilon)$	function that returns the position of the (physical/virtual[1]) object υ as a switch-port pair (i_υ, p_υ)
$V_\delta^d \subseteq V_\delta$	subset of VOs in Q_δ that are hosted in $d \in D_\delta$
$V_{\delta,n} \subseteq V_\delta$	subset of VOs in Q_δ that are bound to c_n
$Q_\delta^{c_n}$	wide-area shortest-path tree with the gateway switch i_{d_n} being the root and the other gateway switches i_{d_m}, $m \in \{0,\dots,N-1\}$, $m \neq n$, being the leaves, $Q_\delta^{c_n} \triangleq SPT(i_{d_n}, \{i_{d_m}\})$
p_{in}	switch port where a frame to be matched enters
out	list of ports where a frame matching the OF rule has to be sent
dl_{src} (dl_{dst})	source (destination) MAC address [2]

The rule priority is given in the first part, with lower x values indicating higher priorities. The second part holds the matching fields and associated actions. Exact and wildcard matches will be represented with the operators '$\equiv$' and '$\equiv_{pfx}$', respectively, with the latter intended for masks hitting the Overlay and/or Center IDs. Other notations used in the OF rules matching fields and action list are defined in Table 4.1, together with a list of key notations as a quick reference guide.

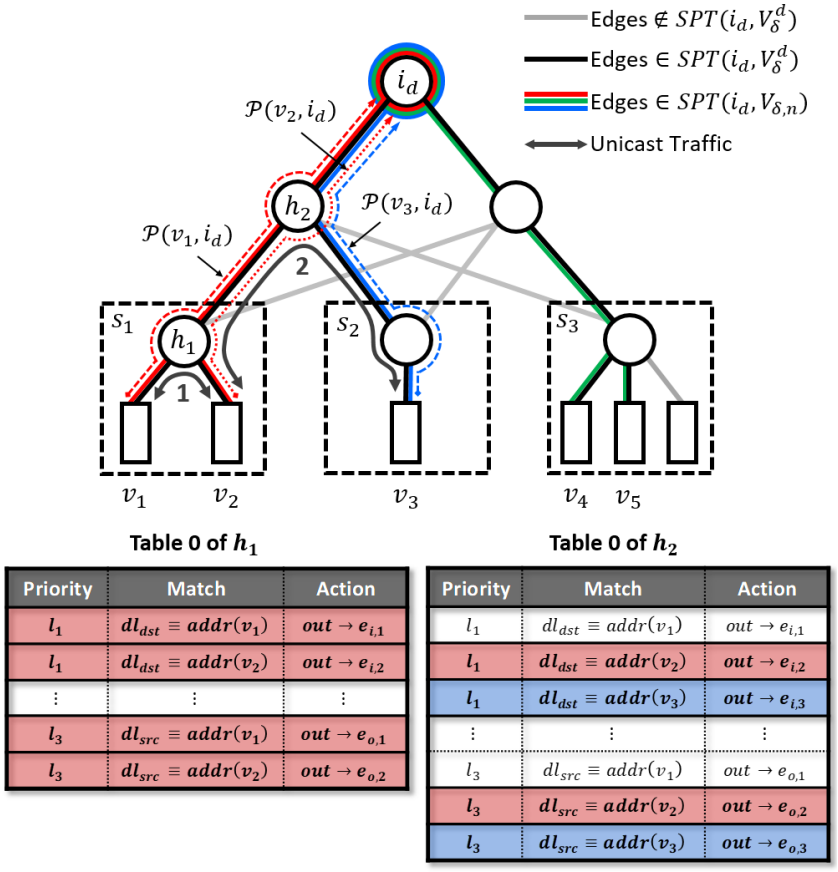
4.2.2 Unicast Forwarding

The following couple of OF rules govern the unicast forwarding inside the datacenter d, for each VO $v \in V_\delta^d$:

$$\mathbf{p} \rightarrow l_1, \textbf{ if } dl_{dst} \equiv addr(v) \Rightarrow out \rightarrow e_i \tag{A}$$

[1] A VO is denoted by v.

[2] Based on Open vSwitch (http://openvswitch.org/) command-line syntax.



Priority	Match	Action
l_1	$dl_{dst} \equiv addr(v_1)$	$out \rightarrow e_{i,1}$
l_1	$dl_{dst} \equiv addr(v_2)$	$out \rightarrow e_{i,2}$
⋮	⋮	⋮
l_3	$dl_{src} \equiv addr(v_1)$	$out \rightarrow e_{o,1}$
l_3	$dl_{src} \equiv addr(v_2)$	$out \rightarrow e_{o,2}$

Priority	Match	Action
l_1	$dl_{dst} \equiv addr(v_1)$	$out \rightarrow e_{i,1}$
l_1	$dl_{dst} \equiv addr(v_2)$	$out \rightarrow e_{i,2}$
l_1	$dl_{dst} \equiv addr(v_3)$	$out \rightarrow e_{i,3}$
⋮	⋮	⋮
l_3	$dl_{src} \equiv addr(v_1)$	$out \rightarrow e_{o,1}$
l_3	$dl_{src} \equiv addr(v_2)$	$out \rightarrow e_{o,2}$
l_3	$dl_{src} \equiv addr(v_3)$	$out \rightarrow e_{o,3}$

FIGURE 4.3: Example of internal datacenter connectivity and unicast frame forwarding among VOs of three co-located clusters.

$$\mathbf{p} \rightarrow l_3, \ \mathbf{if}\ dl_{src} \equiv addr(v) \Rightarrow out \rightarrow e_o \tag{B}$$

Both are installed $\forall h \in \mathcal{P}(v, i_d)$, $h \not\equiv i_d$, while only the rule (A) on i_d. With these, "precise" matching of v's L2 address with the destination and the source L2 addresses, respectively, can be achieved.

In more detail, if a frame generated by v is directed to a another VO $\hat{v} \in V_{\delta,n}$, the frame matches both rules (A) and (B) (i.e., $\mathbf{p} \rightarrow l_1$, $\mathbf{if}\ dl_{dst} \equiv addr(\hat{v}) \Rightarrow out \rightarrow e_i$ and $\mathbf{p} \rightarrow l_1$, $\mathbf{if}\ dl_{dst} \equiv addr(\hat{v}) \Rightarrow out \rightarrow e_i$) in the first interconnection switch i^* where $\mathcal{P}(v, i_d)$ and $\mathcal{P}(\hat{v}, i_d)$ intersect. Such a case is illustrated in Fig. 4.3 (i.e., on the switches h_1 and h_2, for the frames generated by v_2 for v_1 and v_3, respectively). While searching for i^*, rule (B) directs frames towards i_d, independently of their destination. Then, on i^*, rule (A) will be selected for its higher priority, consequently redirecting the frames towards the destination VO (i.e., v_1 and v_3 in the example). Furthermore, it can be noted that these rules allow direct communication between VOs of different clusters (i.e., v_2 and v_3 in Fig. 4.3).

When the destination VO is not hosted in the same datacenter (i.e., destination L2 address is unknown to all the interconnection switches), frames will eventually reach i_d, and the forwarding behavior will be driven by the wide-area algorithm thereon.

In the wide-area, the proposed algorithm relies on wildcard masks matching the

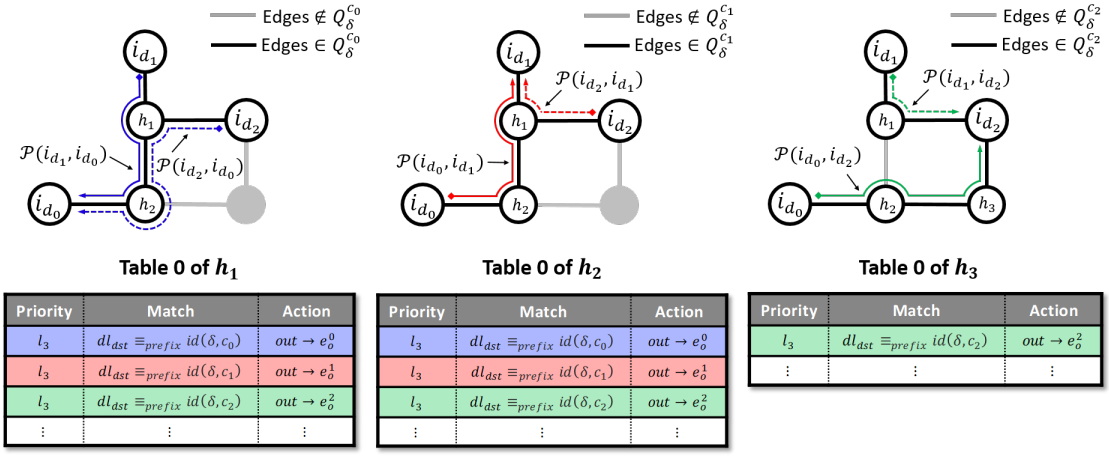


Table 0 of h_1

Priority	Match	Action
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_0)$	$out \rightarrow e_o^0$
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_1)$	$out \rightarrow e_o^1$
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_2)$	$out \rightarrow e_o^2$
⋮	⋮	⋮

Table 0 of h_2

Priority	Match	Action
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_0)$	$out \rightarrow e_o^0$
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_1)$	$out \rightarrow e_o^1$
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_2)$	$out \rightarrow e_o^2$
⋮	⋮	⋮

Table 0 of h_3

Priority	Match	Action
l_3	$dl_{dst} \equiv_{prefix} id(\delta, c_2)$	$out \rightarrow e_o^2$
⋮	⋮	⋮

FIGURE 4.4: Example of wide-area connectivity among three cluster centers located on different datacenters.

Overlay and Center IDs (i.e., $id(\delta, c_n)$), rather than the precise matching of L2 addresses, for improved scalability. This approach is particularly beneficial in the case where a significant number of VOs is clustered in each center.

Recall that the wide-area connectivity is realized through a "fully-meshed" overlay among N cluster centers. The algorithm works by obtaining the tree $Q_\delta^{c_n}$, $\forall c_n \in \{c_0, \ldots, c_{N-1}\}$, according to Eq. (4.2). On each switch $i \in Q_\delta^{c_n}$, $i \not\equiv i_{d_n}$, the following rule is installed:

$$\mathbf{p} \rightarrow l_3, \ \textbf{if } dl_{dst} \equiv_{pfx} id(\delta, c_n) \Rightarrow out \rightarrow e_o \tag{C}$$

This rule aims at directing unicast traffic from any other center c_m (mapped to $i_{d_m} \in Q_\delta^{c_n}$, $m \neq n$), towards i_{d_n} (i.e., the gateway switch of the datacenter hosting the destination VO $v \in V_{\delta,n}$), crossing the $Q_\delta^{c_n}$ tree in upstream (i.e., from the leaf i_{d_m} to the root i_{d_n}); an simple example is illustrated Fig. 4.4.

The wide-area unicast forwarding is designed to use the $Q_\delta^{c_n}$ tree rooted at the destination gateway switch to achieve simpler rules, considering that there is only one path (and port) on the switches along $\mathcal{P}(i_{d_m}, i_{d_n})$, $m \neq n$, that is part of $Q_\delta^{c_n}$. Upon reaching i_{d_n}, the forwarding behavior is again driven by the algorithm inside the datacenter.

As regards path optimality, crossing a tree in upstream always selects the optimal path (e.g., from a VO in one datacenter to the gateway switch of another datacenter). Sub-optimality may result when trees are crossed in downstream, especially in cases with asymmetric edge weights (e.g., different latencies in the two directions) and high levels of path diversity.

4.2.3 Broadcast/Multicast Forwarding

Since broadcast and multicast forwarding rules will be almost identical (i.e., with the I/G bit in the MAC addresses set to '1', and some slight differences with the matching field values), they will be treated as one in this paper. Moreover, translation between

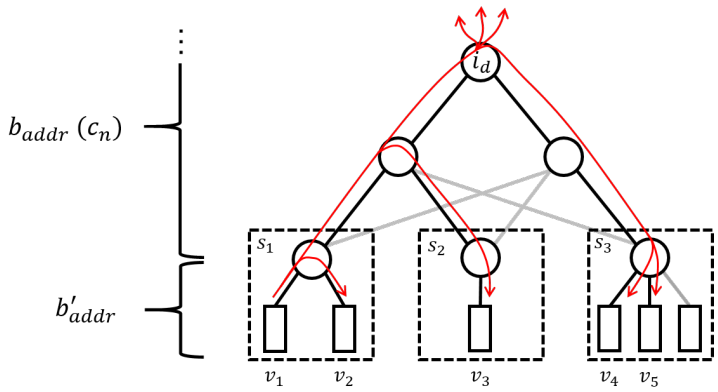


FIGURE 4.5: Example of broadcast/multicast packet forwarding from v_1 to other VOs $v \in V_\delta$ in the same datacenter and towards those in the other datacenters.

universal and overlay broadcast/multicast MAC addresses will be performed at the hypervisor switch.

Inside the datacenter d, the mechanism proposed relies on the same shortest-path tree $SPT(i_d, V_\delta^d)$, as shown in Fig. 4.5. A number of rules corresponding to the number of configured Q_δ cluster centers (N) is installed on every hop $h \in SPT(i_d, V_\delta^d)$.

If the hop $h \not\equiv i_d$ is not directly connected with any VO $v \in V_\delta^d$ (i.e., h does not correspond to the hypervisor switch in a server), the following type of forwarding rule is installed:

$$\mathbf{p} \rightarrow l_2, \ \mathbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \mathcal{E} \tag{D}$$

where $\mathcal{E}$ is the set that contains all edges at h belonging to $SPT(i_d, V_\delta^d)$. On $h \equiv i_d$, the forwarding behavior internally to the datacenter will correspond to the one described in the rule (D). However, as i_d is part of both the datacenter and wide-area networks, a slightly different version of the rule is applied, covering the two portions of the overlay.

$$\mathbf{p} \rightarrow l_2, \ \mathbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \breve{\mathcal{E}}_n \tag{E}$$

where $\breve{\mathcal{E}}_n$ is the set of edges at the gateway switch i_d, $d \in D_\delta : c_n \mapsto i_d$, with elements $e \in \{Q_\delta^{c_n} \cup SPT(i_d, V_\delta^d)\}$, hence considering both internal and wide-area connections.

In case $h \not\equiv i_d$ is directly connected with at least one VO $v \in V_\delta^d$, the rule should take into account the translation between the universal IEEE 802.3 broadcast address b'_{addr} and the ones used in the overlay $b_{addr}(c_n), \forall n \in \{0, \ldots, N-1\}$. Particularly, let h be the hypervisor switch of the server $s \in \mathcal{S}_d$, and let the set $V_\delta^{d,s} \subseteq V_\delta^d$ comprise all VOs $v \in V_\delta$ residing in s. For each cluster center c_n, the following couple of rules is installed on h:

$$\mathbf{p} \rightarrow\ l_2, \ \mathbf{if}\ dl_{dst} \equiv b'_{addr}\ \&\&\ p_{in} \equiv \mathcal{E}_n \Rightarrow \begin{bmatrix} dl_{dst} := b_{addr}(c_n),\ out \Rightarrow \mathcal{E} \cap \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{N-1} \\ out \Rightarrow \mathcal{E}_0 \cup \ldots \cup \mathcal{E}_{N-1} \end{bmatrix} \tag{F}$$

$$\mathbf{p} \rightarrow\ l_2, \ \mathbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow \begin{bmatrix} out \Rightarrow \mathcal{E} \cap \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{N-1} \\ dl_{dst} := b'_{addr},\ out \Rightarrow \mathcal{E}_0 \cup \ldots \cup \mathcal{E}_{N-1} \end{bmatrix} \tag{G}$$

where $\mathcal{E}_n$ is the set of edges at h that directly connects to VOs bound to c_n, i.e.,

$$\mathcal{E}_n \triangleq \{e : e \in \mathcal{E} \wedge vertex(e) \cap V_{\delta,n} \neq \emptyset\} \tag{4.3}$$

As illustrated in Fig. 4.5, rule (F) handles broadcast packets originating from VOs in the datacenter and addressed to $b'_{addr} \equiv \mathbf{FF} : \mathbf{FF} : \mathbf{FF} : \mathbf{FF} : \mathbf{FF} : \mathbf{FF}$. In case of multiple VOs connected to the same hypervisor switch of the broadcast source, it will forward the broadcast frame to them without any change in the frame. Rule (F) is also in charge of translating b'_{addr} into one of the overlay broadcast addresses $b_{addr}(c_n)$, $n \in \{0, \ldots, N-1\}$, and of forwarding the modified frame to the rest of $SPT(i_d, V^d_\delta)$.

Conversely, when a frame arrives at other switches that are directly connected to VOs (s_2 and s_3 in Fig. 4.5), the forwarding behavior will be driven by the rule (G). Such rule will forward a copy of the unmodified frame onto potential links interconnecting further switches, and it will remap the broadcast address back to b'_{addr} in order to deliver the frame to all the VOs directly connected.

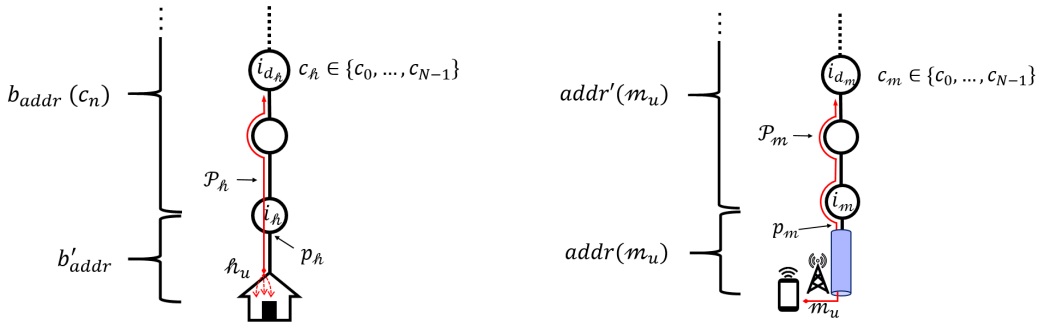
At the wide-area portion of the MCO network, rule (E) can also be applied to make broadcast traffic generated by VOs bound to c_n reach all the other hosts in Q_δ, as the set $\check{\mathcal{E}}_n$ will only include edges $e \in Q^{c_n}_\delta$ on wide-area switches $i \in Q^{c_n}_\delta$, $i \not\equiv i_{d_n}$. In contrast to unicast forwarding, the $Q^{c_n}_\delta$ tree is crossed downstream from the root (which is the origin of broadcast traffic, and has a prefix $id(\delta, c_n)$) to the leaf nodes.

It is worth noting that unwanted frame copies will be avoided since OF switches do not transmit a frame back to the ingress port, even if explicitly listed among the output interfaces. Moreover, this (mandatory) OF feature allows to cross $SPT(i_d, V^d_\delta)$ unidirectionally from the source, which may be one of the VOs in the datacenter, or i_d itself if the broadcast has been originated outside the datacenter, towards all $SPT(i_d, V^d_\delta)$ nodes.

4.2.4 Home and Mobile Access Terminations

A network termination is defined to be a port of an interconnection switch inside the telecommunication network, where traffic transmitted/received by remote devices (in the home/enterprise network or connected through the radio access) is not carried on top of access carrier protocols/point-to-point tunnels. For this purpose, network terminations can be understood as the output ports of nodes performing the authentication and authorization on traffic coming from the wireline/radio access. Without loss of generality, the home and mobile access terminations of the users are supposed to be mapped on the Q_δ cluster centers $c_{\hbar}, c_m \in \{c_0, \ldots, c_{N-1}\}$, respectively.

It is worth noting that the proposed algorithm can support multiple mobile terminals, but only a single home termination. For the sake of simplicity, we will describe the single user case with home termination $\hbar_u$ and only a single mobile device, whose termination is represented by m_u.



(A) home access (B) mobile access

FIGURE 4.6: Support for home and mobile access terminations.

Home connectivity

The algorithm has been specifically designed in order to support physical hosts in the domestic LAN without the need of translating their MAC addresses. This feature has been made possible by adding some further rules for:

- making the traffic originating from/destined to the home LAN reach the center $c_\hbar$; and
- collecting the traffic addressed to devices in the home from all the other cluster centers in Q_δ.

Regarding the interconnectivity between $c_\hbar$ and $\hbar_u$, the shortest path $\mathcal{P}_\hbar$ is calculated, and for each hop $h \in \mathcal{P}_\hbar$, $h \not\equiv i_{d_\hbar}$ (i.e., $c_\hbar \mapsto i_{d_\hbar}$), the following rules are applied:

$$\mathbf{p} \rightarrow l_4, \ \mathbf{if}\ dl_{src} \equiv_{pfx} id(\delta) \Rightarrow out \rightarrow e_i \tag{H}$$

$$\mathbf{p} \rightarrow l_4, \ \mathbf{if}\ dl_{dst} \equiv_{pfx} id(\delta) \Rightarrow out \rightarrow e_o \tag{I}$$

On $i_{d_\hbar}$, only rule (H) is configured, since the node already has rules of type (C) for reaching any VO bound to other centers in Q_δ. On the other hand, on the switch $i_\hbar$ hosting the termination $\hbar_u$ on the port $p_\hbar$, the following rule is applied instead of rule (I):

$$\mathbf{p} \rightarrow l_4, \ \mathbf{if}\ dl_{dst} \equiv_{pfx} id(\delta)\ \&\&\ p_{in} \equiv p_\hbar \Rightarrow out \rightarrow e_o \tag{J}$$

In order to support broadcast traffic to/from the home termination $\hbar_u$, rule (D) is installed for each hop $h \in \mathcal{P}_\hbar$, $h \not\equiv i_\hbar$, and the following couple of rules on $i_\hbar$:

$$\mathbf{p} \rightarrow l_2, \ \mathbf{if}\ dl_{dst} \equiv b_{addr}(c_n) \Rightarrow dl_{dst} := b'_{addr}, \ out \rightarrow e_i \tag{K}$$

$$\mathbf{p} \rightarrow l_2, \ \mathbf{if}\ dl_{dst} \equiv b'_{addr} \Rightarrow dl_{dst} := b_{addr}(c_n), \ out \rightarrow e_o \tag{L}$$

Rules (K) and (L) are designed based on the assumption that $i_\hbar$ is a legacy switch that only hosts home terminations $\hbar_u$, and not VOs $v \in V_\delta^{d_\hbar}$. Fig. 4.6a illustrates the overlay connectivity to/from the home network.

Mobile connectivity

We refer to the 5G 3GPP Release 15 specifications [120]. In such a case, user terminals can be exposed by the Network Exposure Function (NEF) and terminated by specific dedicated ports of User Plane Functions (UPFs), which shall run on the same datacenters hosting VOs. Furthermore, we suppose that $\varphi(m_u) \equiv (i_m, p_m)$, where i_m is the switch connecting to the UPF instance serving user u's terminal and p_m is the port on i_m connected to m_u. It is worth noting that p_m will likely correspond to a virtual port (i.e., the logical termination of a network tunnel).

As in the previous cases, in order to connect m_u to the center c_m, the shortest path $\mathcal{P}_m$ among them is calculated and rules (A) and (B) are installed on each hop $h \in \mathcal{P}_m$, accordingly, except on the switch i_m, on which the following rules will be installed:

$$\mathbf{p} \rightarrow \ l_2, \ \textbf{if } dl_{dst} \equiv addr(m_u) \Rightarrow dl_{dst} := addr'(m_u), \ out \rightarrow p_m \tag{M}$$

$$\mathbf{p} \rightarrow l_2, \ \textbf{if } p_{in} \equiv p_m \Rightarrow dl_{dst} := addr(m_u), \ out \rightarrow e_o \tag{N}$$

where $addr(\cdot)$ and $addr'(\cdot)$ are functions that return the overlay and physical L2 addresses, respectively.

Broadcast forwarding internal to the shortest-path tree $SPT(i_{d_m}, V_\delta^{d_m})$ is supported by the same rules inside the datacenter (i.e., rule (D) is installed on each hop $h \in \mathcal{P}_m$, $h \not\equiv i_{d_m}$, then the rules (F) and (G) on the switch i_m). On i_{d_m}, rule (E) is installed to cover both internal and wide-area interconnections. Fig. 4.6b illustrates the overlay connectivity to/from the mobile device.

4.3 Seamless Migration Support

The MCO is designed to efficiently support the seamless migration of VO(s). By definition, a seamless migration is a VO reallocation from one server to another that is performed with negligible service interruption time. Although such reallocation is achieved with unprecedented ease by exploiting the (advanced) migration mechanisms already supported in state-of-the-art virtualization technologies, network-induced performance drawbacks are still inevitable.

For instance, legacy L2 Ethernet networking can incur further delays, due to reactive rebuilding of switching tables (i.e., a table entry is only updated upon the reception of a frame generated by the VO in its new position after the migration, or after the timeout of the entry expires). In addition, MAC anti-spoofing mechanisms, generally included in the switches, could also prevent updates in the switching table, in effect extending the VO unreachability period up to some seconds (the duration depends on the vendor-specific implementation of the these mechanisms). SDN is well-known to easily overcome such drawbacks, and to enable seamless network reconfigurations during migrations in an effective fashion.

Common approaches for minimizing the service interruption time usually apply a two-step migration procedure, summarized as follows.

Step 1: *Before initiating the migration, network switching/routing rules are temporarily configured in order to duplicate packets destined to the VO(s) on the move towards both "old" and "new" positions. Moreover, in order to guarantee the correct routing of the traffic generated from the new position(s) upon migration completion, also the new forwarding rules are calculated and configured on the involved nodes.*

Step 2: *Upon completion of the migration process, network switching/routing rules are updated in order to remove the connectivity to/from the old position(s), and maintain only the connectivity to/from the new position(s).*

The MCO supports the aforementioned procedure for two types of migration:

- *VO migration*: this only involves one VO, which is being migrated between two servers of the same in-network datacenter; and
- *center migration*: this consists of a bulk migration of all the VOs bound to a cluster center between two in-network datacenters.

These two types have been designed to support reconfiguration of the underlying infrastructure and of overlying services, respectively.

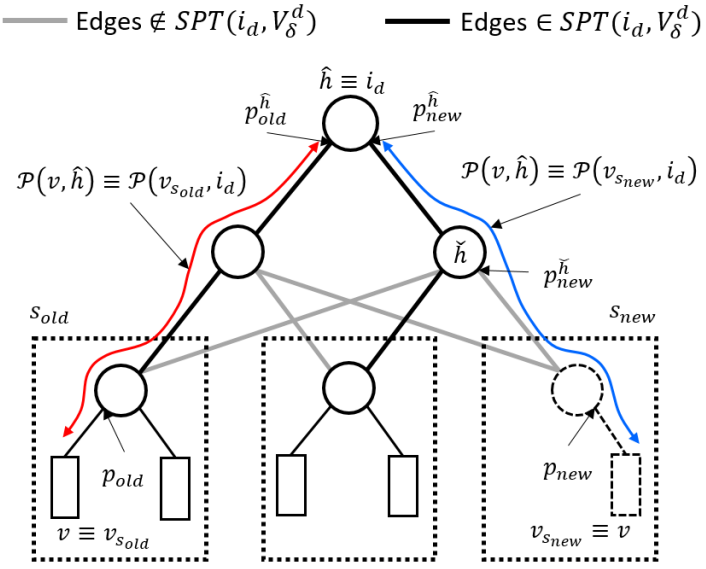
Particularly, the migration of single VOs between two servers of the same datacenter is a primitive operation mandatory for allowing the maintenance of servers, or the dynamic consolidation of datacenter resources (e.g., for reducing the energy consumption by making idle servers entering standby modes). This is especially useful for operations at the infrastructure level, and it should be as transparent as possible to the service level (i.e., to the operation levels and performance provided by services running on VOs).

On the other hand, bulk migration of VOs has been defined for adapting the location of services in an efficient and scalable way. Through the migration of a VO cluster, it may be possible to reduce the end-to-end delay between the services running on such VOs and the end-user devices. For instance, if an end-user accesses services through his/her smartphone, center migrations can be useful to move clusters of VOs with similar QoS requirements closer to his/her mobile termination, also during hand-over from one network access point to another.

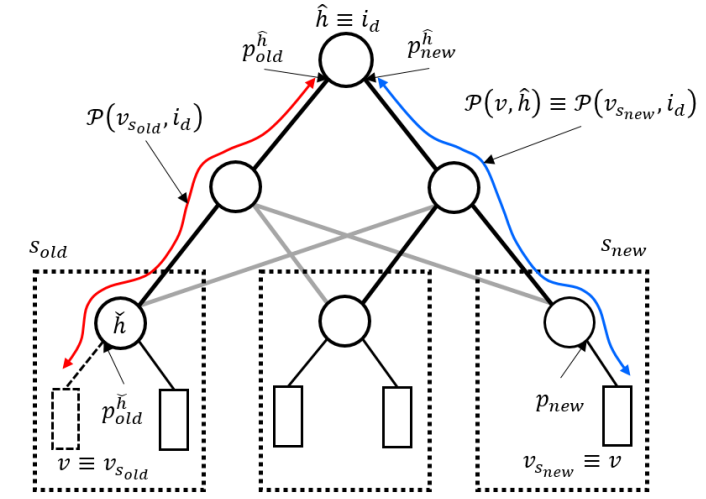
4.3.1 VO Migration

Let v be the VO to be migrated from the server s_{old} to s_{new}, as illustrated in Fig. 4.7. We suppose that $c_n \mapsto i_d$ is the cluster center to which v is bound, and s_{old}, $s_{new} \in \mathcal{S}_d$, $d \in D_\delta$. Moreover, p_{old} and p_{new} are the ports that connect v to the hypervisor switches in s_{old} and s_{new}, respectively.

As previously sketched, this type of migration only involves the overlay configuration inside a single datacenter, and no operations are needed at the backbone level.



(A) before migration



(B) after migration

FIGURE 4.7: Example of VO migration between two servers in the same datacenter.

Therefore, Steps 1 and 2 of the migration procedure only concern the rules acting inside the datacenter, as detailed below.

S1: Let $\hat{h}$ be the switch where the paths $\mathcal{P}(v_{s_{old}}, i_d)$ and $\mathcal{P}(v_{s_{new}}, i_d)$ intersect, where v_s is the instance of v in the server $s \in \{s_{old}, s_{new}\}$. Since there are two paths (i.e., $\mathcal{P}(v, \hat{h}) \in \{\mathcal{P}(v_{s_{old}}, \hat{h}), \mathcal{P}(v_{s_{new}}, \hat{h})\}$) with the same matching rules for v, $\hat{h}$ will also have two edges towards v, $e_i \in \{p_{old}^{\hat{h}}, p_{new}^{\hat{h}}\}$, as shown in Fig. 4.7a – in effect, duplicating the traffic.

On $\hat{h}$, rule (A) is updated to (A′), by adding $p_{new}^{\hat{h}}$ as output interface – this allows unicast frames destined to v to be forwarded to both $p_{old}^{\hat{h}}$ and $p_{new}^{\hat{h}}$. Then, rules of the type (A) and (B) are configured along all the hops $h \in \mathcal{P}(v_{s_{new}}, \hat{h})$.

As regards broadcast traffic, we proceed in a similar fashion, by first identifying the switch $\check{h}$ where $\mathcal{P}(v_{s_{new}}, i_d)$ intersects the shortest-path tree $SPT(i_d, V_\delta^d)$. If $\check{h}$ does not

coincide with the hypervisor switch of s_{new}:

- the rule (D) (or rule (E), if $\check{h} \equiv i_d$) on $\check{h}$ is updated by adding $p^{\check{h}}_{new}$ to the output interfaces, where $p^{\check{h}}_{new}$ is the port on $\check{h}$ through which it is possible to reach v on s_{new};
- rule (D) is added on each hop $h \in \mathcal{P}(s_{new}, \check{h})$; and
- rules (F) and (G) are added on s_{new}'s hypervisor switch.

If $\check{h}$ coincides with the hypervisor switch of s_{new}, only rules (F) and (G) are updated by adding $p^{\check{h}}_{new}$ to the set $\mathcal{E}_n$ of edges at $\check{h}$ that directly connects to VOs that are bound to the center c_n.

S2: This step aims at pruning the unicast and broadcast connectivity towards the old position of v. For this purpose, the rule (A′) on $\hat{h}$ is updated back to (A) by removing the output port $p^{\hat{h}}_{old}$, and rules (A) and (B) are removed from all switches along $\mathcal{P}(v_{s_{old}}, \hat{h})$.

Now, let $\check{h}$ be the switch where $\mathcal{P}(v_{s_{old}}, i_d)$ intersects the shortest-path tree $SPT(i_d, V^d_\delta)$, as illustrated in Fig. 4.7b. If $\check{h}$ does not coincide with the hypervisor switch of s_{old}, the rule (D) (or (E), if $\check{h} \equiv i_d$) on $\check{h}$ is updated, by removing the port $p^{\check{h}}_{old}$ from the output interfaces. Moreover, rule (D) is removed from all switches along $\mathcal{P}(s_{old}, \check{h})$, as well as rules (F) and (G) from the hypervisor switch of s_{old}. On the contrary, if $\check{h}$ coincides with the hypervisor switch of s_{old}, only rules (F) and (G) are updated by removing $p^{\check{h}}_{old} \equiv p_{old}$ from the corresponding set $\mathcal{E}_n$ (defined by Eq. 4.3).

4.3.2 Center Migration

Here we suppose that the VO cluster bound to the center c_n has to be migrated from the datacenter d_{old} to d_{new} (i.e., $c_n \mapsto i_{d_{old}}$ and $c'_n \mapsto i_{d_{new}}$), as depicted in Fig. 4.8. Unlike single VO migrations, Steps 1 and 2 in this type of migration concern the rules acting in both datacenter and wide-area domains, as detailed in the following.

S1: In this step, the datacenter and wide-area portions of the overlay are reconfigured for:

- bidirectionally propagating any frame exchanged among VOs bound to c_n (i.e., $v \in V_{\delta,n}$) between $i_{d_{old}}$ and $i_{d_{new}}$;
- delivering packets from the other centers c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, destined to c_n towards both datacenters d_{old} and d_{new}.

This propagation is needed for guaranteeing that during the bulk migration of VOs $v \in V_{\delta,n}$, those that are still residing in d_{old} (indicated as $V^{d_{old}}_{\delta,n} \subseteq V_{\delta,n}$) can communicate with the ones already transferred to d_{new} (indicated as $V^{d_{new}}_{\delta,n} \subseteq V_{\delta,n}$), and vice versa.

For this purpose, some rules of type (A) related to the VOs bound to c_n, in both d_{old} and d_{new}, are modified in order to make unicast frames destined to such VOs reach the datacenter gateway. Particularly, rule (A″) is derived from (A) by setting the output

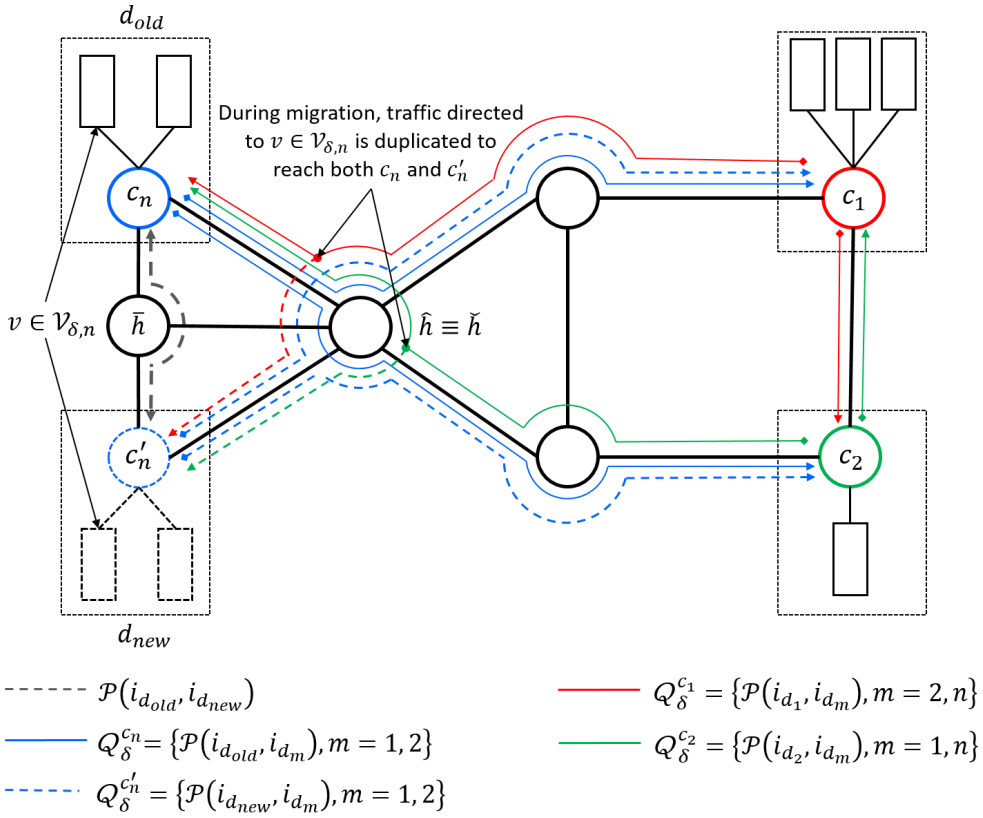


FIGURE 4.8: Example of center migration between two datacenters.

interfaces to ξ – this allows traffic to flow in either e_i or e_o directions of the path. Thus, rules of type (A″) are configured $\forall v \in V_{\delta,n}^{d^*} \cap V_{\delta,n}$, $d^* \in \{d_{old}, d_{new}\}$, and $\forall h \in \mathcal{P}(v, i_{d^*})$. On i_{d^*}, however, rules of type (A) are retained for forwarding of wide-area traffic destined to v. Rules of types (B), (D), (F) and (G) are also configured accordingly in d_{new}.

In order to guarantee the correct traffic exchange between VOs bound to c_n in d_{old} and d_{new}, the path $\mathcal{P}(c_n, c'_n) \equiv \mathcal{P}(i_{d_{old}}, i_{d_{new}})$ in the wide-area portion is computed. Considering this path, the following rule is installed on $i_{d_{old}}$ and $i_{d_{new}}$:

$$\mathbf{p} \rightarrow l_1, \ \mathbf{if}\ dl_{dst} \equiv_{pfx} id(\delta, c_n)\ \&\&\ p_{in} \equiv p_{dc} \Rightarrow out \rightarrow \xi \tag{O}$$

and for each hop $h \in \mathcal{P}(c_n, c'_n)$ (e.g., $\bar{h}$ as shown in the example in Fig. 4.8), the rule:

$$\mathbf{p} \rightarrow l_2, \ \mathbf{if}\ dl_{dst} \equiv_{pfx} id(\delta, c_n) \Rightarrow out \rightarrow \xi \tag{P}$$

In the former, specifying the input port as a datacenter connection (p_{dc}) avoids loops around this path.

On the other hand, the interconnection between the cluster center c'_n in $i_{d_{new}}$ towards/from any other center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, is realized by identifying the intersection switch $\hat{h}$ between $\mathcal{P}(i_{d_{old}}, i_{d_m})$ and $\mathcal{P}(i_{d_{new}}, i_{d_m})$. On each hop $h \in \mathcal{P}(i_{d_{new}}, \hat{h})$, as well as on $i_{d_{new}}$, the rule (C) configured to match the prefix $id(\delta, c_m)$ is installed, allowing unicast frames from c'_n to reach c_m. In such a case, no updates are required on $\hat{h}$,

since from $\hat{h}$ to any other center c_m the rules for the shortest-path trees $Q_\delta^{c_n}$ and $Q_\delta^{c'_n}$ are the same (see Fig. 4.8).

Conversely, to reach c'_n from c_m, the rule (C) configured to match the prefix $id(\delta, c_n)$ is installed on each hop $h \in \mathcal{P}(i_{d_{new}}, \hat{h})$, and is updated on $\hat{h}$ to (C′), by adding $p_{new}^{\hat{h}}$ as output interface, duplicating traffic towards the sub-paths $\mathcal{P}(i_{d_{old}}, \hat{h})$ and $\mathcal{P}(i_{d_{new}}, \hat{h})$, as illustrated in Fig. 4.8.

As regards broadcast traffic, in order to assure the correct delivery of frames generated by $v \in V_{\delta,n}$ in d_{old} towards d_{new}, and vice versa, the following rule is added to every hop $h \in \mathcal{P}(i_{d_{old}}, i_{d_{new}})$:

$$\mathbf{p} \rightarrow l_1, \textbf{ if } dl_{dst} \equiv b_{addr}(c_n) \Rightarrow out \rightarrow \xi \tag{Q}$$

Moreover, thanks to the rule (E) already present in $i_{d_{old}}$, broadcast frames generated by $v \in V_{\delta,n}$ in d_{new} are propagated to the other cluster centers through the shortest-path tree $Q_\delta^{c_n}$, whose root is in $i_{d_{old}}$.

Finally, rules of type (E) on wide-area switches are also installed/updated for delivering broadcast traffic generated in any other cluster center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$. In particular, a new instance of rule (E) configured to match $b_{addr}(c_m)$ as destination address is installed on every hop $h \in \mathcal{P}(i_{d_{new}}, \check{h})$, where $\check{h}$ is the switch where $\mathcal{P}(i_{d_{new}}, i_{d_m})$ intersects the shortest-path tree $Q_\delta^{c_m}$. On $\check{h}$, the rule (E) matching $b_{addr}(c_m)$ is updated to (E′), by adding an output interface towards the sub-path $\mathcal{P}(\check{h}, i_{d_{new}})$.

S2: Upon the completion of all the migration operations $\forall v \in V_{\delta,n}$ from the datacenter d_{old} to d_{new}, this step is activated to prune all forwarding operations to/from d_{old} (at least for what concerns the traffic related to the center c_n).

To this end, rules of types (A) and (B) are removed $\forall v \in V_{\delta,n}$, and $\forall h \in \mathcal{P}(v, i_{d_{old}})$. Also, the shortest-path tree $SPT(i_{d_{old}}, V_\delta^{d_{old}})$ used for delivering broadcast messages inside d_{old} is modified accordingly, removing all the leaf nodes (i.e., corresponding to v, $\forall v \in V_{\delta,n}$) and related sub-paths to them. These operations are consequently reflected in the possible update and removal of the hops $h \in SPT(i_{d_{old}}, V_\delta^{d_{old}})$ of rules of type (D), and/or (F) and (G).

As regards the wide-area connectivity, the rule (C) matching $id(\delta, c_n)$ is removed $\forall h \in \mathcal{P}(i_{d_{old}}, \hat{h})$, while the rule (C′) on $\hat{h}$ is updated back to (C), by removing the port $p_{old}^{\hat{h}}$ from the output interfaces.

The shortest-path trees $Q_\delta^{c_m}$, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, are updated in a similar fashion in the case where no other VO cluster is residing in d_{old}. First, rules of type (C)/(C′) matching $id(\delta, c_m)$ are removed/updated, accordingly. Rules of type (E) configured to match the broadcast traffic generated in the center c_m, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, are also removed from $i_{d_{old}}$ and $\forall h \in \mathcal{P}(i_{d_{old}}, \check{h})$, with $\check{h}$ now being the switch where $\mathcal{P}(i_{d_{old}}, i_{d_m})$ intersects $Q_\delta^{c_m}$. Then on $\check{h}$, rule (E) is updated by removing the port $p_{old}^{\check{h}}$ from the output interfaces.

Finally, the shortest-path tree $Q_\delta^{c_n}$ is rebuilt, changing its root from $i_{d_{old}}$ to $i_{d_{new}}$. Let $Q_\delta^{c_n}$ and $Q_\delta^{c'_n}$ be the shortest-path trees calculated according to Eq. (4.2), with $i_{d_{old}}$ and $i_{d_{new}}$ as root nodes, respectively. The rule (E) matching $b_{addr}(c_n)$ is consequently:

- removed from hop $\tilde{h}$, $\forall \tilde{h} : \tilde{h} \in Q_\delta^{c_n} \wedge \tilde{h} \notin Q_\delta^{c'_n}$; and
- added on hop $\tilde{h}$, $\forall \tilde{h} : \tilde{h} \notin Q_\delta^{c_n} \wedge \tilde{h} \in Q_\delta^{c'_n}$.

Thanks to the optimal substructure property of the shortest paths composing $Q_\delta^{c_n}$ and $Q_\delta^{c'_n}$, the resulting spanning tree obtained with the above rule removals and additions will still be composed of the shortest paths among $i_{d_{new}}$ and any other cluster center c_m, $\forall m \in \{0, \ldots, N-1\}, m \neq n$.

Note that this procedure cannot produce switching loops, since the forwarding rules deliver packets only towards the leaf nodes of $Q_\delta^{c_n}$ and $Q_\delta^{c'_n}$; hence, the possible and transitory presence of two root nodes could only produce duplicated broadcast packets. Moreover, since the same VO can reside in only one datacenter at a time, such duplicates can be avoided by removing rules of type (O), (P) and (Q), just before passing from $Q_\delta^{c_n}$ to $Q_\delta^{c'_n}$.

4.4 Proximity- and Affinity-aware VO Clustering

VOs in V_δ can have varying user proximity and inter-VO affinity requirements that must be taken into account as users move around. With this in mind, the concept of cluster centers provides an abstraction for clusters of VOs with similar QoS/QoE requirements and/or strong affinities among each other, which is particularly useful for managing bulk migrations during user/user base mobility.

For completeness, a simple VO clustering policy is presented as an example that jointly considers the aforementioned requirements in this section. Moreover, the approach also adopts a subscription-based parameter (that could vary among tenants) to allow service differentiation. Suppose that for each VTN Q_δ, the user proximity Π_δ, inter-VO affinity π_δ requirements and subscription-based parameter P_δ are given.

4.4.1 User Proximity

Different QoS parameters (e.g., path lengths, latencies, available bandwidths, etc.) can be used to measure user proximity, and the paths from user (user base) u's access device(s) to the subset of datacenters $D_\delta \subseteq D$ currently hosting the VTN's VOs must guarantee the ones specified in the SLA. We suppose that latencies and bandwidths are represented in the edge weights and consider the path lengths in the VO clustering.

We define *proximity levels* as a range of indexes $\{\mathbf{1}, \mathbf{P}\}$ by looking at the proximity requirements $\Pi_\delta = \{\Pi_\delta(a), a = 1, \ldots, |V_\delta|\}$ and the subscription-based parameter P_δ that specifies the maximum number of proximity levels allowed by the user (i.e., $\mathbf{P} = P_\delta$). Each index is in turn mapped to a range of path lengths, as detailed in the following.

Let $L_{min} = min(\Pi_\delta)$ and $L_{max} = max(\Pi_\delta)$ correspond to the VOs with the tightest and loosest proximity requirements, respectively. The p-th range of path lengths, $p = 1, \ldots, P_\delta$, is given by

$$[\, r_{min}(p),\, r_{max}(p)\,] = [\, L_{min} + (p-1) \cdot R,\, L_{min} + p \cdot R\,] \tag{4.4}$$

where $R = (L_{max} - L_{min})/P_\delta$. VOs that fall on the $p-$th range will have a proximity level **p**, with higher levels indicating looser proximity requirements.

As regards service differentiation, tenants with higher subscription rates may opt for smaller values of P_δ, resulting in lesser proximity levels with longer range intervals.

4.4.2 Inter-VO Affinity

ETSI NFV has defined "affinity/antiaffinity rules" to specify the proximity among a certain (sub)set of virtual resources (e.g., sharing the same physical NFV infrastructure node) [121]. The inter-VO affinity is closely linked to this concept, where the VOs correspond to the resources that can have proximity requirements towards users and among one another. In order to distinguish the two metrics, we refer to the latter as 'affinity' hereinafter.

Although a range of indexes corresponding to *affinity levels* can be generally defined as we have seen with user proximity, inter-VO affinity requirements may not explicitly appear in SLAs. Interactions among VOs are highly user-dependent, and advanced learning techniques may be necessary to extract affinities from inter-VO traffic in each VO network [122]. For the sake of simplicity, but without losing generality, we only consider two levels in this example – i.e., the distance $\pi_{(a,b)}$ between any pair of u's VOs $(v_a, v_b) \in V_\delta$, $a \neq b$, is either 0 or ∞, leaving multi-level affinity to future work.

User/user base u's affinity requirements are given by $\pi_\delta = \{\pi_{(a,b)}, a = 1, \ldots, |V_\delta|, b = 1, \ldots, |V_\delta|, a \neq b\}$. The affinity levels **1** and **2** correspond to $\pi_{(a,b)} = 0$, specifying that the VOs v_a and v_b must be hosted by the same datacenter, and $\pi_{(a,b)} = \infty$, where the VO pair is independent of each other, respectively. This means that pairs of VOs with 0 distance must be placed in the same datacenter, while the rest can be placed in any datacenter $d \in D$, provided that their user proximity requirements are met.

4.4.3 VO Clustering

Basically, the considered proximity- and affinity-aware VO clustering policy is a two-step process, which can be summarized as follows.

Step 1: *VO pairs $(v_a, v_b) \in V_\delta$, $a \neq b$, with affinity levels* **1** *($\pi_{(a,b)} = 0$) among them are grouped together, obtaining the initial set of clusters $\hat{C}$. The minimum requirements $min(\Pi^{\hat{c}}_\delta)$ of each cluster $\hat{c} \in \hat{C}$ are then determined for the second step.*

Step 2: *The range intervals of the P_δ proximity levels are obtained by adapting Eq. (4.4) to consider clusters instead of individual VOs, letting $L_{min} = min(\{min(\Pi^{\hat{c}}_\delta), \hat{c} \in \hat{C}\})$ and*

$L_{max} = max(\{min(\Pi_{\delta}^{\hat{c}}), \hat{c} \in \hat{C}\})$. *In effect, subset of clusters* $\{\hat{c}\} \subseteq \hat{C}$ *with* $\{min(\Pi_{\delta}^{\hat{c}})\}$ *falling on the same range will be merged. This gives the final set of clusters* C*, with their corresponding minimum proximity requirements* $\{min(\Pi_{\delta}^{c}), c \in C\}$.

VOs in each cluster $c \in C$ can now be considered as an aggregate entity, in effect simplifying network management during inter-datacenter bulk live migrations for user mobility support.

4.5 Scalability and Performance Metrics

In this section, the set of metrics considered to evaluate MCO's scalability and performance is introduced, along with the different parameters used to understand their behavior.

Taking a look at different scalability aspects, the key metrics that we considered are two: the number of OF rules needed to set up an MCO overlay, and the number of updates needed in case of VO/cluster migration. This choice was mainly driven by the fact that OF switches have finite-sized rule tables (usually in the order of some thousands [48]), and that the number updates/instantiations of OF rules directly affects the time needed by the SDN controller to apply the network reconfiguration to the involved switches (since each rule addition, deletion, or update has to be signalled through a separate OF message [11]).

As regards the performance, we take a look at path optimality, as well as some metrics related to L2 overlay connectivity instantiation (i.e., rule calculation times) and center migration (i.e., rule update calculation times) processes that give indications on MCO's computational overhead. The latter are measurable from the timestamps generated by the MCO code – hence, it is necessary to run the MCO algorithm over a given (emulated) telecommunication infrastructure topology in order to collect measurements.

4.5.1 Number of Forwarding Rules

The number of forwarding rules basically gives indications on the scalability of the overlay implementation. This metric depends on the number of VOs and switches involved in the overlay – and consequently, on the geographical distribution and topology of the datacenters, as well as the VO placement among them. For this purpose, we define three datacenter *depths* to represent the upcoming Fog/Cloud interplay using the number of intermediate hops η between the VO and the datacenter gateway switch, as illustrated in Fig. 4.9. The traditional three-layer datacenter topology [123] of the Cloud ($\dddot{d}$) is supposed to be represented by the datacenter depth $\eta = 3$, while the heterogeneity among Fog nodes ($\dot{d}$ and $\ddot{d}$) by the depths $\eta \in \{1, 2\}$. This is particularly useful in counting unicast forwarding rules.

The rules for unicast (R_U) and broadcast (R_B) forwarding are counted separately in order to show their distinct behaviors – note that the sum of these two components

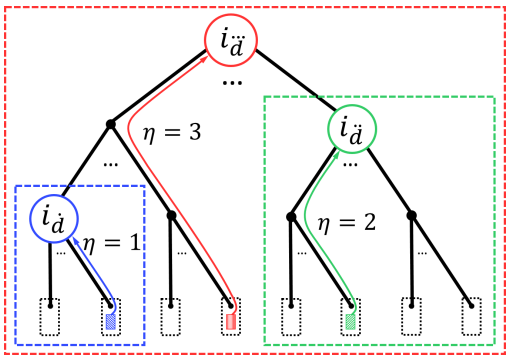


FIGURE 4.9: Different datacenter depths $\eta \in \{1, 2, 3\}$.

indicates the total number of rules ($R = R_U + R_B$) needed to be installed for overlay connectivity.

Unicast

R_U is simply the sum of the rules of types (A)–(C), and is given by:

$$R_U = \sum_n \overbrace{\left[|V_{\delta,n}|(\eta_{d_n}+1)}^{\text{(A)}} + \overbrace{|V_{\delta,n}|(\eta_{d_n})}^{\text{(B)}} + \overbrace{(|Q_\delta^{c_n}|-1)\right]}^{\text{(C)}} = \sum_n \left[|V_{\delta,n}|(2\eta_{d_n}+1) + |Q_\delta^{c_n}| - 1\right] \tag{4.5}$$

where η_{d_n} is the number of intermediate hops from a VO $v \in V_{\delta,n}$ to the gateway switch i_{d_n}, $\forall n \in \{0, \ldots, N-1\}$.

Broadcast

R_B, on the other hand, is obtained by summing up the rules of types (D)–(G), as follows, $\forall d \in D$:

$$R_B = \sum_d \overbrace{\left[N(|I_d^\delta|-1)}^{\text{(D)}} + \overbrace{2N|\mathcal{S}_d^\delta|\right]}^{\text{(F) and (G)}} + \sum_n \overbrace{|Q_\delta^{c_n}|}^{\text{(E)}} = \sum_d \left[N(|I_d^\delta| + 2|\mathcal{S}_d^\delta| - 1)\right] + \sum_n |Q_\delta^{c_n}| \tag{4.6}$$

Considering that multiple cluster centers can be mapped to the same gateway switch, the *cluster/datacenter ratio* may vary over time – it would also be interesting to look at how the number of rules vary with this parameter, as we shall see later on.

4.5.2 Number of Rule Updates

On the other hand, the scalability of handling bulk migrations between two datacenters is evaluated in terms of the number of rule updates in the wide-area portion of the overlay. This covers the rules installed/modified/removed on $i_{d_{old}}$, $i_{d_{new}}$, $\forall h \in \mathcal{P}(i_{d_{old}}, i_{d_{new}})$, and $\forall h \in \{\mathcal{P}(i_{d_{old}}, i_{d_m}), \mathcal{P}(i_{d_{new}}, i_{d_m})\}$, $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$, in performing a center migration.

For the sake of readability, we decompose the number of wide-area rule updates ($R_{M_{wa}}$) into two components (i.e., $R_{M_{wa}} = R^1_{M_{wa}} + R^2_{M_{wa}}$) that correspond to the two-step seamless migration procedure. Particularly, the number of rule updates in Step 1 is given by:

$$\begin{aligned} R^1_{M_{wa}} &= \overbrace{2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1)}^{\text{(O) – (Q)}} + \sum_m \overbrace{\left[2(|\mathcal{P}(i_{d_{new}}, \hat{h}_m)| + 1)\right.}^{\text{(C) and (C')}} + \overbrace{\left.(|\mathcal{P}(i_{d_{new}}, \check{h}_m)| + 1)\right]}^{\text{(E) and (E')}} \\ &= 2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1) + \sum_m \left[2|\mathcal{P}(i_{d_{new}}, \hat{h}_m)| + |\mathcal{P}(i_{d_{new}}, \check{h}_m)| + 3\right] \end{aligned} \tag{4.7}$$

where $\hat{h}_m$ (respectively, $\check{h}_m$), $\forall m \in \{0, \ldots, N-1\}$, $m \neq n$ are the intersection switches between $\mathcal{P}(i_{d_{new}}, i_{d_m})$ and $\mathcal{P}(i_{d_{old}}, i_{d_m})$ (respectively, $Q^{c_m}_{\delta}$). A similar expression is obtained for the number of rule updates in Step 2:

$$\begin{aligned} R^2_{M_{wa}} &= \overbrace{2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1)}^{\text{(O) – (Q)}} + \sum_m \overbrace{\left[2(|\mathcal{P}(i_{d_{old}}, \hat{h}_m)| + 1)\right.}^{\text{(C) and (C')}} + \overbrace{\left.(|\mathcal{P}(i_{d_{old}}, \check{h}_m)| + 1)\right]}^{\text{(E) and (E')}} \\ &= 2(|\mathcal{P}(i_{d_{old}}, i_{d_{new}})| + 1) + \sum_m \left[2|\mathcal{P}(i_{d_{old}}, \hat{h}_m)| + |\mathcal{P}(i_{d_{old}}, \check{h}_m)| + 3\right] \end{aligned} \tag{4.8}$$

with $\check{h}_m$ now being the intersection switch between $\mathcal{P}(i_{d_{old}}, i_{d_m})$ and $Q^{c_m}_{\delta}$.

Furthermore, we define the *clustering index* as the number of VOs clustered in the center to be migrated, which is a parameter useful in studying the behavior of this metric.

4.5.3 Path Lengths

Recalling that path sub-optimality may result when trees are crossed in downstream, we focus on the case when two communicating VOs are inside the same datacenter to evaluate MCO's path optimality. Moreover, equal (eqW) and random asymmetrical (rndW) edge weights are considered for different datacenter topologies.

4.5.4 Rule Calculation Times

As users subscribe to new services, SCs are instantiated and added to their corresponding VTNs. In this process, the time it takes to calculate the rules needed to be installed for the new SC gives indication on the computational complexity of the MCO algorithm. The behavior of this metric is studied by considering different service chaining scenarios.

4.5.5 Rule Update Calculation Times

As users move around, center migration(s) may be initiated to meet the desired QoS/QoE. In this process, the time it takes to calculate the rule updates needed to perform a

center migration gives indication on the time overhead incurred by the MCO algorithm in supporting seamless bulk migrations between datacenters.

4.6 Numerical Results

MCO's scalability (in terms of overlay implementation and bulk migration support) and performance (in terms of path optimality) are evaluated through a series of numerical simulations. In this section, details on the simulation framework and considered baselines are presented together with the obtained results.

4.6.1 Simulation Framework

A simulation framework for a city-wide telecommunication infrastructure with 30 in-network datacenters interconnected by 100 transit nodes is implemented in Matlab. Datacenter depths $\eta \in \{1, 2, 3\}$ are generated according to the probability mass function $f_\eta = \{0.6, 0.35, 0.05\}$, respectively. These parameters have been chosen by taking inspiration from state-of-the-art datacenter network architectures [123–125], as well as from the analyses done in [126]. Letting $\mathcal{T}$ be the set of transit nodes (i.e., access and interconnection switches), the logical interconnections $E_\mathcal{T}$ among the nodes $t \in \mathcal{T}$ are then generated randomly to form a graph-based topology $G(\mathcal{T}, E_\mathcal{T})$. Finally, based on the resulting topology, a transit node $t \in \mathcal{T}$ is randomly chosen for each $d \in D$, with constraint on the minimum number of hops (H_{min}) between any pair of datacenters $(d_a, d_b) \in D$, whose value highly depends on the graph size and topology. In this work, $H_{min} \leq 3$ is required in order to obtain a solution, and the value $H_{min} = 3$ is used to maximize the topological distribution of datacenters.

For simplicity, but without loss of generality, the best-case scenario where all the VOs in the same datacenter are hosted in the same server is considered in the simulations in order to remove the dependence on the VOs' placement inside the datacenter; the latter is beyond the scope of this work and treated in another on-going work. 20 runs with varying seeds are executed for each parameter configuration to show the 95% confidence intervals in the results, with each run corresponding to a unique infrastructure topology.

4.6.2 Overlay Implementation

For unicast forwarding, the proposed approach (**MCO**) is evaluated in comparison with three baselines – $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$. $\mathbf{B_1}$ and $\mathbf{B_3}$ basically correspond to the *fully-meshed* and *OpenStack* cases considered in [127], respectively. In more detail, we suppose to install exact matching rules for each source/destination pair of VOs $(v_a, v_b) \in V_\delta$, $a \neq b$, $\forall h \in \mathcal{P}(v_a, v_b)$ on all switches involved in the former; in the latter, rules are installed only on the hypervisor switches of the servers involved in Q_δ, assuming that legacy routers and/or switches are in place instead of SDN-enabled switches. It is important to note that conventional routing/tunneling protocols used on top of the OpenStack

TABLE 4.2: Overlay implementation baselines.

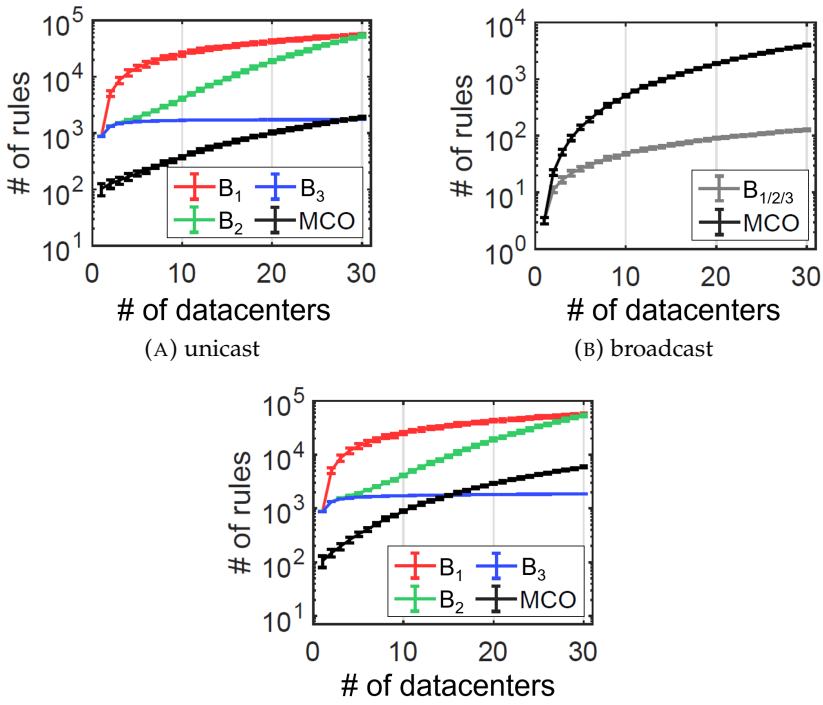
Baseline	Description
$\mathbf{B_1}$	*fully-meshed:* installs exact matching rules for each source/destination pair of VOs on all switches involved
$\mathbf{B_2}$	*two-level flow aggregation:* compression of rules to combine flows towards the same server/datacenter
$\mathbf{B_3}$	*OpenStack:* installs exact matching rules for each source/destination pair of VOs only on hypervisor switches of servers involved
$\mathbf{B_{1/2/3}}$	single overlay broadcast address, wide-area connectivity determined by the minimum spanning tree

platform [128] in $\mathbf{B_3}$ also incur additional costs. On the other hand, $\mathbf{B_2}$ corresponds to a case of *two-level flow aggregation* (i.e., on the server and datacenter levels) and falls in between the other two cases; this baseline is supposed to cover recent developments on flow table minimization, like the one presented in [48].

As regards broadcast forwarding, the baseline $\mathbf{B_{1/2/3}}$ corresponds to the case with a single overlay broadcast address; hence, only one rule is installed on the switches involved in Q_δ – except on the hypervisor switches, where two rules are installed for the mapping between the overlay and universal broadcast addresses. The wide-area connectivity is determined by the minimum spanning tree $Q_\delta^{i_{d^*}} : |Q_\delta^{i_{d^*}}| = min_{d \in D_\delta} |Q_\delta^{i_d}|$. Adding the values obtained in $\mathbf{B_{1/2/3}}$ to those of $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ results in the total number of forwarding rules in the considered baselines. A summary of the aforementioned baselines is reported in Table 4.2, for quick reference.

To evaluate path optimality, path lengths between two communicating VOs are calculated using Dijkstra-based shortest path (SP) and the proposed MCO algorithms. The comparison is built on a conservative assumption that conventional frame forwarding mechanisms use the optimal/shortest path.

Suppose that there is one-to-one correspondence between clusters and datacenters, and VOs are uniformly distributed among N datacenters. With 30 VOs in V_δ, all communicating with each other, Fig. 4.10a shows that **MCO** has up to over $1 \sim 2$ order of magnitude less rules than the three baselines, depending on the number of datacenters involved. On the other hand, **MCO** has more broadcast rules than $\mathbf{B_{1/2/3}}$, as shown in Fig. 4.10b, stemming from the N overlay broadcast addresses and wide-area trees involved in broadcast forwarding. Despite this, the total number of forwarding rules of **MCO** remains better than $\mathbf{B_1}$ and $\mathbf{B_2}$ by a considerable difference, as well as for small values of N in $\mathbf{B_3}$ (i.e., at $N \approx 15$, **MCO** starts to have more rules than $\mathbf{B_3}$), as shown in Fig. 4.10c. Moreover, the N wide-area trees rooted at each datacenter gateway involved are expected to provide better paths based on where broadcast traffic is generated, compared to those given only by $Q_\delta^{i_{d^*}}$.



(A) unicast

(B) broadcast

(C) total

FIGURE 4.10: Number of forwarding rules in the multicenter overlay (**MCO**), fully-meshed (**B_1**), two-level flow aggregation (**B_2**) and OpenStack (**B_3**) cases, given 30 VOs.

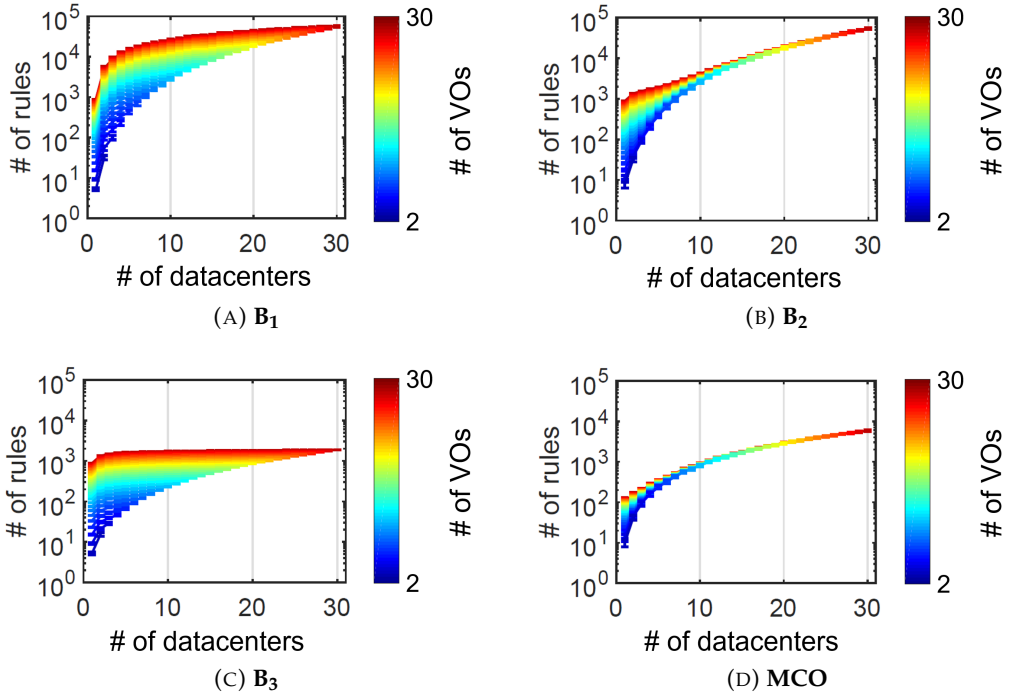
Impact of the number of VOs

The impact of the number of VOs in V_δ on the number of forwarding rules in **MCO** and the three baselines is illustrated in Fig. 4.11. A stronger dependence on the number of VOs can be observed for **B_1** and **B_3** due to the exact matching rules installed for each source/destination pair of VOs in unicast forwarding. By counting only the unicast rules on the hypervisor switches plus the baseline broadcast rules, the latter also displays a stabilizing behavior with increasing number of datacenters. Conversely, the advantage of flow aggregation in **B_2** becomes more evident as the number of datacenters involved increases, while **MCO** exhibits the least dependence on the number of VOs with its particular forwarding algorithm inside and among the datacenters.

Impact of the cluster/datacenter ratio

Recall that multiple cluster centers can be mapped to the same datacenter gateway over time – in such a case, the initial assumption of one-to-one correspondence between clusters and datacenters is no longer true.

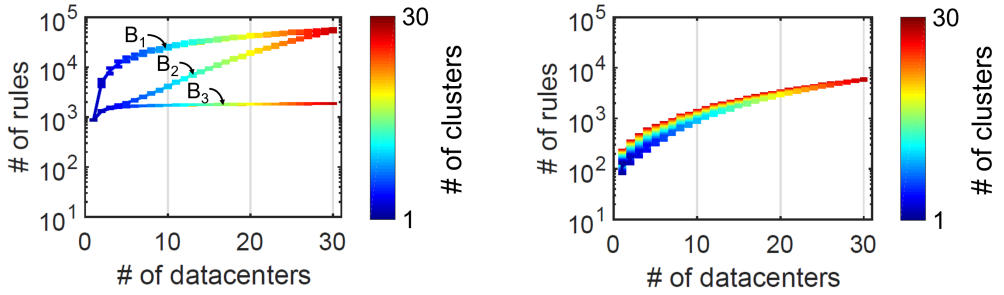
With this in mind, Fig. 4.12 illustrates the impact of the cluster/datacenter ratio on the number of forwarding rules in **MCO** and the three baselines. As expected, **B_1**, **B_2** and **B_3** are only dependent on the number of datacenters involved, while for **MCO**, the number of forwarding rules increases with the cluster/datacenter ratio. This behavior



(A) $\mathbf{B_1}$ (B) $\mathbf{B_2}$

(C) $\mathbf{B_3}$ (D) **MCO**

FIGURE 4.11: Impact of the number of VOs in an MCO network on the number of forwarding rules.



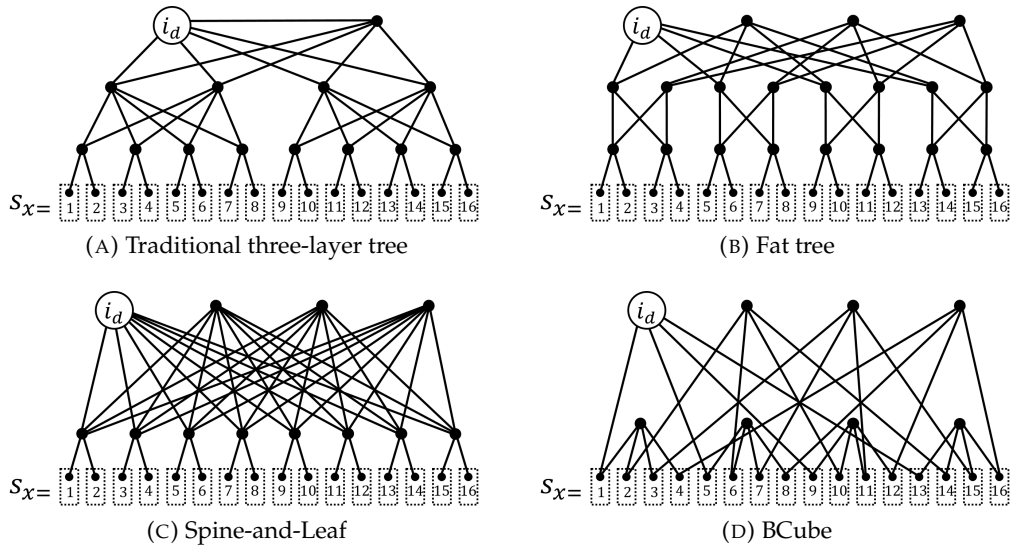
(A) $\mathbf{B_1}$, $\mathbf{B_2}$ and $\mathbf{B_3}$ (B) **MCO**

FIGURE 4.12: Impact of the cluster/datacenter ratio on the number of forwarding rules, given 30 VOs.

of the **MCO** presents a trade-off between scalability and flexibility. Particularly, in handling bulk migrations, flexibility is improved as VOs in V_δ^d with similar QoS/QoE requirements are clustered together – possibly, into multiple centers that can be migrated independently.

Path lengths in the datacenter

Here we consider four datacenter topologies (i.e., Traditional three-layer and Fat trees [123], Spine-and-Leaf [124], and BCube [125]) interconnecting 16 servers, as illustrated in Fig. 4.13. The conventional tree-based architectures are commonly used in datacenters, while the other two are more recently conceived to better support East/West (E/W) traffic. For a given topology, we initially place two VOs v_1 and v_2 in server s_1. Then, path lengths are obtained using the SP and MCO algorithms, varying the location



(A) Traditional three-layer tree (B) Fat tree

(C) Spine-and-Leaf (D) BCube

FIGURE 4.13: Different datacenter topologies interconnecting 16 servers.

of v_2 from s_1 through s_{16}. The eqW cases have all edge weights set to '3', while rndW ones have weights drawn from the discrete uniform distribution $\boldsymbol{U}\{1, 5\}$.

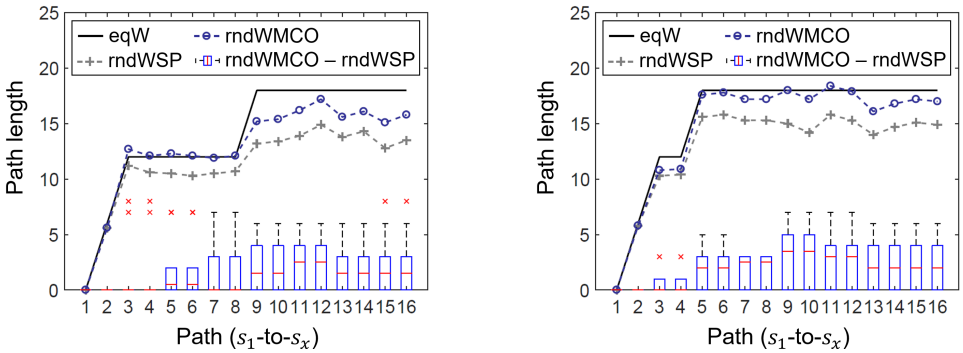
In all four topologies, the paths obtained for both algorithms coincide when all edges in the datacenter have equal weights, as indicated by the **eqW** curves in Fig. 4.14. It can also be observed that indeed the Spine-and-Leaf and BCube architectures yield shorter path lengths on average than the tree-based ones, demonstrating their suitability for E/W traffic support.

Furthermore, Fig. 4.14 shows that when the edges have random asymmetrical weights, our approach (i.e., **rndWMCO** curves) has, on average, higher path lengths compared to SP (i.e., **rndWSP** curves), demonstrating cases of path sub-optimality; differences between the MCO and SP path lengths in all the runs are illustrated by the box plots (i.e., **rndWMCO** $-$ **rndWSP**) for statistical significance. However, it can be observed that, for a given topology, path lengths from s_1 to a subset of servers $\{s_x\}$ correspond or are close to the optimal ones. This implies that knowledge on the datacenter topology can be used to better place VOs of the same overlay in a datacenter.

Although MCO does not fully exploit the path diversity offered by such datacenter topologies, it is important to note that, in a multi-tenant context, centers of different overlays can be mapped to different gateway switches for datacenter load balancing.

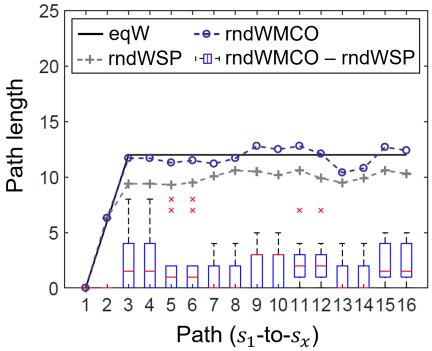
4.6.3 Mobility Support

For the number of wide-area rule updates during seamless bulk migrations, **MCO**'s center migration approach is evaluated with the baseline $\mathbf{B_M}$, which corresponds to the case of multiple VO migrations. Then, we look into how the VOs are clustered into centers – comparing the resulting number of clusters when considering: a) only the inter-VO affinity, and b) both user proximity and inter-VO affinity.

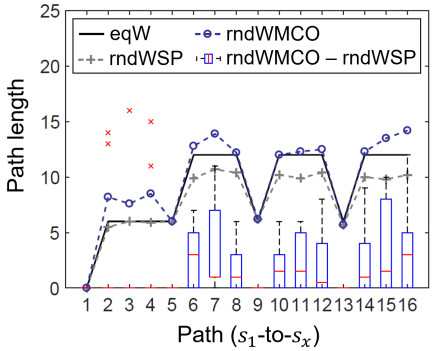


(A) Traditional three-layer tree

(B) Fat tree



(C) Spine-and-Leaf



(D) BCube

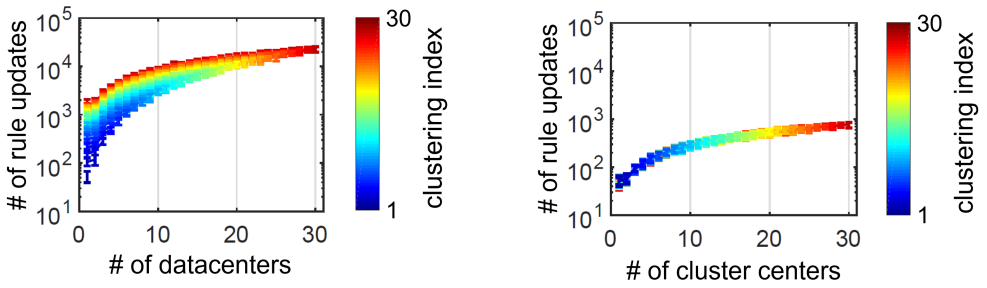
FIGURE 4.14: Path lengths obtained using the shortest path (SP) and MCO algorithms for the different datacenter topologies with equal (eqW) and random (rndW) edge weights.

Impact of the clustering index

Fig. 4.15 illustrates how the number of wide-area rule updates vary with the clustering index and the number of datacenters/clusters involved for MCO and the considered baseline. With a migration initiated for each VO, $\mathbf{B_M}$ shows a stronger dependence on the clustering index, while **MCO** is basically agnostic to the parameter and only depends on the number of clusters involved. Consequently, **MCO** results in up to over one order of magnitude less number of rule updates than $\mathbf{B_M}$.

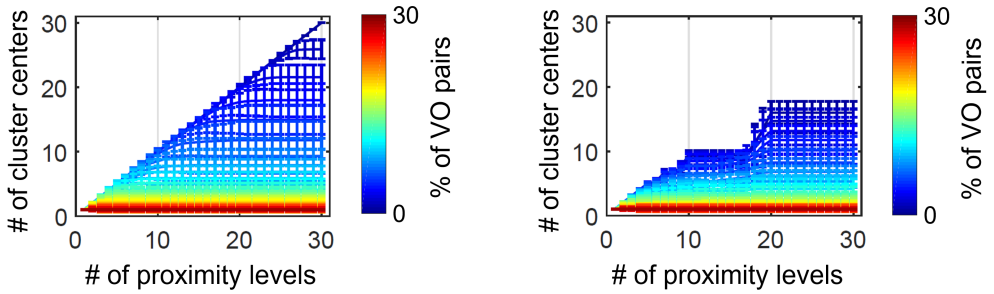
Impact of the user proximity and inter-VO affinity

Varying the percentage of VO pairs generated to have affinity with each other, as well as the number of proximity levels (P_δ) allowed by the user, Fig. 4.16 demonstrates that the number of clusters generally decreases with increasing percentage of VO pairs with affinity among them, although a further reduction is achieved by jointly considering the user proximity and inter-VO proximity. Moreover, the resulting number of clusters in both cases (i.e., $\hat{C}$ and C, respectively) are always less than or equal to the subscription-based parameter P_δ.



(A) $\mathbf{B_M}$ (B) **MCO**

FIGURE 4.15: Impact of the clustering index on the number of rule updates during bulk migrations.



(A) inter-VO affinity (B) user proximity and inter-VO affinity

FIGURE 4.16: Impact of the percentage of VO pairs with affinity among them and proximity levels allowed on the number of VO clusters, given 30 VOs.

4.7 Experimental Results

While MCO has multiple possible applications (e.g., mass-scale services), we further evaluate its performance experimentally by considering the INPUT use case of PN-as-a-service (PNaaS) [78, 129]. Particularly, a PN is a special type of VTN that interfaces the user's private network with the VTN(s) of service providers (referred to as BNs henceforth). In this section, details on the use case, experimental testbed and considered SC scenarios are presented together with the obtained results.

4.7.1 The INPUT Use Case

The INPUT framework seeks to provide seamless experiences to its (mobile) users by guaranteeing a certain level of proximity to the VOs involved – not only in their PNs, but also in the BNs; PN-BN interactions highly depend on users' subscription to services. To do so, it leverages VO portability in the Fog/MEC domain, enabling services to "follow" users, as needed.

In more detail, each VTN (PN/BN) is associated to an MCO network, in which VOs are organized into clusters based on the required proximity level $\mathbf{p}$. Note that the concept of proximity levels – similar to the Differentiated Services Code Point (DSCP) class – provides an abstraction to the actual QoS parameter(s) (e.g., path lengths, latencies,

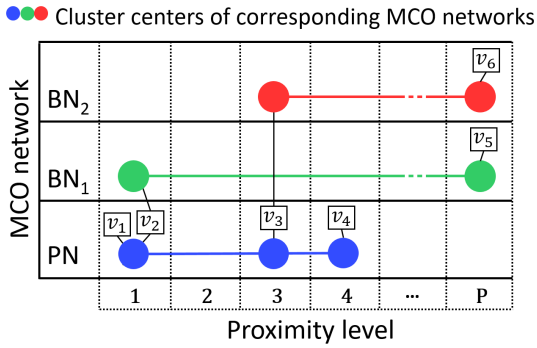


FIGURE 4.17: Clustering VOs based on the required proximity level **p** and network domain (PN/BNs).

available bandwidths, etc.) considered. As users move, center migrations are initiated depending on the current and required **p**.

Fig. 4.17 illustrates an example of a PN that interacts with 2 BNs. As previously anticipated, some VOs (i.e., v_2 and v_3) in the PN are also associated to cluster centers of BNs (i.e., $\mathbf{BN_1}$ and $\mathbf{BN_2}$, respectively). Since a VO v can only reside on a single datacenter, it follows that the PN and BN(s) centers to which v is bound to must have the same proximity level, and be mapped on the same datacenter. Therefore, when a PN center with proximity level **p** is migrated, the corresponding center(s) in the BN(s) is/are also migrated.

The MCO algorithm is currently implemented in the *crater* module of the **OpenVolcano** platform [130] for the INPUT project testbed.

4.7.2 Test Environment

The **OpenVolcano** platform (which is running on a Linux server equipped with an Intel® Xeon® E5-2620 v4 2.10GHz processor) is used to emulate an underlying telecommunication infrastructure, starting from a real wide-area topology (namely, the **Interroute** topology, obtained from the datasets available in [131]).

In more detail, we adapt the **Interroute** topology to consist of 20 in-network datacenter nodes (i.e, randomly selected, and as before, with $H_{min} = 3$ as constraint in order to obtain a solution that maximizes the topological distribution of datacenters) and 90 transit nodes, interconnected by 148 edges (i.e., after removing self-loops). Contrary to the simulation framework, we consider three datacenter *sizes* in the experiments, rather than depths, to indicate the number of servers in each Fog/MEC facility (i.e., d_S (20), d_M (50) and d_L (100)). The size is chosen based on the number of wide-area edges a datacenter has: 2, 3 and > 3, respectively, under the assumption that large datacenters are more central and well connected than small and medium-sized ones. Fig. 4.18 illustrates an example of infrastructure topology generated from adapting the **Interroute** topology in such fashion. For simplicity, but without loss of generality, interconnection switches between the datacenter gateways and the servers are not considered in the emulation.

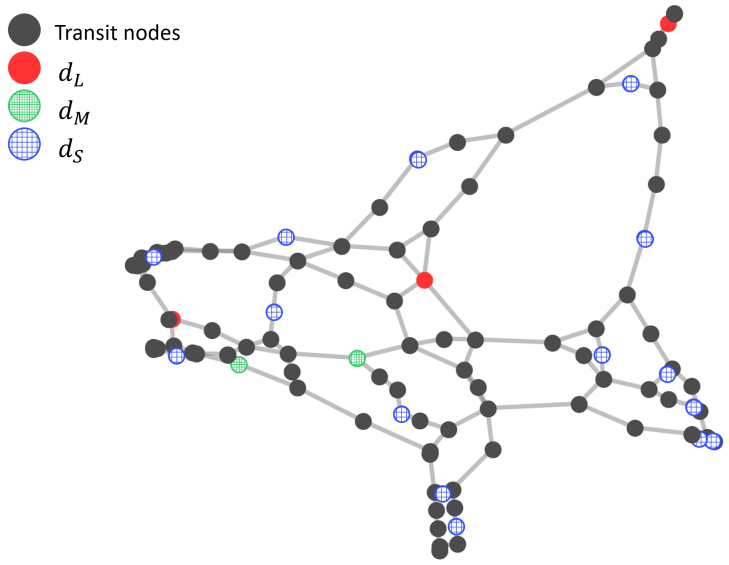


FIGURE 4.18: Example of **Interroute** topology adaptation.

To add statistical significance in the results, each test is repeated 100 times with random CPU and RAM capacities/requirements among servers/VOs, introducing variations in the VO placement inside a datacenter (i.e., based on **OpenVolcano**'s placement policy, which is currently on a testing phase and beyond the scope of this work), from which the 95% confidence intervals are obtained. Nonetheless, any VO placement/consolidation policy can be applied.

4.7.3 Computational Overhead

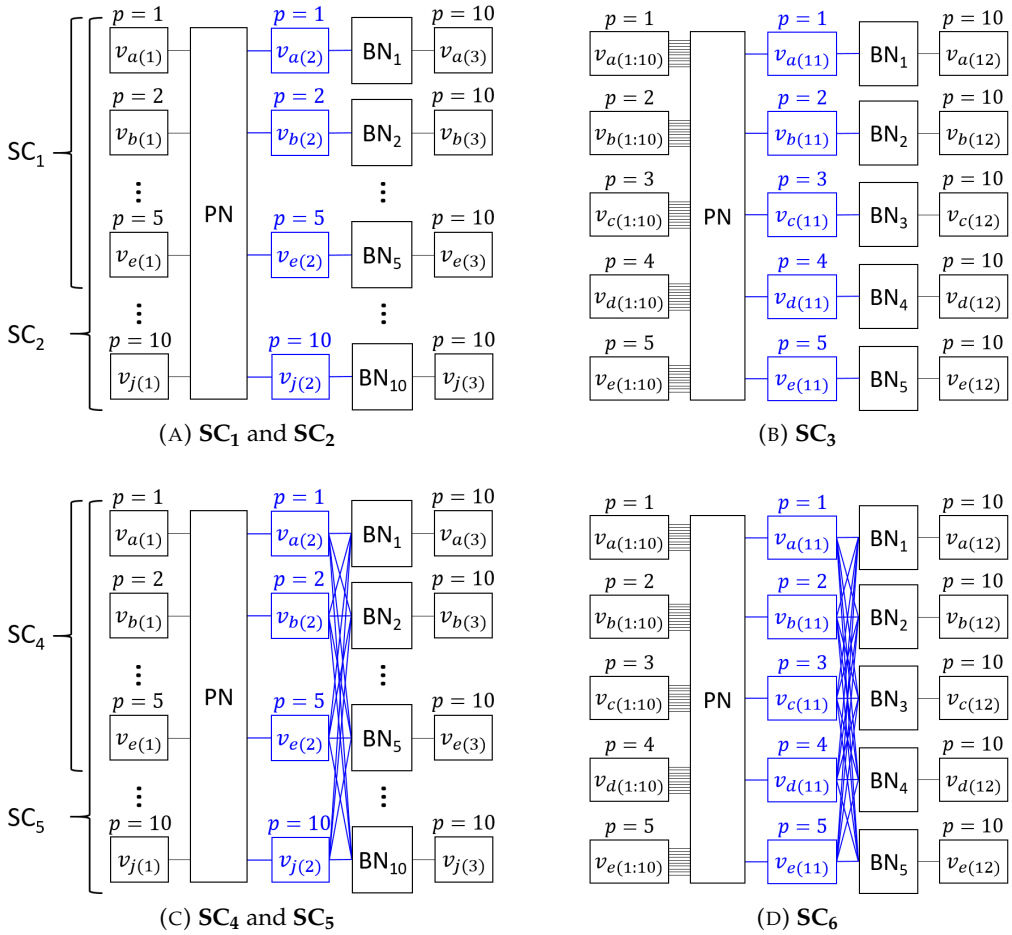
As previously noted, we experimentally evaluate MCO's computational overhead with respect to: a) SC instantiation (in terms of rule calculation times), and; b) center migration (in terms of the number and calculation times of rule updates).

SC instantiation complexity

Six SC scenarios (i.e., $\mathbf{SC_1}, \ldots, \mathbf{SC_6}$) are considered in this work. As illustrated in Fig. 4.19 and summarized in Table 4.3, these SCs are designed to cover variations in the number of BNs (and their interaction with the PN), clusters (PN + BNs) and VOs involved, while keeping modularity to easily automate their generation in the experiments.

Looking at Fig. 4.19, the VOs highlighted in blue (i.e., $v_{a(2)}, \ldots, v_{j(2)}$ or $v_{a(11)}, \ldots, v_{e(11)}$) are the ones that are identified with two or more cluster centers (i.e., one in the PN and another one in each BN connected); recall that all centers associated to the same VO must have the same p value and be mapped on the same datacenter.

Note that the time required to deploy SCs highly depends on the instantiation of their respective L2 overlay connectivity. With MCO, the rules required to instantiate either of the six SCs can be calculated in less than 350 ms, as shown in Fig. 4.20; this demonstrates MCO's low computational complexity, even for more intricate SC scenarios (e.g., $\mathbf{SC_5}$ and $\mathbf{SC_6}$).



(A) **SC$_1$** and **SC$_2$** (B) **SC$_3$**

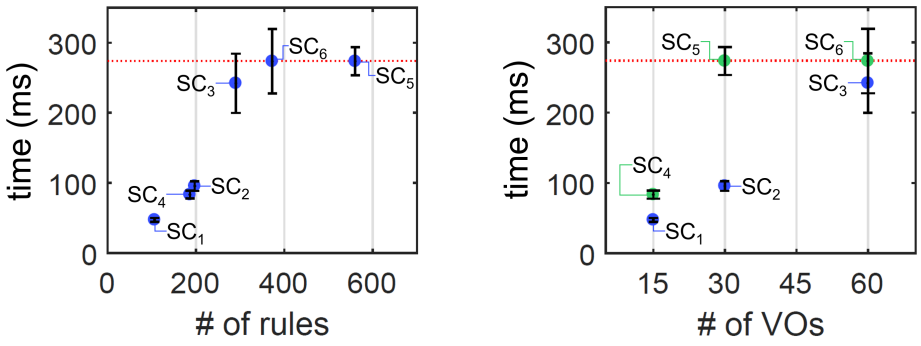
(C) **SC$_4$** and **SC$_5$** (D) **SC$_6$**

FIGURE 4.19: SC scenarios.

TABLE 4.3: MCO parameters for the considered SC scenarios.

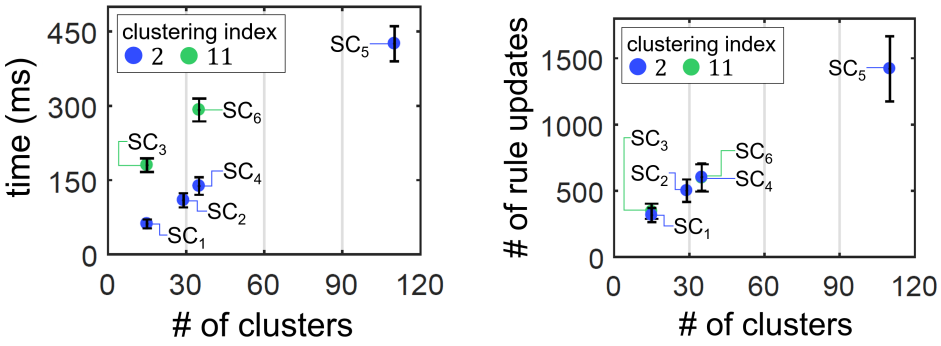
SC	# of BNs	# of VOs (PN + BNs)	# of Clusters		# of VOs/cluster	
			PN	BNs	PN	BNs
SC$_1$	5	15	5	10	2	2
SC$_2$	10	30	10	19	2	2
SC$_3$	5	60	5	10	11	2
SC$_4$	5	15	5	30	2	1
SC$_5$	10	30	10	100	2	$1 \sim 2$
SC$_6$	5	60	5	30	11	1

It is interesting to note that while the number of rules increases with SC complexity, the average rule calculation time approaches a limit of around 273 ms, as indicated by the red dotted lines. In Fig. 4.20a, this sublinear behavior manifests the impact of PN-BN interactions, mainly driven by both the number of VOs and VTNs involved — for instance, **SC$_3$** and **SC$_6$** correspond to the highest number of VOs (at least twice than



(A) vs the # of rules (B) vs the # of VOs

FIGURE 4.20: Rule calculation times for SC instantiation.



(A) calculation times (B) # of rule updates

FIGURE 4.21: Rule updates and calculation times during a center migration, for varying number of clusters involved (PN + BNs).

other SCs), while **SC$_5$** to the highest number of BNs (over three times than other SCs). In Fig. 4.20b, a similar relationship is also observed between the measured times and the number of VOs involved, with the slope determined by the PN-BN interactions, until the said limit is reached. Wider confidence intervals are observed for SCs involving more VOs (i.e., **SC$_3$** and **SC$_6$**) since the VO clusters are more likely to be distributed to different number of servers in each test.

Center migration overhead

Considering the same SCs (which also cover variations in the clustering index), we emulate a center migration for c_1 (i.e, $p = 1$) of the PN to various destination datacenters in the telecommunication infrastructure, with distances between d_{old} and d_{new} ranging from 8 to 12 hops. As previously mentioned, the corresponding cluster center(s) (i.e, $p = 1$) in the involved BN(s) will also be migrated to d_{new}.

Note that a migration can only begin when the necessary updates in the L2 overlay connectivity are already in place. The rule updates required for the center migration in either of the six SCs can be calculated in less than 500 ms, as illustrated in Fig. 4.21a; this demonstrates that the time overhead incurred by MCO's center migration approach for seamless bulk migration is practically negligible with respect to the total migration

duration (e.g., tens to hundreds of seconds [78], and highly depends on the virtualization/migration technologies used, workloads and migration paths [82, 106, 132]).

It can be observed in Fig. 4.21b that the number of rule updates increases linearly with the number of clusters involved (PN + BNs), almost independently of c_1's clustering index; the latter only impacts the number of rule updates inside the datacenters, which is expected of MCO's center migration approach. A wider confidence interval is observed for $\mathbf{SC_5}$ since migrating the PN's center c_1 also migrates the corresponding centers (i.e, $p = 1$) of BNs $\mathbf{BN_1}$ through $\mathbf{BN_{10}}$, resulting in a multiplicative effect in the variation of the number of rule updates for different distances between d_{old} and d_{new}.

A similar linear behavior is also observed for the calculation times, except that the $y-$intercept is determined by the clustering index. Particularly, the increase in the calculation times with increasing clustering index can be attributed to the (re-)running of the shortest-path algorithm for each of the VOs being migrated, in both d_{old} and d_{new}; still, the increase in time is not linearly proportional to the increase in the clustering index.

4.8 Summary

In this chapter, a novel multi-clustering approach, called MCO, is proposed to address the scalability issue of SDN. Particularly, MCO realizes geo-distributed VTNs and effectively supports dense deployments of mobile VOs at the network edge, as well as seamless user/services mobility through bulk inter-datacenter VO live migrations, with significantly reduced number of OF rules and low computational overhead.

The L2 connectivity among VTN endpoints are built on the basis of paths and shortest-path trees. MCO's OF rules for unicast and broadcast/multicast forwarding, fixed and mobile access terminations, as well as for seamless migrations of VOs (both intra- and inter-datacenter) are then defined accordingly. In addition, a simple poximity- and affinity-aware VO clustering policy is described as an example of how to bundle VOs into cluster centers, and ultimately, considered as an aggregate entity in the wide-area.

In contrast to state-of-the-art SDN mechanisms (i.e., fully-meshed, two-level flow aggregation and OpenStack cases), numerical results show that MCO achieves up to over one order of magnitude smaller number of OF rules in the VTN implementation and rule updates during center migrations, demonstrating its high scalability. Moreover, possible path sub-optimality may occur, albeit knowledge on the datacenter topology can be exploited for VO placement optimization and load balancing.

MCO's performance has also been experimentally evaluated using the **OpenVolcano** platform in the context of the INPUT use case (i.e., PNaaS). Experimental results demonstrate its low computational complexity in terms of rule and rule update calculation times during SC instantiation and wide-area bulk migrations, respectively.

Chapter 5

Applications to Network/Services Management and Control

Now, as it is the end goal of this thesis, how do we enable scalable and sustainable softwarized 5G environments? From the discussions made up to this point, this can be generally translated into:

- optimizing the trade-off between power and performance in the underlying COTS hardware according to their corresponding workload dynamics;
- realizing the connectivity among (possibly, geo-distributed) network service components, and effectively support its dynamic reconfiguration; and
- steering traffic flows according to the current (re)configuration of their corresponding services in a seamless fashion,

with low computational and network overhead, as well as low technological requirements. Chapters 3 and 4 detailed two mechanisms that proved to be handy in achieving these objectives.

This chapter presents some spin-off works that apply the model-based analytics and MCO mechanisms of the previous chapters, as well as a more recent work based on team theory, to network/services management and control in softwarized 5G environments. Specifically, we address the problems of VNF consolidation, service migration with user mobility and load balancing among multiple service instances, in each of the following sections, respectively.

5.1 Joint Power Scaling and Consolidation

(In-network) datacenters are expected to be deployed practically anywhere in the unified 5G infrastructure, and host execution containers or VOs (e.g., VMs, Linux containers, etc.) running VNFs (or VNFCs), among others. In the context of datacenter networks, this section presents a power- and performance-aware resource allocation scheme to manage the latter among a pool of ACPI-enabled physical resources (processors/cores).

Supposing that a real-time analytics mechanism for workload profiling and estimation of network KPIs (specifically, power) is already in place, resources are dynamically

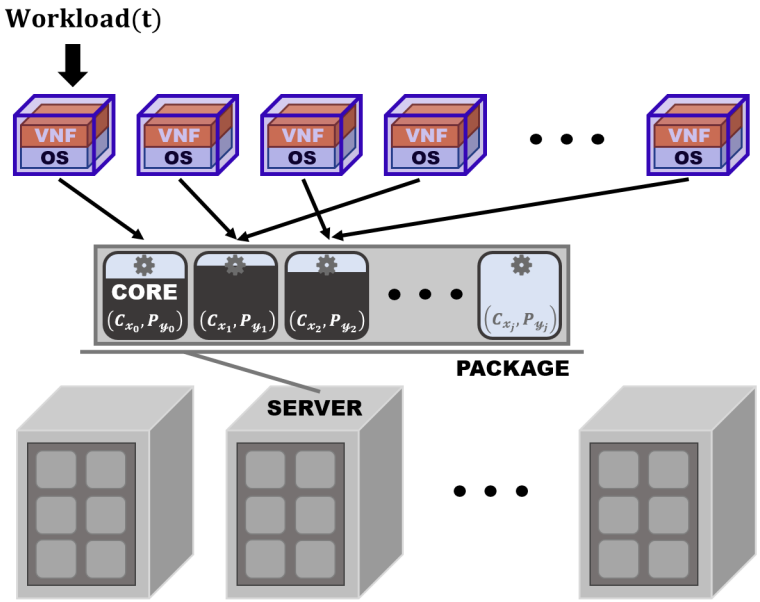


FIGURE 5.1: High-level view of the VNF consolidation approach.

managed by jointly performing power scaling and in-server consolidation according to the actual VNF workload variations. In particular, a power consumption model is incorporated in the consolidation decision to take into account the power management capabilities (i.e., AR and LPI) of the underlying COTS hardware.

Towards this end, we consider a set of VMs running VNFs (or VNFCs) dedicated to perform certain network functionalities on incoming traffic streams of various nature. For the sake of simplicity, a one-to-one correspondence between VNFs and VMs is supposed; the rationale behind this is that for a VNF consisting of multiple VMs (one for each VNFC), the overall VNF performance can be derived from the individual VM performances according to the chaining defined by the VNF provider. In any case, the VNF consolidation reduces to a VM consolidation problem.

The classical First-Fit Decreasing (FFD) bin-packing algorithm [133] is considered as a baseline for VM consolidation, although the approach can be easily adapted to other packing algorithms. VMs are initially placed among a given set of multicore servers through FFD based on the workloads specified in the SLA. Since such specifications are generally based on peak workloads, our goal is to dynamically manage VM consolidation in each server according to actual workload variations by jointly tuning the ACPI configuration and minimizing the number of active cores, as shown in Fig. 5.1.

5.1.1 System Modeling

Consider to have Λ servers, each containing a set of multicore processor packages. The VMs are mapped among these servers, and generally, each VM can be allocated a certain number of cores, depending on the processing requirements of the running applications inside. In this work, however, we limit the VM workloads to be less than the maximum core capacity for the sake of simplicity, and each core serves a subset of VMs.

Queueing model

Suppose that each VM has its own queue and is allocated a vCPU that is subject for scheduling. Although this suggests a multiple-queue single-server queueing model for the core, an equivalent single-queue model can be easily derived considering the aggregate workload, as described in [134].

Recalling the model considered in Chapter 3, and applying it in a core sharing scenario, we suppose that the system has Γ cores, each one modeled as an $M^X/G/1/SET$ queue, with an average packet service rate $\mu^{(j)}$, aggregate workload $OL^{(j)} = \lambda^{(j)}\beta^{(j)}_{(1)}$ and core utilization $\rho^{(j)} = OL^{(j)}/\mu^{(j)}$, $j \in \{1, \ldots, \Gamma\}$. In order to make the equations hereinafter more readable, we omit the index $^{(j)}$ of the core, and express $\beta_{(1)}$ simply as β.

Traffic model

Suppose that the traffic incoming to the i-th VM is represented as a BMAP with batch arrival rate λ_i and average batch size β_i, $i \in \{1, \ldots, |\mathcal{I}|\}$, where $|\mathcal{I}|$ is the cardinality of the set $\mathcal{I}$ of VMs served by a core.

Since the sum of independent Poisson processes is a Poisson process with rate given by the sum of the individual rates, then the batch arrival rate at the core is given by

$$\lambda = \sum_{i=1}^{|\mathcal{I}|} \lambda_i \tag{5.1}$$

Moreover, the average batch size at the core is obtained as

$$\beta = \frac{1}{\lambda} \sum_{i=1}^{|\mathcal{I}|} \lambda_i \beta_i \tag{5.2}$$

Both parameters can be easily obtained using the approach detailed in Chapter 3.

Power and performance model

For a given workload at a core, the power model expressed in Eq. (3.10) can be used to determine which ACPI configuration gives the minimum power consumption.

In this first evaluation of the consolidation strategy, we focus on assessing the potential power saving by keeping the enforcement of performance constraints in the simplest possible form. Specifically, as we will see in the following sub-section, we will only impose a limit on the maximum utilization for each core to avoid unacceptable performance degradation. However, the performance of a VNF can be better controlled by considering the average system latency D (i.e., Eq. (3.12)), and is left for future work.

5.1.2 Power- and Performance-aware Consolidation

Suppose a datacenter performs a global FFD bin-packing consolidation every time period T. Considering that in typical consolidation policies significant variations in the

total number of active servers and VMs might occur on a much longer time–scale than the dynamics of power scaling policies, we can suppose the time interval T to be in the order of tens of minutes or even hours. Similarly, the number of VMs in each server can be considered relatively stationary over shorter time intervals ΔT. Then, we assume that the number of VMs in the system does not change in this interval (i.e. no arrival and/or departure of VMs – if any occur, they will be accounted for with some delay in the successive interval). We propose a consolidation policy that can be performed in each active server at every sub-interval ΔT during T, dynamically managing the VM consolidation among cores according to actual workload variations.

Firstly, we adopt a slightly looser provisioning than the core network capacity planning rule-of-thumb, constraining the offered workload to be less than 80% of the maximum capacity, instead of 50% [135]. This still provides a safety headroom for any fluctuation in the VMs' workload, ensuring that the required QoS constraints are not violated [136]. Additionally, we limit the choices of power states $\{C_x\}$ based on their respective sleeping times (which we express as $\tau_s(C_x)$), and the aggregate batch arrival rate at the core such that $\tau_s(C_x) < 1/\lambda$, in order to have a relatively low probability of arrivals during $C_0 \rightarrow C_x$ transitions.

The scheme also includes classification rules that define the most energy efficient configuration to be applied to the core, given the statistical features of its workload. Specifically, given a certain aggregate workload (characterized by λ and β), we evaluate the power model for all possible pairs of (C_x, P_y) – i.e., $\Phi(C_x, P_y)$, ensuring at the same time the satisfaction of the aforementioned utilization constraint, in order to find the ACPI configuration that yields the minimum average power consumption. It is worth noting that this computation can be performed offline for a whole range of λ and OL values, for a specific processor architecture, giving rise to regions in the (λ, OL) space that correspond to the most suitable configuration for the points in the region (a specific example of such regions will be provided in Sub-section 5.1.3). When two or more configurations give the minimum value, the one with the better performance (i.e., greater capacity and/or lighter sleeping state) is selected.

Taking into account that some states are set on a per-package basis [137], for the sake of simplicity in the power management of the server farm, we suppose that all cores in a processor package have the same configuration. Hence, if the optimum configuration varies among the cores in a package, a suboptimal solution is to set them according to the one with the highest requirement.

Supposing that the workload in each VM can be monitored via its vCPU's usage, we exploit the obtained data to perform a dynamic VM consolidation inside the server. Now, let Γ_k indicate the set of cores in the k-th server, and $\mathcal{I}_k = \bigcup_{j=1}^{\Gamma_k} \mathcal{I}_k^{(j)}$ the set of VMs they serve, where $\mathcal{I}_k^{(j)}$ is the subset of VMs served by the j-th core. Based on the actual VM workload in a ΔT sub-interval, the FFD algorithm is jointly performed with power scaling to find the minimum core capacity required to serve $\mathcal{I}_k$, obtaining $|\Gamma_k|$ updated groupings of VMs, $\{\hat{\mathcal{I}}_k^{(j)}, j \in \Gamma_k\}$. At this point, some groups may be empty – this only means that some cores will be idle. Considering the new aggregate workload of each

Algorithm 5.1 Consolidation Policy

$\mathcal{I}_k, \lambda_i, \beta_i, \forall i \in \{1, \ldots, |\mathcal{I}_k|\}$
$\mathcal{P} \leftarrow \{\}$
for $\mathcal{y} = 0$ to $\mathcal{Y}$ **do**
  $\mu^{(j)} \leftarrow \mu(P_{\mathcal{Y}-\mathcal{y}}), \forall j \in \Gamma_k$
  perform the FFD algorithm to obtain $\hat{\mathcal{I}}_k^{(j)}, \forall j \in \Gamma_k$
  if all VMs have been allocated **then**
    $\mathcal{P} \leftarrow \{P_0, \ldots, P_{\mathcal{Y}-\mathcal{y}}\}$
    $\{\mathcal{I}_k^{(j)}, \forall j \in \Gamma_k\} \leftarrow \{\hat{\mathcal{I}}_k^{(j)}\}, \forall j \in \Gamma_k\}$
    break
  end if
end for
for $j \in \Gamma_k$ **do**
  $\lambda^{(j)} \leftarrow \sum_{i=1}^{|\mathcal{I}_k^{(j)}|} \lambda_i$
  $OL^{(j)} \leftarrow \sum_{i=1}^{|\mathcal{I}_k^{(j)}|} \lambda_i \beta_i$
  $\mathcal{C}^{(j)} \leftarrow \{\}, \forall j \in \Gamma_k$
  for $x = 0$ to $\mathcal{X} - 1$ **do**
    if $\tau_s(C_{\mathcal{X}-x}) < \frac{1}{\lambda^{(j)}}$ **then**
      $\mathcal{C}^{(j)} \leftarrow \{C_1, \ldots, C_{\mathcal{X}-x}\}$
      break
    end if
  end for
  evaluate Eq. (3.10) considering the states in $\mathcal{P}$ and $\mathcal{C}^{(j)}$, and apply classification rules
end for
re-order $\{\mathcal{I}_k^{(j)}, \forall j \in \Gamma_k\}$ with decreasing performance requirement
for $j \in \Gamma_k$ **do**
  allocate a core to $\mathcal{I}_k^{(j)}$
end for
set processor package to most suitable configuration

group, the core classification rules are then applied to determine their suitable configuration. The groups are ordered with decreasing performance requirement and each one is allocated a core accordingly. Finally, the configuration of each processor package is set according to the core with the highest requirement and all idle packages are put to a deep sleep. The proposed power- and performance-aware consolidation policy is summarized in Algorithm 5.1.

5.1.3 Performance Evaluation

In order to evaluate the performance of the consolidation policy, a simulation framework for a scaled-down datacenter is implemented in Matlab, considering Intel Xeon E5-2690 2.9GHz processors. Besides being configurable through the ACPI, this processor has also been used by Intel in evaluating the DPDK virtual switch (vSwitch) for NFV [138].

TABLE 5.1: ACPI configuration parameters for the consolidation evaluation.

Parameter	ACPI Configuration			
	(C_1, P_0)	(C_3, P_0)	(C_1, P_8)	(C_3, P_8)
Φ_a (W)	16.75	16.75	10.5	10.5
Φ_t (W)	40	25	17	13
Φ_i (W)	7	5.4	7	5.4
μ (pps)	891875	891875	515539	515539
τ_p (μs)	100	200	100	200
τ_s (μs)	5	10	5	10
Φ_{s-oh} (W)	195			
Φ_{c-ds} (W)	2			

For the sake of simplicity, but without loss of generality, we only consider the following subset of configurations: $\{(C_1, P_0), (C_3, P_0), (C_1, P_8), (C_3, P_8)\}$, in this work. The C_1 and C_3 power states correspond to the *Halt* and *Sleep* modes, respectively. In the former, almost all clock signals to the core are stopped; although no instructions are executed in this state, the core continues to process bus snoops and can return to the C_0 state almost instantaneously. Conversely, all core clocks are stopped in the latter, and the core's Level 1 and Level 2 caches are also flushed, requiring a significantly longer wake-up time. The P_0 and P_8 performance states are supposed to correspond to the maximum and minimum core frequencies, respectively. Based on the data presented in [103, 138, 139], the parameter values listed in Table 5.1 can be derived. Φ_{s-oh} is the power consumption overhead for each active server, and Φ_{c-ds} is the power consumption of a core in deep sleep.

Core classification

We express the power consumption model as a function of λ and OL by substituting the given configuration parameters into Eq. (3.10). By varying the workload and applying the core classification rules, regions in the (λ, OL) space are obtained. Fig. 5.2 shows that the resulting regions are simply defined by constant discriminant functions. However, as we add more variables (e.g., latency constraint, trade-off parameters, etc.) into the rules, we also expect to add complexity into these functions.

Numerical example

For this example, we consider a system with 500 servers and 10000 VMs. Each server is supposed to have 2 octa core processors, as in [138].

The maximum workload $[OL_i]^{max}$ of each VM is generated from the continuous uniform distribution $\boldsymbol{U}(0.05\mu_{max}, 0.8\mu_{max})$. Similarly, to consider the time variance of

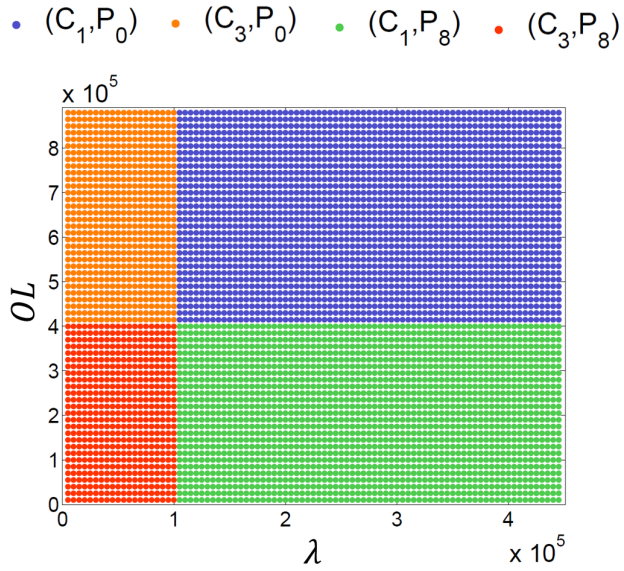


FIGURE 5.2: Example of regions defined by the core classification rules.

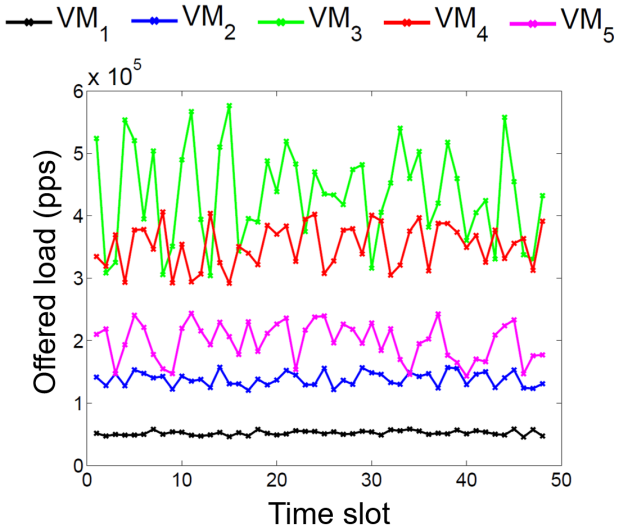


FIGURE 5.3: VM workload variations.

the VMs' workloads, the minimum workload $[OL_i]^{min}$ is generated from $\boldsymbol{U}(0.5[OL_i]^{max}, 0.8[OL_i]^{max})$, $i \in \{1, \ldots, 10000\}$. In addition, the minimum β_i^{min} and maximum β_i^{max} average batch sizes are generated from the discrete uniform distributions $\boldsymbol{U}\{1, 15\}$ and $\boldsymbol{U}\{\beta_i^{min} + 1, 50\}$, respectively.

Moreover, the number of sub-intervals is set to 48 – this corresponds to the number of times the consolidation policy is performed in a simulation run. In each sub-interval, the workloads vary according to $\boldsymbol{U}([OL_i]^{min}, [OL_i]^{max})$, while the batch sizes according to $\boldsymbol{U}(\beta_i^{min}, \beta_i^{max})$. Fig. 5.3 shows the workload variations of five representative VMs in the system, illustrating the diversity among the VM workloads and behaviors in this evaluation. As regards the reliability of the results, 10 runs with different seeds are performed for a given set of VMs to show the 95% confidence intervals through error bars.

Taking a look at the entire system, Fig. 5.4 illustrates the average workload of the

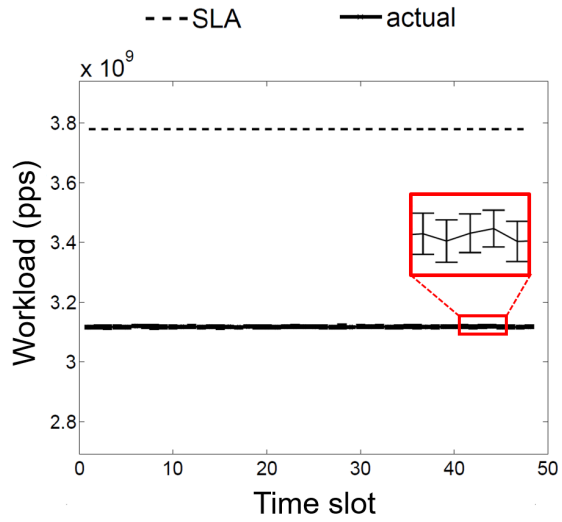


FIGURE 5.4: Datacenter workload.

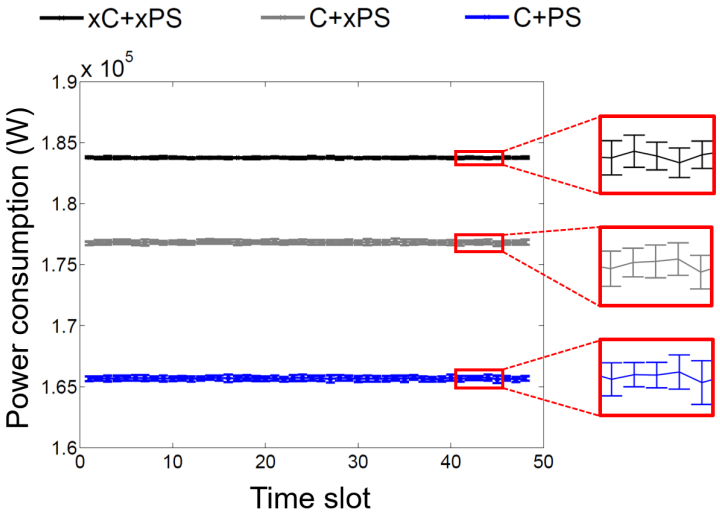


FIGURE 5.5: Datacenter power consumption.

scaled-down datacenter. Despite the workload variations among the VMs, the total workload in the system remains stable at around 3.1 Gpps, which is approximately 18% lower than the one specified in the SLA.

In order to assess the power saving potential of the approach, we compare the average datacenter power consumptions generated by the proposed policy (**C+PS**) and two baseline scenarios – (**xC+xPS**) and (**C+xPS**). **C** and **PS** denote in-server consolidation and power scaling, respectively, and **x** is appended to indicate the absence of the specified capability. Fig. 5.5 shows that simply performing in-server consolidation reduces the power consumption by around 4%, which can be further improved to 10% when jointly performed with power scaling.

To better grasp the impact of this improvement, we can put it into figures by deriving the annual savings ς of the datacenter as:

$$\varsigma = \phi \cdot \left[\tilde{\Phi}_{(\mathbf{xC+xPS})} - \tilde{\Phi}_{(\mathbf{C+PS})}\right] \cdot \frac{1}{1000} \cdot 24 \cdot 365 \tag{5.3}$$

where ϕ is the energy cost per kilowatt hour (kWh). Based on the values reported in [140], electricity prices for industrial consumers in the European Union averaged 0.12 €/kWh during the second half of 2014. With this, we obtain around 19000 € of annual savings for the scaled-down datacenter considered in this example, which is quite a good number given its size. Now, imagine the potential savings for a datacenter operator like Google that has been estimated to run over 1 million servers since a couple of years back.

5.2 User-centric Service Migration

On the other hand, looking at the service level, softwarization enables portability of network service components across the unified 5G infrastructure, granting NSPs more degrees of freedom in the dynamic management of their services. In the context of supporting user mobility, this section presents a proximity- and affinity-aware service migration approach built on MCO's center migration (see Chapter 4).

Particularly, we suppose that a user is subscribed to certain services, and each service component (referred to as a VO hereinafter) in his/her PN (as well as the ones in the third parties' BNs) corresponds to certain proximity and affinity requirements. Taking into account both requirements, we seek to enable services components to "move with the user" in a scalable and differentiated fashion, through a VO clustering and migration policy.

Fig. 5.6 illustrates our conceptual framework in an example based on the *virtual Set-Top-Box (vSTB)* use case evaluated in [78], considering the vSTB *service applications* as the user's VOs. Both the user's private domain (e.g., PN) and the third parties' shared domain (e.g., BNs) are indicated through a multi-point link model, and the VOs' QoS/QoE requirements through user proximity levels; note that VO clusters with lower proximity levels may require migrations more often than those with higher proximity levels.

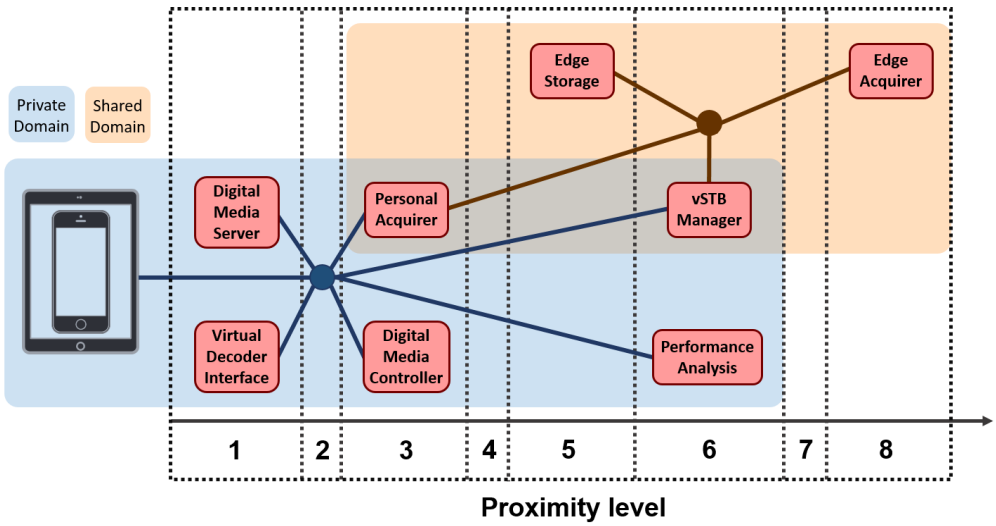


FIGURE 5.6: Example based on the vSTB use case in [78].

5.2.1 System Description

Suppose that each user u is associated to a set of VOs V_u that can be placed in a distributed and dynamic fashion within the set of Cloud/Fog/MEC (in-network) datacenters D. As u moves from one access point to another in each time instant, (bulk) migrations may be necessary to meet the close proximity requirements of some VOs; additionally, such VOs may be tightly coupled to other VOs with loose proximity requirements (i.e., as SCs).

Proximity and affinity requirements

As in Chapter 4, the user proximity requirements $\Pi_u = \{\Pi_u(i), i = 1, \ldots, |V_u|\}$ are expressed in terms of path lengths (i.e., from user u's access device to the subset of datacenters $D_u \subseteq D$ currently hosting his/her VO network), under the assumption that other QoS parameters are already represented in the link weights; for instance, larger weights can be assigned to high latency/low bandwidth links, resulting in "longer" path lengths.

On the other hand, inter-VO affinities highly depend on the user's service subscriptions, and hence, the inter-VO interactions that may arise. For the sake of simplicity, but without losing generality, we express the inter-VO affinity requirements $\pi_u = \{\pi_{(i,j)}, i = 1, \ldots, |V_u|, j = 1, \ldots, |V_u|, i \neq j\}$ in terms of 0 or ∞ distances – i.e., with $\pi_{(i,j)} = 0$, u's VOs $(v_i, v_j) \in V_u, i \neq j\}$, must be placed in the same datacenter, otherwise they can be placed in any datacenter $d \in D$, provided that the requirements in Π_u are met.

Cluster center model

Recall that VOs can be clustered together based on both Π_u and π_u, as well as the subscription-based parameter P_u (that indicates the maximum number of proximity levels allowed by the user u), and each cluster can be mapped to a certain proximity level $\mathbf{p}$, as described in Chapter 4. Particularly, a two-step VO clustering is performed by:

S1: considering π_u to obtain an initial set of clusters $\hat{C}$, with each cluster $\hat{c} \in \hat{C}$ corresponding to the minimum requirement $min(\Pi_u^{\hat{c}})$, and;

S2: considering $\{min(\Pi_u^{\hat{c}}), \hat{c} \in \hat{C}\}$ to find the the range intervals of the P_u proximity levels, and obtain the final set of clusters C.

Then, each VO cluster $c \in C$ is dynamically placed according to the minimum proximity requirement $min(\Pi_u^c)$ among VOs in c and u's current access point.

5.2.2 Proximity- and Affinity-aware Center Migration

Suppose that each datacenter $d \in D$ has enough resources for hosting VOs, focusing on the QoS improvement achieved by allowing VO clusters to "move with the user", when necessary. Particularly, as user u moves around throughout the day – e.g., from home to

Algorithm 5.2 Center migration at time instant t

In: C, $\{min(\Pi_u^c), c \in C\}$, $\{D_u^c(t-1), c \in C\}$, $\mathbf{u} \mapsto \mathbf{ac}(t)$
$\{L_c, c \in C\} \leftarrow distances(\mathbf{u}, \{D_u^c(t-1)\})$
$\{D_u^c(t), c \in C\} \leftarrow \{\}$
for $c \in C$ **do**
  if $L_c > min(\Pi_u^c)$ **then**
    $S \leftarrow shortestpath(\mathbf{ac}(t), D_u^c(t-1))$
    $L_S \leftarrow distances(\mathbf{u}, S)$
    for $i = 0$ to $|S| - 1$ **do**
      if $L_S(|S| - i) \leq min(\Pi_u^c)$ **then**
        $D_u^c(t) \leftarrow S(|S| - i)$
        break
      end if
    end for
  else
    $D_u^c(t) \leftarrow D_u^c(t-1)$
  end if
end for
Out: $\{D_u^c(t)\}$

work or to do some errands, etc., and then, back home – some of the clusters' proximity requirements may be violated at some point, necessitating bulk VO migrations to keep the desired QoS.

For instance, at a time instant t, the network detects that u's access point changed from $\mathbf{ac}(t-1)$ to $\mathbf{ac}(t)$, and $\{D_u^c(t-1), c \in C\}$ is the previous placement of the clusters (i.e., the datacenter locations that meet $\{min(\Pi_u^c),\ c \in C\}$ when u was connected to $\mathbf{ac}(t-1)$). Algorithm 5.2 summarizes how migrations are initiated at such time instants, where $\{L_c, c \in C\}$ are the shortest-path lengths from u's device $\mathbf{u}$, via $\mathbf{ac}(t)$, to the previous placement $\{D_u^c(t-1)\}$, and $\{D_u^c(t), c \in C\}$ is the new placement obtained.

For a given cluster c, a migration is only initiated if L_c exceeds $min(\Pi_u^c)$. In such a case, the shortest path S between $\mathbf{ac}(t)$ to c's previous location $D_u^c(t-1)$ is obtained, as well as the corresponding path lengths L_S from each of its hops to $\mathbf{u}$. Starting from the hop closest to $D_u^c(t-1)$, the first one that satisfies $min(\Pi_u^c)$ is chosen as c's new location.

5.2.3 Performance Evaluation

In order to assess the QoS improvement and service management simplification offered by cluster migration, a series of numerical evaluations are performed on a graph-based logical topology.

Generally, VOs can be hosted in the Cloud, Fog and/or MEC domains; hence, we classify datacenters as: a) cloud (**cl**), b) transit/aggregation (**t/a**) or c) access (**ac**) nodes. Through a simulation framework implemented in Matlab, we consider a scaled-down, city-wide telecommunication infrastructure with 30 datacenters: 2 are **cl** nodes (e.g., like Telecom Italia's Sparkle nodes in Milan [141]), while the rest are **t/a** and **ac** nodes generated according to the probability mass function $\mathcal{P} = \{0.4, 0.6\}$, respectively. The

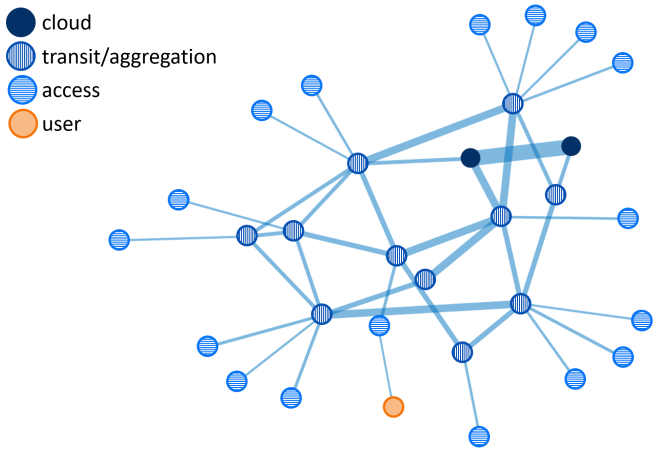


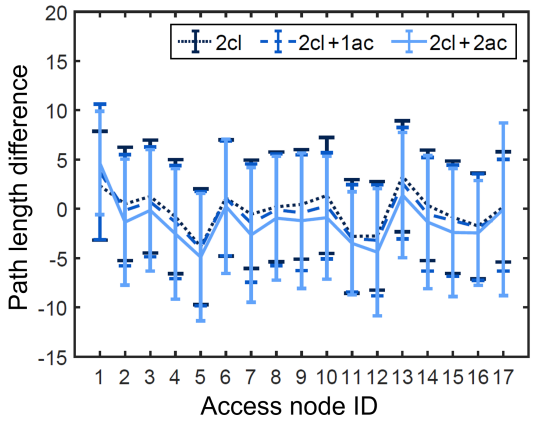
FIGURE 5.7: Graph-based logical topology of a scaled-down city-wide infrastructure.

logical interconnections E among these nodes are randomly generated to form a graph $G(D, E)$ – except for the 2 **cl** nodes that are supposed to be part of the nationwide network backbone.

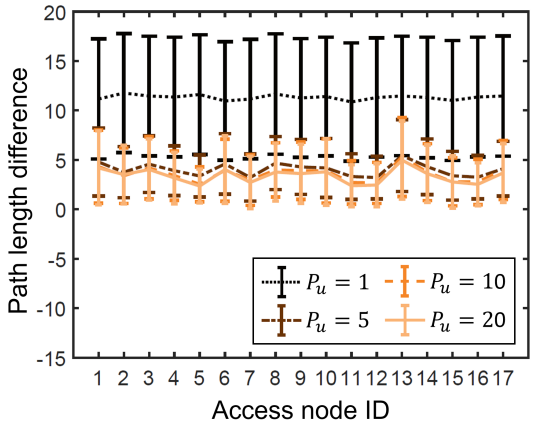
The links interconnecting any pair of datacenters $(d_n, d_m) \in D$, $n \neq m$, are characterized by their corresponding weights $\{w_{(n,m)}\}$. As previously noted, one way to incorporate the latency/bandwidth aspect in a QoS evaluation based on path lengths is by assigning higher weights to high latency/low bandwidth links. With this in mind, the link between the 2 **cl** nodes has weight set to '1', links between **cl** and **t/a** nodes or between 2 **t/a** nodes have weights drawn from the discrete uniform distribution $\boldsymbol{U}\{2, 4\}$, while those interconnecting **t/a** and **ac** nodes from $\boldsymbol{U}\{5, 7\}$. Finally, the link between a user u's device **u** and an **ac** node has weight drawn from $\boldsymbol{U}\{8, 10\}$. Fig. 5.7 shows an example of a graph-based logical topology generated in such fashion, where the widths of edges decrease with increasing link weights.

We further suppose that a user u has 20 VOs with proximity requirements drawn from $\boldsymbol{U}\{10, 30\}$ – this range of values is chosen based on the link weights to cover different user proximity cases. Particularly, VOs are uniformly generated such that some require to be on or close to the current **ac** node, some with '*don't care*' proximity, while others are somewhere in between these two extremes. Moreover, the inter-VO affinity is specified in terms of percentage (i.e., 0%, 5% and 10%, in this work), which corresponds to the percentage of VO pairs $(v_i, v_j) \in V_u$, $i \neq j$, with $\pi_{(i,j)} = 0$.

In this evaluation, we consider both static and dynamic user cases, studying the differences between the required (SLA) and actual path lengths from user u to his/her VOs', with and without migrations. Additionally, for the latter case, we take a look at the number of migrations initiated in terms of VOs and clusters, to provide insights on the network management simplification obtained with cluster migration. Statistical significance in the results are illustrated through 95% confidence intervals obtained from 20 simulation runs of varying seeds.



(A) without migration



(B) with migration

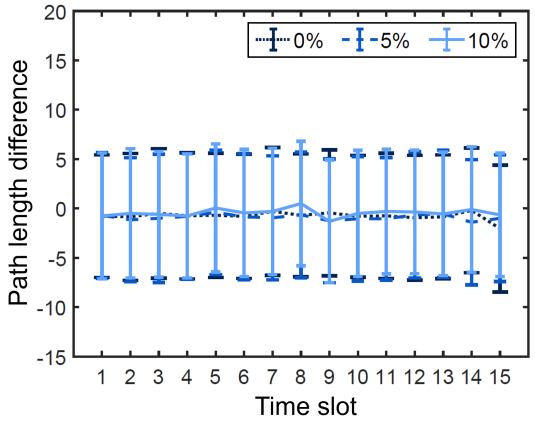
FIGURE 5.8: Required and actual path length differences with and without migrations for the static user case.

Static user case

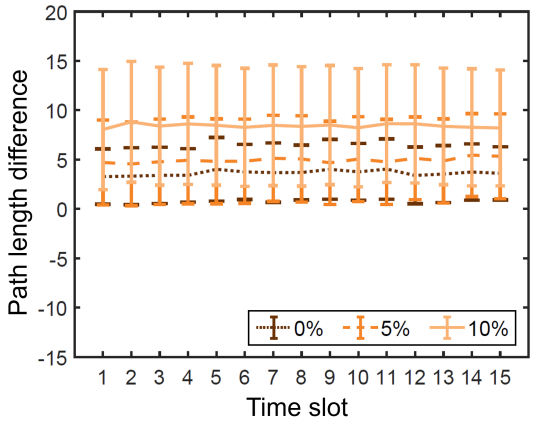
For the static user case, we compare the required (SLA) and actual path length differences when user u accesses his/her VO network via the node **ac**-x, $x = 1, \ldots, 17$, with and without migrations, supposing that the VOs are independent of one another.

Without migrations, clusters are randomly placed among: a) 2 **cl** nodes, b) 2 **cl** and 1 **ac** nodes, or c) 2 **cl** and 2 **ac** nodes, to simulate the traditional Cloud scenario and the Cloud-Fog-MEC interplay with 1 (e.g., at Home) or 2 (e.g., at Home and at Work) Fog nodes, respectively. Here, we suppose that **ac**-1 is at user u's Home, while **ac**-17 is at Work. Fig. 5.8a shows that the traditional Cloud case has generally better performance due to the VOs' central location. Path length improvements in the Cloud-Fog-MEC cases are only observed when u is at Home or at Work, with close proximity to VOs placed in node **ac**-1 or **ac**-17. In all three cases, some SLA violations are observed, as indicated by the negative path length differences.

Then, by introducing migrations, SLA specifications are always met, as shown in Fig. 5.8b. It can also be observed how the subscription-based parameter impacts the



(A) without migration



(B) with migration

FIGURE 5.9: Required and actual path length differences with and without migrations for the dynamic user case.

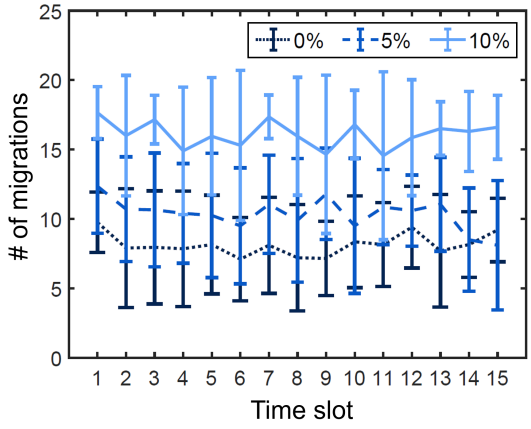
path length improvements. For instance, users with premium subscriptions can invoke $P_u = 1$ so that the network will consider an entire VO network as one cluster that follows its user according to the minimum proximity requirement.

Dynamic user case

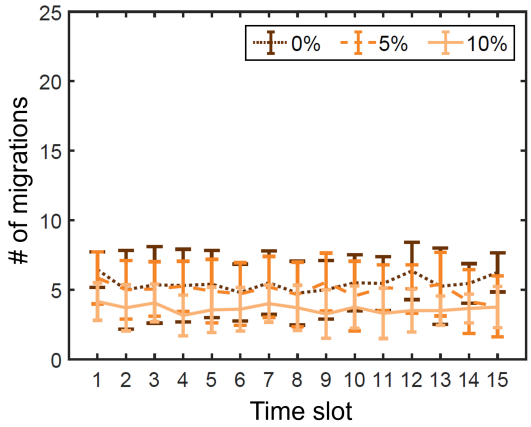
For the dynamic user case, we compare the required and actual path length differences as user u accesses his/her VO network via the node $\mathbf{ac}(t)$, at time instants $\{t = 1, \ldots, T\}$, with and without migrations. We suppose to have $T = 15$ time slots (e.g., considering 1-hr. granularity from 7:00 to 22:00), during which the user u's access point changed from $\mathbf{ac}(t-1)$ to $\mathbf{ac}(t)$.

At this point of the evaluation, we fix $P_u = 20$ to maximize the number of clusters and study the impact of inter-VO affinity on the path length improvements. Moreover, for tests without migrations, we only consider the traditional Cloud case since it generally gave better performance than the other cases, as previously seen.

Fig. 5.9a shows that the actual path lengths do not vary much with the inter-VO affinity since clusters are placed in either of the 2 **cl** nodes anyway, and as before, some



(A) VO migrations



(B) cluster migrations

FIGURE 5.10: Number of migrations in terms of VOs and clusters for the dynamic user case.

SLA violations are observed. The impact of inter-VO affinity on the path lengths is more evident when migrations are introduced, as illustrated in Fig. 5.9b. Note that since lesser number of clusters are generated with increasing percentage of VO pairs with affinity among them, cluster sizes will increase for a given number of VOs. This means that more VOs will be carried over by the same (possibly, tighter) proximity requirement, and hence, the greater path length improvements.

Additionally, in order to get a hint on how user mobility support can be simplified by considering VO clusters as aggregate entities during migrations, we also take a look at the number of migrations initiated in terms of VOs and clusters. Fig. 5.10 shows the number of migrations generated by the proposed approach, in terms of VOs and clusters, when supporting user mobility. When VOs are independent of one another, considering VO clusters as aggregate entities will initiate around 40% less migrations, on average, and such improvement increases with inter-VO affinity. For instance, when 10% of VO pairs have affinity among them, up to over 80% improvement is achieved.

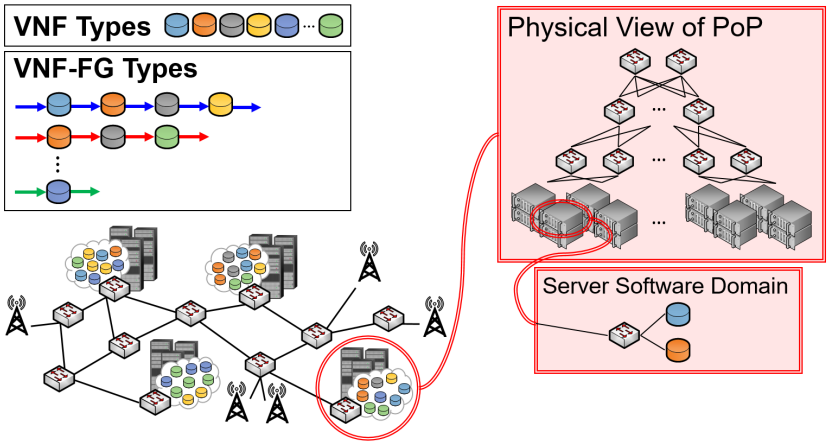


FIGURE 5.11: Network slice load balancing scenario.

5.3 Decentralized Load Balancing

Now, in a first attempt to jointly consider the infrastructure- and service-level aspects in the problem, this section addresses the load balancing among multiple network SCs (which represent network slice instantiations of an NSP referring to a specific vertical application) originating from different PoPs, based on a team-theoretic approach that takes into account the energy-aware LCPs of the IPr. In contrast to the more generalized frameworks of stochastic games and multi-objective optimization, this choice is mainly motivated by the fact that the actors of the "game" have a common goal (or cost function), and hence can act as a team, regardless of a decentralized knowledge on the network state and system parameters [142, 143].

Particularly, the actors of the team problem are *decision makers (DMs)* located at the PoPs of an NSP, through which user applications access a specific network service being offered via a VNF-FG. The NSP is in its turn a specific tenant of the IPr and runs the VNFs on VMs deployed on the IPr's hardware. The DMs are meant to balance the workload offered to their own PoPs among a number of VNF-FGs performing the required functionality; VNF-FGs are differentiated in terms of a global cost that accounts for both performance and energy consumption. For scalability, signaling reduction and fast reactivity reasons, we are interested in finding informationally decentralized (per-PoP) load balancing solutions in strategic form (i.e., mapping available instantaneous information into the required decisions, over the whole possible range of observable variables).

We suppose a given number of VNF-FGs to be active to provide the network functionalities requested by the different instantiations of the user's application service. Upon request of the network service pertaining to the slice, the PoP DM has the possibility to choose the chain of VNFs that are needed by the service, among a number of possible alternatives. An overview of the considered scenario is illustrated in Fig. 5.11. Different VNF-FGs may require common VNFs, and therefore they may share the computational capacity of the VMs performing them.

5.3.1 System Modeling

Each chain is characterized by an execution cost, which is the sum of the costs pertaining to each VNF composing it. The cost of a specific VNF is determined by the workload on the VM where it is executed and by the energy-aware LCP adopted by the IPr. We suppose each PoP DM to be aware of the LCPs and of its own workload (the rate of service requests), but not of the workloads generated at the other PoPs.

By adopting specific models for a server's core energy consumption [144] and for the power/delay cost of a server [57], we construct the cost associated to each VM as the product of the power consumption and the processing delay. Under a certain behavior of the IPr, this cost turns out to be quadratic in the total workload on the hardware hosting the VM; this fact renders the team problem mathematically equivalent to the one considered in [65] in a different setting and allows the application of the solution derived therein.

We define the optimization cost J associated to the VMs implementing the complete slice functionality as the sum of the products of processing delays D_j and of the power consumption Φ_j induced by every VM instantiated onto the NFVI:

$$J = \sum_{j=1}^{M} J_j = \sum_{j=1}^{M} D_j \Phi_j \tag{5.4}$$

where M is the number of VMs deployed by the tenant for the network slice.

Parallelizability and speed-up

Given the advanced support/capabilities of recent computing platforms and virtualization layer implementations, VMs running on a server can be considered as almost completely isolated among themselves, and — especially in high performance scenarios — exploiting different server internal components (e.g., cores, etc.). In more detail, considering the j-th VM in the NFV service, we assume that c_j vCPUs will be bound to c_j physical cores in the server, and made available by the hypervisor.

Even though the overall VM processing capacity scales linearly according to c_j, software applications hosted in the VM are well-known to exhibit a different performance behavior which depends on their parallelizability. In this respect, the well-known Amdahl's Law and its recent generalizations [57] suggest that the software-level performance depends on the number of available cores c_j and on their capacity μ_j by a speed-up factor S_j:

$$\mu_j(c_j) = S_j \mu_j = \frac{1}{\sum_{n=1}^{c_j} \frac{\beta_n^{(j)}}{n}} \mu_j \tag{5.5}$$

where $\beta_n^{(j)}$ is the n-th fractional component into which an algorithm implementing the j-th NFV service component can be split $\left(\sum_{n=1}^{c_j} \beta_n^{(j)} = 1\right)$. In general, $\beta_n^{(j)}$ is parallelizable on n cores; $\beta_1^{(j)}$ represents the fraction that is not parallelizable and, without loss

of generality, we number the cores in ascending order of utilization.

Power- and performance-aware cost function

Now, we introduce a cost function capable of capturing both power and performance aspects of VM j.

Firstly, in a very simple formulation that considers the aggregate action of the speed-up introduced by parallelization, the delay term can be taken as that of a M/M/1 queueing system, which is given by

$$D_j(c_j) = \frac{1}{S_j \mu_j - f_j} \tag{5.6}$$

indicating with f_j the total load on VM j. Despite the possible inaccuracy of the M/M/1 model, its simple expression for $D_j(c_j)$ results to be an effective penalty function with respect to the saturation of the processor's capacity, which is an essential characteristic to be reflected in our optimization problem.

Regarding the power consumption, we start from the model considered in [144], which takes into account the presence of both AR and LPI effects in the power profile of a core; namely, if δ_{jn} is the load fraction processed on its assigned core n, the power consumption Φ_{jn} is given by

$$\Phi_{jn} = K_j \mu_j^3 \left(\frac{\delta_{jn} f_j}{\mu_j} \right)^{1/\alpha_j} \tag{5.7}$$

where $K_j \in \mathbb{R}^+$ and $\mathbb{R}^+ \ni \alpha_j \geq 1$ are parameters depending on the hardware type (we denote by $\mathbb{R}^+$ the set of positive real numbers). Given our ordering, δ_{jn} turns out to be

$$\delta_{jn} = \sum_{s=n}^{c_j} \frac{\beta_s}{s} \tag{5.8}$$

Thus, the power consumption induced by VM j on the c_j cores can be expressed as follows.

$$\Phi_j = \sum_{n=1}^{c_j} \Phi_{jn} = \sum_{n=1}^{c_j} K_j \mu_j^3 \left(\frac{\delta_{jn} f_j}{\mu_j} \right)^{1/\alpha_j} \tag{5.9}$$

Finally, the cost J_j is associated to each VM j, which is expressed as the product of power consumption and processing delay

$$\begin{aligned} J_j = \Phi_j D_j &= \left[\sum_{n=1}^{c_j} K_j \mu_j^3 \left(\frac{\delta_{jn} f_j}{\mu_j}\right)^{1/\alpha_j}\right] \frac{1}{S_j \mu_j - f_j} \\ &= \frac{K_j \mu_j^{3-1/\alpha_j} f_j^{1/\alpha_j}}{S_j \mu_j - f_j} \sum_{n=1}^{c_j} (\delta_{jn})^{1/\alpha_j} \\ &= K'_j \frac{\mu_j^{3-1/\alpha_j} f_j^{1/\alpha_j}}{S_j \mu_j - f_j} \end{aligned} \tag{5.10}$$

where all multiplicative coefficients related to VM j are collected in the term K'_j, which, given the characteristics of the specific network application and of the hardware, is a known constant.

By minimizing J_j with respect to μ_j we obtain

$$\mu_j^* = \underset{\mu_j}{argmin} J_j = \frac{3\alpha_j - 1}{S_j(2\alpha_j - 1)} f_j = \theta_j f_j \tag{5.11}$$

$$J_j^* = \underset{\mu_j}{min} J_j = K'_j \frac{\theta_j^{3-1/\alpha_j}}{S_j \theta_j - 1} f_j^2 = h_j f_j^2 \tag{5.12}$$

where $\theta_j = (3\alpha_j - 1)/S_j(2\alpha_j - 1)$ and $h_j = K'_j\left(\theta_j^{3-1/\alpha_j}\right)/(S_j\theta_j - 1)$. Thus, the application of a simple proportional control law like Eq. (5.11) on the part of the IPr has the effect of making the cost associated to the VM using its core(s) quadratic in the total load. As noted in [72], we are considering a continuous solution to the server operating capacity adjustment. In practice, the physical resources allow a discrete set of working frequencies, with corresponding processing capacities; this would also ensure that the processing speed does not decrease below a lower threshold, avoiding excessive delay in the case of low load.

5.3.2 Problem Statement and Solution

We want to determine the decentralized control policy that maps the workloads at the PoPs to the respective shares assigned to each of the VNF-FGs referring to a specific network service. The problem is posed in a team-theoretic setting [142, 143, 145], as all PoP DMs of the NSP have the same goal, which consists of the minimization of a common overall cost for the usage of the resources, but they possess different information on their respective incoming flows.

Generally, the total flow offered to the j-th VM implementing a specific VNF is composed of a number of contributions pertaining to the VNF-FGs that use its functionality. Let S be the total number of PoPs, F the total number of VNF-FGs, and $\mathcal{F}_i$, with $|\mathcal{F}_i| = \mathcal{F}_i \leq F, i = 1, \ldots, S$, the subset of VNF-FGs used by the i-th PoP. We indicate

by $u_{ki}, k \in \mathcal{F}_i, i = 1, \ldots, S$, the fraction of the workload r^i generated at PoP i that is offered to VNF-FG k. Moreover, let $\underline{r} = [r^1, \ldots, r^S]^T$ (the superscript T indicates transpose) be the vector collecting all PoP workloads, and $\mathcal{V}_j$ the set of VNF-FGs that use the services of VM j. We assume the components of $\underline{r}$ to be independent non-negative continuous random variables with a given probability distribution. The team optimization problem can then be stated as

$$\min_{\gamma_{ki}(\cdot),\, k\in\mathcal{F}_i,\, i=1,\ldots,S} \bar{J} \tag{5.13}$$

where

$$\bar{J} = \frac{1}{2}\underset{\underline{r}}{E}\left\{\sum_{j=1}^{M} w_j \left(\sum_{i=1}^{S}\sum_{k\in\mathcal{F}_i \bigcap \mathcal{V}_j} u_{ki}\right)^2\right\} \tag{5.14}$$

with

$$u_{ki} = \gamma_{ki}(r^i),\ k \in \mathcal{F}_i,\ i = 1, \ldots, S \tag{5.15}$$

under the constraints

$$\sum_{k\in\mathcal{F}_i} u_{ki} = r^i,\ i = 1, \ldots, S \tag{5.16}$$

$$u_{ki} \geq 0,\ k \in \mathcal{F}_i,\ i = 1, \ldots, S \tag{5.17}$$

The weighting coefficient w_j in Eq. (5.14) accounts for both the value of h_j (stemming from the capabilities of the underlying hardware, and the parallelizability of the VNF code) and the influence of the network topology on the contributing flows that enter the VM (as they may traverse different network paths to reach it from the previous VMs in their chains). Eqs. (5.15) entail a decentralization constraint – i.e., DM i decides on the shares of its workload among the VNF-FGs only on the basis of the knowledge of its own workload, and not those of the others. The only centralized information is constituted by the a priori knowledge on the number of VMs and of DMs, the topology and the probability distributions of the inputs.

Following [65], we consider finding *person-by-person optimal (p.b.p.o.)* strategies of the form Eqs. (5.15) for the above problem. By defining $\gamma^i(\cdot) = \{\gamma_{ki}(\cdot), k \in \mathcal{F}_i\}$, $i = 1, \ldots, S$, the p.b.p.o. strategy of DM i entails the minimization of the cost in Eq. (5.14), under fixed (functional) values of the strategies of the other agents $\gamma^{-i}(\cdot) = \{\gamma_{kj}(\cdot),$

$k \in \mathcal{F}_j$, $j = 1, \ldots, S$, $j \neq i\}$; namely, we are looking for functions $\gamma^{i^*}(\cdot)$ such that

$$\begin{aligned}
\gamma^{i^*} &= \underset{\gamma^i(\cdot)}{argmin}\ \bar{J}\left[\gamma^i(\cdot), \gamma^{i^*}(\cdot)\right] \\
&= \underset{\gamma^i(\cdot)}{argmin}\ \frac{1}{2} \underset{\underline{r}}{E}\left\{\sum_{j=1}^{M} w_j \left(\sum_{k \in \mathcal{F}_i \bigcap \mathcal{V}_j} \gamma_{ki}(r^i) + \sum_{\substack{\ell=1 \\ \ell \neq i}}^{S} \sum_{k \in \mathcal{F}_\ell \bigcap \mathcal{V}_j} \gamma^*_{k\ell}(r^\ell) \right)^2 \right\} \\
&= \underset{u^i}{argmin}\ \frac{1}{2} \underset{r^\ell, \ell=1,\ldots,S, \ell \neq i}{E} \left\{ \sum_{j=1}^{M} w_j \left(\sum_{k \in \mathcal{F}_i \bigcap \mathcal{V}_j} u_{ki} + \sum_{\substack{\ell=1 \\ \ell \neq i}}^{S} \sum_{k \in \mathcal{F}_\ell \bigcap \mathcal{V}_j} \gamma^*_{k\ell}(r^\ell) \right)^2 \middle| r^i \right\}, \\
& \quad i = 1, \ldots, S, \ \forall r^i
\end{aligned} \tag{5.18}$$

where $u^i = [u_{\ell i}, \ldots, u_{\mathcal{F}_i i}]^T$. Conditioning the expectation in Eq. (5.18) transforms the functional optimization problem of DM i into an ordinary minimization, which can be handled by the application of Karush-Kuhn-Tucker conditions. However, the solution for DM i depends on the average of the strategies of the other DMs, as can be seen by the coefficient of the linear term that arises in Eq. (5.18) by expanding the square.

Let the quantities $(\eta_i^m)^*$, $m = 1, \ldots, M$, $i = 1, \ldots, S$ be a set of parameters that characterize the strategies. By indicating the parametrized p.b.p.o. strategies as $\hat{\gamma}^{i^*}(r^i, \{(\eta_j^n)^*, n = 1, \ldots, M, j = 1, \ldots, S, j \neq i\})$, the unknown quantities $(\eta_i^m)^*$ can be found by solving a set of non-linear fixed-point equations of the form

$$\begin{aligned}
(\eta_i^m)^* = \sum_{\substack{\ell=1 \\ \ell \neq i}}^{S} \sum_{k \in \mathcal{F}_\ell \bigcap \mathcal{V}_j} \underset{r^\ell}{E}\left\{ \hat{\gamma}^*_{k\ell}\left(r^\ell, \left\{(\eta_j^n)^*,\ n = 1, \ldots, M,\ j = 1, \ldots, S,\ j \neq \ell\right\}\right)\right\}, \\
m = 1, \ldots, M,\ i = 1, \ldots, S
\end{aligned} \tag{5.19}$$

The procedure for finding p.b.p.o. strategies for the problem outlined by Eqs. (5.13)-(5.17) can then be split into two parts:

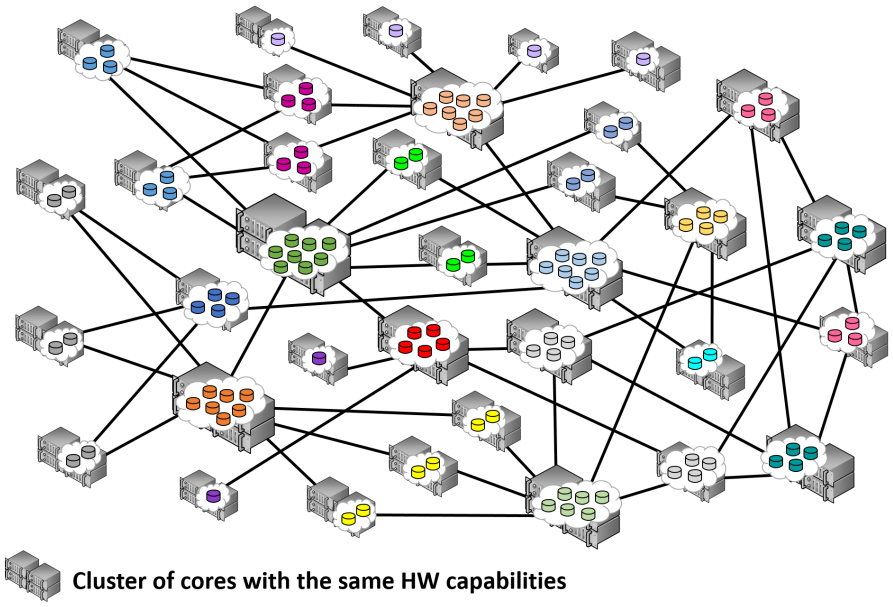
- having fixed a set of parameters $(\eta_i^m)^*$, $m = 1, \ldots, M$, $i = 1, \ldots, S$, we first derive the analytical expression of strategies Eq. (5.18), and then;
- we seek a numerical solution to the fixed-point equations Eq. (5.19).

Since the mathematical problem outlined here is equivalent to the one that we solved in [65], we do not repeat the derivation. Note that the team solutions we can find by applying this methodology would coincide with a unique team-optimal solution only in case of existence of a unique fixed point for equations Eq. (5.19), whose investigation, however, is beyond the scope of this work. Optimality of the strategies we have derived is therefore assured only in p.b.p.o. sense, which also corresponds with the concept of *Nash equilibrium* [142, 143].

Decision Makers	VNF-FG Types	Available VNF-FGs
DM_1, DM_{11}		DM_1: 4, DM_{11}: 2
DM_2, DM_{12}		DM_2: 3, DM_{12}: 2
DM_3, DM_{13}		DM_3: 3, DM_{13}: 2
DM_4, DM_{14}		DM_4: 4, DM_{14}: 2
DM_5, DM_{16}		DM_5: 4, DM_{16}: 4
DM_6		DM_6: 4
DM_7, DM_{19}, DM_{20}		DM_7: 4, DM_{19}: 2, DM_{20}: 2
DM_8, DM_{17}, DM_{18}		DM_8: 4, DM_{17}: 2, DM_{18}: 2
DM_9		DM_9: 3
DM_{10}, DM_{15}		DM_{10}: 4, DM_{15}: 2

(A) NFV service specifications

VNF Types	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
# of Instances	6	7	6	4	8	2	5	6	4	6	7	7	8	4	6	6	4	2	4	8



(B) NFV network

FIGURE 5.12: Scenario with 20 DMs, each one accessing an NFV service through a given number of available VNF-FGs over the shared NFV network.

But, since every team-optimal solution is necessarily a pbp optimal solution, the latter plays an important role in the derivation of the former,

5.3.3 Performance Evaluation

To evaluate the performance of the proposed load balancing approach, we consider the scenario in Fig. 5.12, and find the p.b.p.o. strategies of the DMs by using the numerical method in [65]. Then, we compare the normalized dynamic power consumption induced by the VNF-FGs with the team-optimal and uniform flow distribution, as well as the one that corresponds to concentrating each DM's load on their respective least-cost paths.

In more detail, we evaluate a system with 20 DMs, each one accessing a specific NFV service over a shared network of 35 resource clusters, hosting 84 VNF instances of 20 VNF types, as illustrated in Fig. 5.12. Generally, instances of the same VNF type can be hosted among resources with the same or varying capabilities — this is captured in the considered scenario through clustering of resources and various combinations of hardware parameters. Moreover, some DMs accessing the same NFV service can have different number of available VNF-FGs (i.e., paths), simulating a more general scenario with differentiated services.

The (physical/logical) links served by the VMs $\{j\}$ in the system are identified with an index j, and a weight coefficient w_j that depends on a number of components. Particularly, w_j is given by the sum of the coefficient h_j of the serving VM, and of a random number generated from the continuous uniform distribution, $\boldsymbol{U}(0, 10)$. The former is determined by the underlying hardware (i.e., K_j and α_j parameters), and the degree of parallelizability of the VNF code (i.e., c_j and $\{\beta_n^{(j)}\}$ information), while the latter accounts for the different network paths traversed by the contributing flows that enter the VM, as noted in Sub-section 5.3.2; here we decided for a random choice, to avoid being bound to a fixed network topology.

Following [72], in which $K_j \in \boldsymbol{U}_K(1, 10)$ and $\alpha_j \in \boldsymbol{U}_\alpha(2, 3)$, we consider two cases in this work:

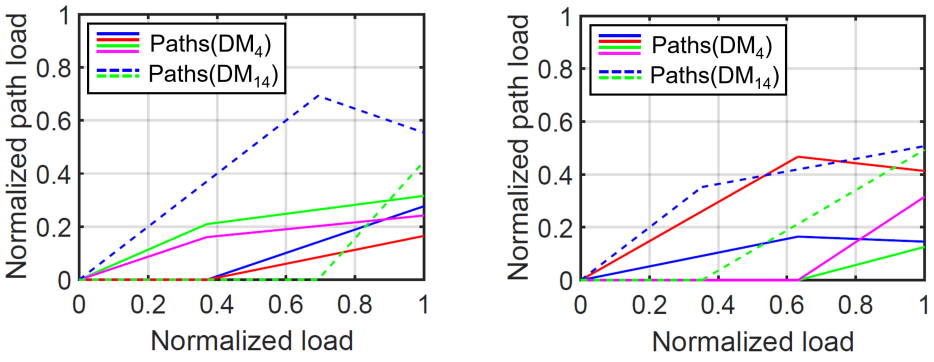
- *homogeneous hardware*, where $K_j = K \in \boldsymbol{U}_K$, $\alpha_j = \alpha \in \boldsymbol{U}_\alpha$, $\forall j$, and;
- *heterogeneous hardware*, where the parameters among VMs $\{j\}$ hosted on the resource cluster σ are generated as $K_j = K_\sigma \in \boldsymbol{U}_K$, $\alpha_j = \alpha_\sigma \in \boldsymbol{U}_\alpha$, $\forall j \mapsto \sigma$.

The latter is supposed to cover not only the possible hardware heterogeneity inside a PoP (e.g., server level), but also the scenario where a VNF-FG spans multiple PoPs of varying capabilities.

The parallelizability of each VNF instance in the system (even the ones performing the same functionality) is generated randomly, supposing that a VNF code is parallelizable into 1, 2, 4, 6 or 8 vCPUs. Random permutation is used to generate varying $\{\beta_n^{(j)}\}$ values, even with the same c_j; this emulates the performance variations of a code on different execution environments. For the sake of simplicity, but without loss of generality, we suppose that the 20 DMs have the same maximum load R_{max}, and their instantaneous loads $r(\mathbf{DM}_i)$, $i = 1, \ldots, 20$, are uniformly distributed in $[0, R_{max}]$.

Team-optimal load balancing

In both homogeneous and heterogeneous hardware cases, the resulting p.b.p.o. load distribution policies of the DMs highly depend on w_j and the number of VNF-FGs sharing a VM. Some DMs have relatively "static" strategies, allocating a constant fraction of their load to (a subset of) the available VNF-FGs, while others result in more interesting strategies, adapting their distributions with the load. Fig. 5.13 shows examples of p.b.p.o. load distribution policies of $\mathbf{DM}_4$ and $\mathbf{DM}_{14}$, where paths indicated with the



(A) homogeneous hardware (B) heterogeneous hardware

FIGURE 5.13: P.b.p.o. load distribution policies of DM_4 and DM_{14}.

same color correspond to the same VNF-FGs. It can be observed that the general form of the team-optimal solutions is piecewise-linear.

Dynamic power consumption

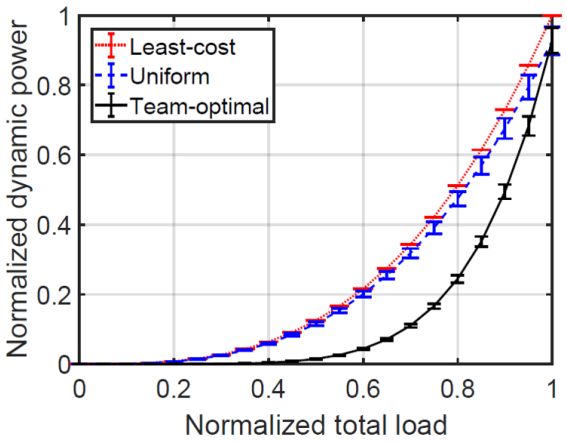
Then, we evaluate the normalized dynamic power consumption induced by the 20 DMs when the team solutions are applied in the homogeneous and heterogeneous hardware cases. As comparison, the following two baseline policies are considered:

- *least-cost path*, in which the DMs route all the load to the VNF-FG with the minimum execution cost (i.e., sum of the link weights) among the available paths, and;
- *uniform flow distribution*, in which all available paths are allocated equal fractions of the load.

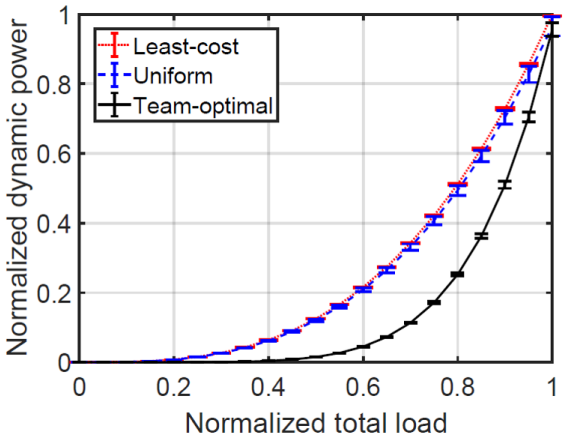
To add statistical significance in the results, 95% confidence intervals are obtained from 10 runs of varying seeds.

As shown in Fig. 5.14, the p.b.p.o. team solutions gave better performance in terms of energy saving with respect to the two baselines, achieving improvements of up to over 45% to around 4 orders of magnitude when the normalized total load is less than 80%. While similar behaviors can be observed with both homogeneous and heterogeneous hardware, with the latter there are some cases in which uniform load distribution result in the highest consumption, rather than the one using only the least-cost paths (which is always the case for the former). This can be expected especially when the costs of a DM's available VNF-FGs vary greatly.

Though in a relatively simple topology of VNF-FGs, the results highlight some characteristics that can be expected by the application of p.b.p.o. team strategies in the NFV environment we have considered. In particular, in the presence of different types of hardware (i.e., heterogeneous hardware case) and multiple interactions among DMs' paths, we expect higher energy saving gains.



(A) homogeneous hardware



(B) heterogeneous hardware

FIGURE 5.14: Normalized dynamic power consumption induced by the VNF-FGs when the least-cost, uniform and p.b.p.o. load distribution policies are applied.

5.4 Summary

In this chapter, approaches for: a) dynamic VNF consolidation on ACPI-enabled resources, b) service migration according to user mobility, and c) decentralized load balancing among multiple service instantiations, are collectively proposed for network/services management and control in softwarized 5G environments, by building on the mechanisms presented in Chapters 3 and 4, and on the lessons learned in their respect.

At the infrastructure-level perspective, the proposed power- and performance-aware VNF consolidation relies on VNF workload profiles and a power model that captures ACPI configurations, to classify core workloads (i.e., the aggregate workload of the VNFs sharing each core) according to the most suitable ACPI configuration. Note that the classification can be done offline for a whole range of workload profiles and specific CPU architecture; then, the joint power scaling and in-server consolidation of VNFs can be performed according to actual workload variations. Numerical results show that the average datacenter power consumption can be reduced by up to 10% with respect to

the considered baselines, suggesting a considerable amount of annual savings. For better controlled VNF performances, the latency aspect is yet to be incorporated in the consolidation decision.

On the other hand, at the service-level perspective, the proposed user-centric service migration relies on MCO's center migration approach for scalably supporting wide-area bulk migrations, with the user's VO network organized into clusters according to proximity and affinity requirements, as well as a subscription-based parameter (for service differentiation among users). Numerical results show that introducing migrations improves the QoS to always meet or exceed the SLA requirements. Moreover, considering VO clusters as aggregate entities will initiate around 40% less migrations, on average – an improvement that increases with inter-VO affinity and could potentially simplify network/services management when supporting user mobility. Moving forward, the policy can be extended to cover multiple affinity levels, as well as add constraints on the available capacities among datacenters.

Finally, in an attempt to jointly look into the infrastructure- and service-level perspectives of network/services management and control, the proposed decentralized load balancing among multiple service instances (implemented as VNF-FGs that are differentiated in terms of a global, power- and performance-aware cost) relies on simple, state-of-the-art power and latency models to find parametrized p.b.p.o. team strategies, which are piecewise linear in the workload of each specific DM. As such aggregated workloads can vary dynamically over relatively short time scales (e.g., in the order of a few seconds, depending on end users' density and mobility), informationally decentralized strategies lend themselves to fast reaction without the need of additional signaling. Numerical results show that the approach achieves up to over 45% reduction in the dynamic power consumption with respect to the considered baselines.

Chapter 6

Conclusion

The upcoming 5G is foreseen to play a fundamental role in our socio-economic growth by supporting various and radically new vertical applications, as a one-fits-all technology that is enabled by emerging softwarization solutions – specifically, the Fog, MEC, NFV and SDN paradigms.

In more detail, the introduction of Fog computing and MEC brings Cloud-like services much closer to end-users and network-connected things, which can be exploited to improve both network- and service-level performances. NFV is similarly built on IT virtualization, in which specific service components (i.e., VNFs and VNFCs) are implemented in software with high customizability and improved time to market, while SDN deals with the chaining among them (i.e., VNF-FGs) and/or with physical service components. Moreover, the latter is foreseen to provide the network programmability levels required to enable network/service "agility". With the upcoming Cloud-Fog-MEC interplay, NFV and SDN, a unified, multi-domain and multi-tenant 5G infrastructure, on which highly flexible and programmable network services and applications can be deployed, is expected to emerge.

Despite the notable potential in such softwarized environments, there are still a number of open issues at the infrastructure and service levels that need to be addressed to guarantee the smooth rollout of 5G. In the context of network/services management and control, a review of related literature is presented to provide an overview of the technological scenario, on which we have built the considered problems and the contributions proposed in their respect.

Particularly, this thesis seeks to enable scalable and sustainable softwarized 5G environments by:

- optimizing the trade-off between power and performance in the underlying COTS hardware according to their corresponding workload dynamics;
- realizing the connectivity among (possibly, geo-distributed) network service components, and effectively support its dynamic reconfiguration; and
- steering traffic flows according to the current (re)configuration of their corresponding services in a seamless fashion,

with low computational and network overhead, as well as low technological requirements. The specific contributions are organized in the following three research axes,

with the first two covering the main contributions, and the third demonstrating their possible applications to network/services management and control, through some spin-off works.

Infrastructure Modeling and Analytics: In this research axis, we seek to effectively and efficiently handle the operational issues stemming from the underlying COTS hardware and virtualization solutions adopted in NFVIs, taking into account the growing need for automated management. Particularly, a real-time analytics approach based on the $M^X/G/1/SET$ queueing model is proposed for VNF workload profiling and estimation of network KPIs (specifically, power and latency) using – and adding value to – available hardware/software PMCs.

Experimental results show good estimation accuracies for both VNF workload profiling and network KPI estimation, with respect to the input traffic and actual measurements, respectively, demonstrating how the approach can be a powerful tool in augmenting the capabilities of a NFVI's VIM, as well as in the development of next-generation resource/service provisioning solutions, among others.

Network Slicing and Mobility Management: In this research axis, we seek to exploit SDN's potential for enabling network/service "agility", while taking into account the finite-sized rule tables in SDN devices. Particularly, a novel multi-clustering approach, called MCO, is proposed to realize geo-distributed VTNs and effectively support dense deployments of mobile VOs at the network edge, as well as seamless user/services mobility through bulk inter-datacenter VO live migrations.

Numerical results show that MCO achieves up to over one order of magnitude smaller number of OF rules in the VTN implementation and rule updates during center migrations, demonstrating its high scalability with respect to state-of-the-art SDN mechanisms. While possible path sub-optimality may occur, knowledge on the datacenter topology can be exploited for VO placement optimization and load balancing. Furthermore, in the context of the INPUT use case (i.e., PNaaS), experimental results demonstrate MCO's low computational complexity in terms of rule and rule update calculation times during SC instantiation and wide-area bulk migrations, respectively.

Network/Services Management and Control: In this research axis, we seek to demonstrate some directions towards ultimately achieving the end goal of the thesis, by building on the model-based analytics and MCO approaches developed in the previous research axes, and on the lessons learned therein. Particularly, approaches for: a) dynamic VNF consolidation on ACPI-enabled resources, b) service migration according to user mobility, and c) decentralized load balancing among multiple service instantiations, are collectively proposed as network/services management and control applications in softwarized 5G environments.

The first item looks into the infrastructure-level perspective with a power- and performance-aware VNF consolidation approach that can be performed according to actual workload variations, while the second looks into the service-level perspective with

user-centric service migrations that takes into account the proximity and affinity requirements of VO networks, as well as the (differentiated) service subscriptions of the users. Finally, the third item attempts to jointly look into the infrastructure- and service-level perspectives of network/services management and control with a team-theoretic approach for the decentralized load balancing among multiple service instantiations, taking into account a power- and performance-aware cost function.

The numerical results obtained in each of these works show substantial improvements with respect to the considered baselines, which further manifest the potentials of the main contributions of this thesis towards enabling scalable and sustainable softwarized 5G environments.

Bibliography

[1] Ericsson Mobility Rep. June 2018. URL: https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf.

[2] *The Zettabyte Era: Trends and Analysis*. Cisco White Paper. June 2017. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf.

[3] C. Tao et al. *Vision on Software Networks and 5G*. 5GPPP SN WG White Paper. Jan. 2017. URL: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP_SoftNets_WG_whitepaper_v20.pdf.

[4] *Towards 5G*. 2018. URL: https://ec.europa.eu/digital-single-market/en/towards-5g.

[5] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. Cisco White Paper. 2015. URL: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[6] S. Kekki et al. *MEC in 5G networks*. ETSI White Paper. June 2018. URL: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf.

[7] M. Taneja and A. Davy. "Resource Aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm". In: *Proc. 2017 IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*. Lisbon, Portugal, May 2017, pp. 1222–1228.

[8] A. Yousefpour, G. Ishigaki, and J. P. Jue. "Fog Computing: Towards Minimizing Delay in the Internet of Things". In: *Proc. 2017 IEEE Int. Conf. Edge Comput. (EDGE)*. Honolulu, HI, June 2017, pp. 17–24.

[9] M. Chiosi et al. *Network Functions Virtualization: An Introduction, Benefits, Enablers, Challenges & Call For Action*. ETSI White Paper. 2012. URL: http://portal.etsi.org/NFV/NFV_White_Paper.pdf.

[10] D. Kreutz et al. "Software-Defined Networking: A Comprehensive Survey". In: *Proc. IEEE* 103.1 (Jan. 2015), pp. 14–76.

[11] *OpenFlow Switch Specifications Version 1.3.1 (Wire Protocol (0x04)*. Sept. 2012. URL: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-specv1.3.1.pdf.

[12] F. Leitão, R. David Carnero Ros, and J. Rius i Riu. "Fixed-Mobile Convergence Towards the 5G Era: Convergence 2.0: The Past, Present and Future of FMC Standardization". In: *Proc. 2016 IEEE Conf. Std. Commun. Netw. (CSCN)*. DOI: 10.1109/CSCN.2016.7785155. Berlin, Germany, Oct. 2016.

[13] A. Manzalini et al. *Towards 5G Software-Defined Ecosystems: Technical Challenges, Business Sustainability and Policy Issues*. IEEE SDN Initiative Whitepaper. 2016. URL: https://resourcecenter.fd.ieee.org/white-papers/sdn/FDSDNWP0002.html.

[14] *View on 5G Architecture, Version 2.0*. 5GPPP AWG Whitepaper. 2017. URL: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf.

[15] J. Ordonez-Lucena et al. "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges". In: *IEEE Commun. Mag.* 55.5 (May 2017), pp. 80–87.

[16] S. Thalanany and P. Hedman. *Description of Network Slicing Concept*. NGMN Alliance Deliverable. 2016. URL: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf.

[17] *Study on Management and Orchestration of Network Slicing for Next Generation Network*. TR 28.801, 3GPP TSG Tech. Rep. 2018. URL: https://www.ngmn.org/fileadmin/user_upload/161010_NGMN_Network_Slicing_framework_v1.0.8.pdf.

[18] *Advanced Configuration and Power Interface Specification*. URL: http://www.acpi.info/DOWNLOADS/ACPIspec40a.pdf.

[19] R. Bolla et al. "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures". In: *IEEE Commun. Surveys Tuts.* 13.15 (2011), pp. 223–244.

[20] E. Hernandez-Valencia, S. Izzo, and B. Polonsky. "How Will NFV/SDN Transform Service Provider OPEX?" In: *IEEE Netw.* 29.3 (May 2015), pp. 60–67.

[21] R. Bolla et al. "The Dark Side of Network Functions Virtualization: A Perspective on the Technological Sustainability". In: *Proc. 2017 IEEE Int. Conf. Commun. (ICC)*. Paris, France, May 2017. DOI: 10.1109/ICC.2017.7997129.

[22] *Network Functions Virtualisation (NFV); Management and Orchestration*. ETSI NFV ISG Spec. 2014. URL: http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf.

[23] G. Choudhury. "An $M^X/G/1$ Queueing System with a Setup Period and a Vacation Period". In: *Queueing Syst.* 36.1-3 (2000), pp. 23–38.

[24] B. Kim and B. Lee. "Integrated Management System for Distributed Micro-Datacenters". In: *Proc. 18th Int. Conf. Adv. Commun. Technol. (ICACT)*. PyeongChang, Korea, Jan. 2016, pp. 466–469.

[25] M. Yannuzzi et al. "A New Era for Cities with Fog Computing". In: *IEEE Internet Comput.* 21.2 (Mar. 2017), pp. 54–67.

[26] D. Sabella et al. "Mobile-Edge Computing Architecture: The role of MEC in the Internet of Things". In: *IEEE Consum. Electron. Mag.* 5.4 (Oct. 2016), pp. 84–91.

[27] D. Taibi, V. Lenarduzzi, and C. Pahl. "Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation". In: *IEEE Cloud Comput.* 4.5 (Sept. 2017), pp. 22–32.

[28] *LTE; Telecommunication Management; Life Cycle Management (LCM) for Mobile Networks that Include Virtualized Network Functions; Requirements*. ETSI NFV Tech. Spec. July 2017. URL: http://www.etsi.org/deliver/etsi_ts/128500_128599/128525/14.01.00_60-/ts_128525v140100p.pdf.

[29] M. Nitti et al. "The Virtual Object as a Major Element of the Internet of Things: A Survey". In: *IEEE Commun. Surveys Tuts.* 18.2 (2016), pp. 1228–1240.

[30] R. Kawashima and H. Matsuo. "Non-Tunneling Edge-Overlay Model using OpenFlow for Cloud Datacenter Networks". In: *Proc. 5th IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*. Bristol, UK, Dec. 2013, pp. 176–181.

[31] S. Hanks et al. *Generic Routing Encapsulation (GRE)*. IETF RFC 1701. Oct. 1994. URL: https://tools.ietf.org/html/rfc1701.

[32] M. Mahalingam et al. *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks*. IETF RFC 7348. Aug. 2014. URL: https://tools.ietf.org/html/rfc7348.

[33] *Network Functions Virtualisation (NFV); Architectural Framework*. ETSI NFV ISG Spec. 2013. URL: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf.

[34] *Setting the Scene for 5G: Opportunities & Challenges*. ITU Rep. 2018. URL: https://www.itu.int/en/ITU-D/Documents/ITU_5G_REPORT-2018.pdf.

[35] *IMT Vision — Framework and Overall Objectives of the Future Development of IMT for 2020 and Beyond*. ITU-R Rec. M.2083-0. Sept. 2015. URL: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf.

[36] *Virtualization Overview*. VMware White Paper. 2006. URL: https://www.vmware.com/pdf/virtualization.pdf.

[37] *Network Functions Virtualisation (NFV) Release 3; Virtualised Network Function; Specification of the Classification of Cloud Native VNF Implementations*. ETSI NFV ISG Spec. 2018. URL: https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/011/03.01.01_60/gs_NFV-EVE011v030101p.pdf.

[38] *Network Functions Virtualisation (NFV); Virtualisation Technologies; Report on the application of Different Virtualisation Technologies in the NFV Framework*. ETSI NFV ISG Spec. 2016. URL: https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/004/01.01.01_60/gs_nfv-eve004v010101p.pdf.

[39] T. Barbette, C. Soldani, and L. Mathy. "Fast Userspace Packet Processing". In: *Proc. 11th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*. Oakland, CA, May 2015, pp. 5–16.

[40] P. Kutch and B. Johnson. *SR-IOV for NFV Solutions: Practical Considerations and Thoughts*. Feb. 2017. URL: https://www.intel.com/content/dam/www/public/us/en/documents/technology-briefs/sr-iov-nfv-tech-brief.pdf.

[41] M. Kourtis et al. "Enhancing VNF Performance by Exploiting SR-IOV and DPDK Packet Processing Acceleration". In: *Proc. 2015 IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*. San Francisco, CA, Nov. 2015, pp. 49–59.

[42] J. Li et al. "When I/O Interrupt Becomes System Bottleneck: Efficiency and Scalability Enhancement for SR-IOV Network Virtualization". In: *IEEE Trans. Cloud Comput.* (Dec. 2018). DOI: 10.1109/TCC.2017.2712686.

[43] P. Li et al. "Designing Virtual Network Functions for 100 GbE Network Using Multicore Processors". In: *Proc. 2017 ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*. Beijing, China, May 2017, pp. 49–59.

[44] M. Rifai et al. "Minnie: An SDN World With Few Compressed Forwarding Rules". In: *Comput. Netw.* 121 (2017), pp. 185–207.

[45] A. Schwabe and K. Holger. "Using MAC Addresses as Efficient Routing Labels in Data Centers". In: *Proc. 3rd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*. Chicago, IL, Aug. 2014, pp. 115–120.

[46] K. Agarwal et al. "Shadow MACs: Scalable Label-switching for Commodity Ethernet". In: *Proc. 3rd Workshop Hot Topics Soft. Defined Netw. (HotSDN)*. Chicago, IL, Aug. 2014, pp. 157–162.

[47] A. Hari, T. V. Lakshman, and G. Wilfong. "Path Switching: Reduced-State Flow Handling in SDN Using Path Information". In: *Proc. 11th ACM Int. Conf. Emerg. Netw. Experiments Technol. (CoNEXT)*. Heidelberg, Germany, Dec. 2015.

[48] B. Leng et al. "FTRS: A Mechanism For Reducing Flow Table Entries in Software Defined Networks". In: *Comput. Netw.* 122 (2017), pp. 1–15.

[49] S. Hu et al. "Explicit Path Control in Commodity Data Centers: Design and Applications". In: *IEEE/ACM Trans. Netw.* 24.5 (Oct. 2016), pp. 2768–2781.

[50] H. Wang et al. "Virtual Machine Migration Planning in Software-Defined Networks". In: *IEEE Trans. Cloud Comput.* (2017). DOI: 10.1109/TCC.2017.2710193.

[51] C. Clark et al. “Live Migration of Virtual Machines”. In: *Proc. 2nd USENIX Symp. Netw. Sys. Design Impl. (NSDI)*. Vol. 2. Boston, MA, May 2005, pp. 273–286.

[52] R. Niranjan Mysore et al. “PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric”. In: *ACM SIGCOMM Comput. Commun. Rev.* 39.4 (Oct. 2009), pp. 39–50.

[53] R. Bolla et al. “Green Networking with Packet Processing Engines: Modeling and Optimization”. In: *IEEE/ACM Trans. Netw.* 22.1 (Feb. 2014), pp. 110–123.

[54] R. Bolla et al. “Corrections to: "Green Networking With Packet Processing Engines: Modeling and Optimization"”. In: *IEEE/ACM Trans. Netw.* (2018). DOI: 10.1109/TNET.2017.2761892.

[55] L. Duan, D. Zhan, and J. Hohnerlein. “Optimizing Cloud Data Center Energy Efficiency via Dynamic Prediction of CPU Idle Intervals”. In: *Proc. 8th IEEE Int. Conf. Cloud Comput. (CLOUD)*. New York, NY, June 2015, pp. 985–988.

[56] R. Bolla et al. “Green Network Technologies and the Art of Trading-off”. In: *Proc. 2011 IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*. Shanghai, China, Apr. 2011, pp. 301–306.

[57] A. S. Cassidy and A. G. Andreou. “Beyond Amdahl’s Law: An Objective Function That Links Multiprocessor Performance Gains to Delay and Energy”. In: *IEEE Trans. Comput.* 61.8 (Aug. 2012), pp. 1110–1126.

[58] P. Mell and T. Grance. *The NIST definition of Cloud Computing*. NIST Special Publ. 800-145. 2011. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[59] L.M. Vaquero, L. Rodero-Merino, and R. Buyya. “Dynamically Scaling Applications in the Cloud”. In: *SIGCOMM Comput. Commun. Rev.* 41.1 (Jan. 2011), pp. 45–52.

[60] A. Fischer et al. “Virtual Network Embedding: A Survey”. In: *IEEE Commun. Surveys Tuts.* 15 (2013), pp. 1888–1906.

[61] S. D’Oro et al. “Exploiting Congestion Games to Achieve Distributed Service Chaining in NFV Networks”. In: *IEEE J. Sel. Areas Commun.* 35.2 (Feb. 2017), pp. 407–420.

[62] A. Leivadeas et al. “A Graph Partitioning Game Theoretical Approach for the VNF Service Chaining Problem”. In: *IEEE Trans. Netw. Service Manag.* 14.4 (Dec. 2017), pp. 890–903.

[63] L. Yang et al. “Cost Aware Service Placement and Load Dispatching in Mobile Cloud Systems”. In: *IEEE Trans. Comput.* 65.5 (May 2016), pp. 1440–1452.

[64] V. Burger et al. “Load Dynamics of a Multiplayer Online Battle Arena and Simulative Assessment of Edge Server Placements”. In: *Proc. 7th Int. Conf. Multimedia Syst. (MMSys)*. Klagenfurt, Austria, May 2016, 17:1–17:9.

[65] M. Aicardi et al. "A Decentralized Team Routing Strategy among Telecom Operators in an Energy-Aware Network". In: *Proc. 2015 SIAM Conf. Control Appl.* Paris, France, July 2015, pp. 340–347.

[66] B. Heller et al. "ElasticTree: Saving Energy in Data Center Networks". In: *Proc. 7th USENIX Conf. Netw. Syst. Design Implementation (NSDI)*. San Jose, CA, Apr. 2010. URL: http://dl.acm.org/citation.cfm?id=1855711.1855728.

[67] M. H. Ferdaus et al. "Virtual Machine Consolidation in Cloud Data Centers Using ACO Metaheuristic". In: *Proc. 2014 Int. European Conf. Parallel Process. (Euro-Par)*. Porto, Portugal, Aug. 2014, pp. 306–317.

[68] W. Song et al. "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing". In: *IEEE Trans. Comput.* 63.11 (Nov. 2014), pp. 2647–2660.

[69] A. Murtazaev and S. Oh. "Sercon: Server Consolidation Algorithm using Live Migration of Virtual Machines for Green Computing". In: *IETE Tech. Rev.* 28.3 (June 2011), pp. 212–231.

[70] L. Guo et al. "A Game Based Consolidation Method of Virtual Machines in Cloud Data Centers With Energy and Load Constraints". In: *IEEE Access* 6 (2018), pp. 4664–4676.

[71] M. M. Tajiki et al. "Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining". In: *IEEE Trans. Netw. Service Manag.* (2018). DOI: 10.1109/TNSM.2018.2873225.

[72] R. Bruschi, A. Carrega, and F. Davoli. "A Game for Energy-Aware Allocation of Virtualized Network Functions". In: *J. Electr. Comput. Eng.* (Feb. 2016). DOI: 10.1155/2016/4067186.

[73] R. Bolla et al. "Fine-Grained Energy-Efficient Consolidation in SDN Networks and Devices". In: *IEEE Trans. Netw. Service Manag.* 12.2 (June 2015), pp. 132–145.

[74] C. Raiciu et al. "Improving Datacenter Performance and Robustness with Multipath TCP". In: *SIGCOMM Comput. Commun. Rev.* 41.4 (Aug. 2011), pp. 266–277.

[75] G. Detal et al. "Revisiting Flow-based Load Balancing: Stateless Path Selection in Data Center Networks". In: *Comput. Netw.* 57.5 (Apr. 2013), pp. 1204–1216.

[76] M. Shafiee and J. Ghaderi. "A Simple Congestion-Aware Algorithm for Load Balancing in Datacenter Networks". In: *IEEE Trans. Netw.* 25.6 (Dec. 2017), pp. 3670–3682.

[77] Z.Á. Mann. "Allocation of Virtual Machines in Cloud Data Centers: A Survey of Problem Models and Optimization Algorithms". In: *ACM Comput. Surveys* 48.1 (Sept. 2015).

[78] R. Bruschi et al. "An SDN/NFV Platform for Personal Cloud Services". In: *IEEE Trans. Netw. Service Manag.* (2017). DOI: 10.1109/TNSM.2017.2761860.

[79] T. Braud et al. “Future Networking Challenges: The Case of Mobile Augmented Reality”. In: *Proc. 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*. Atlanta, GA, June 2017, pp. 1796–1807.

[80] T. Taleb, A. Ksentini, and P. Frangoudis. “Follow-Me Cloud: When Cloud Services Follow Mobile Users”. In: *IEEE Trans. Cloud Comput.* (2017). DOI: 10.1109/TCC.2016.2525987.

[81] N. Tziritas et al. “Online Inter-Datacenter Service Migrations”. In: *IEEE Trans. Cloud Comput.* (2017). DOI: 10.1109/TCC.2017.2680439.

[82] W. Cerroni and F. Esposito. “Optimizing Live Migration of Multiple Virtual Machines”. In: *IEEE Trans. Cloud Comput.* (2017). DOI: 10.1109/TCC.2016.2567381.

[83] G. Sun et al. “Live Migration for Multiple Correlated Virtual Machines in Cloud-based Data Centers”. In: *IEEE Trans. Services Comput.* (2017). DOI: 10.1109/TSC.2015.2477825.

[84] C. Möbius, W. Dargie, and A. Schill. “Power Consumption Estimation Models for Processors, Virtual Machines, and Servers”. In: *IEEE Trans. Parallel Distrib. Syst.* 25.6 (June 2014), pp. 1600–1614.

[85] M. Dayarathna, Y. Wen, and R. Fan. “Data Center Energy Consumption Modeling: A Survey”. In: *IEEE Commun. Surveys Tuts.* 18.1 (2016), pp. 732–794.

[86] M. Dimitrov et al. “Coordinated Optimization: Dynamic Energy Allocation in Enterprise Workload”. In: *Intel® Technol. J.* 16.2 (2012), pp. 32–51.

[87] *Volume 3B: System Programming Guide, Part 2*. Intel® 64 and IA-32 Architectures Software Developer’s Manual. 2016. URL: https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf.

[88] J. Prados-Garzon et al. “Latency Evaluation of a Virtualized MME”. In: *Proc. 2015 Wireless Days (WD)*. Toulouse, France, Mar. 2016. DOI: 10.1109/WD.2016.7461500.

[89] W. Chiang and J. Wen. “Design and Experiment of NFV-Based Virtualized IP Multimedia Subsystem”. In: *Proc. 3rd Int. Conf. Comput. Commun. Syst. (ICCCS)*. Nagoya, Japan, Apr. 2018, pp. 397–401.

[90] M. S. Yoon and A. E. Kamal. “NFV Resource Allocation Using Mixed Queuing Network Model”. In: *Proc. 2016 IEEE Global Commun. Conf. (GLOBECOM)*. Washington, DC, Dec. 2016. DOI: 10.1109/GLOCOM.2016.7842023.

[91] F. C. Chua et al. “Stringer: Balancing Latency and Resource Usage in Service Function Chain Provisioning”. In: *IEEE Internet Comput.* 20.6 (Nov. 2016), pp. 22–31.

[92] S. Gebert et al. "Performance Modeling of Softwarized Network Functions Using Discrete-Time Analysis". In: *Proc. 28th Int. Teletraffic Congr. (ITC28)*. Vol. 1. Würzburg, Germany, Sept. 2016, pp. 234–242.

[93] G. Choudhury and A. Krishnamoorthy. "Analysis of the $M^X/G/1$ Queue with a Random Setup Time". In: *Stoch. Anal. Applicat.* 22.3 (2004), pp. 739–753.

[94] H. S. Lee and M. M. Srinivasan. "Control Policies for the $M^X/G/1$ Queueing System". In: *Manage. Sci.* 35.6 (1989), pp. 708–721.

[95] J. Medhi. *Stochastic Models in Queueing Theory*. 2nd. Elsevier Science USA, 2003.

[96] E. Hyytiä et al. "Value (Generating) Functions for the $M^X/G/1$ Queue". In: *Proc. 29th Int. Teletraffic Congr. (ITC29)*. Vol. 1. Genoa, Italy, Sept. 2017, pp. 232–240.

[97] H. Tijms. *A First Course in Stochastic Models*. John Wiley & Sons Ltd, England, 2003.

[98] A. Klemm, C. Lindemann, and M. Lohmann. "Modeling IP Traffic using the Batch Markovian Arrival Process". In: *Perform. Eval.* 54 (Oct. 2003), pp. 149–173.

[99] P. Salvador, A. Pacheco, and R. Valadas. "Modeling IP Traffic: Joint Characterization of Packet Arrivals and Packet Sizes using BMAPs". In: *Comput. Networks* 44 (Feb. 2004), pp. 335–352.

[100] D. Lucantoni. "The BMAP/G/1 Queue: A Tutorial". In: *Perform. Eval. Comput. Commun. Syst.* Ed. by L. Donatiello and R. Nelson. Springer Berlin Heidelberg, 1993, pp. 330–358.

[101] P. Mevert. *The Alternating Queueing Process with Setup Times*. Tech. Memo. 63. Case Institute of Technology, June 1966.

[102] J.-C. Ke. "Batch Arrival Queues Under Vacation Policies With Server Breakdowns and Startup/Closedown Times". In: *Appl. Math. Model.* 31.7 (2007), pp. 1282–1292.

[103] R. Bolla, R. Bruschi, and P. Lago. "The Hidden Cost of Network Low Power Idle". In: *Proc. 2013 IEEE Int. Conf. Commun. (ICC)*. Budapest, Hungary, June 2013, pp. 4148–4153.

[104] R. Schöne, D. Molka, and M. Werner. "Wake-up Latencies for Processor Idle States on Current x86 Processors". In: *Comput. Sci. - Res. Develop.* 30.2 (2015), pp. 219–227.

[105] *KVM*. URL: http://www.linux-kvm.org/.

[106] W. von Hagen. *Using KVM Virtualization*. Tech. rep. 2014. URL: https://www.ibm.com/developerworks/library/l-using-kvm/l-using-kvm-pdf.pdf.

[107] S. Zeng and Q. Hao. "Network I/O Path Analysis in the Kernel-Based Virtual Machine Environment Through Tracing". In: *Proc. 1st Int. Conf. Inform. Sci. Eng. (ICISE)*. Nanjing, China, Dec. 2009, pp. 2658–2661.

[108] C. Xu et al. "Energy Efficiency of Cloud Virtual Machines: From Traffic Pattern and CPU Affinity Perspectives". In: *IEEE Syst. J.* 11 (June 2017), pp. 835–845.

[109] *Ethtool(8) - Linux Man Page*. URL: https://linux.die.net/man/8/ethtool.

[110] *Idlestat - A CPU Power-state Analysis Tool*. URL: http://manpages.ubuntu.com/manpages/xenial/man1/idlestat.1.html.

[111] *vnStat - A Console-based Network Traffic Monitor*. URL: http://manpages.ubuntu.com/manpages/bionic/man1/vnstat.1.html.

[112] P. W. Glynn and W. Whitt. "Estimating the Asymptotic Variance with Batch Means". In: *Oper. Res. Lett.* 10.8 (Nov. 1991), pp. 431–435.

[113] *OpenWrt*. URL: https://openwrt.org/.

[114] R. Bolla et al. "An Experimental Evaluation of the TCP Energy Consumption". In: *IEEE J. Sel. Areas Commun.* 33.12 (Dec. 2015), pp. 2761–2773.

[115] *Turbostat - Report Processor Frequency and Idle Statistics*. URL: http://manpages.ubuntu.com/manpages/xenial/man8/turbostat.8.html.

[116] D. Hackenberg et al. "Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison". In: *Proc. 2013 IEEE Int. Symp. Perform. Anal. Syst. Softw.* Austin, TX, Apr. 2013, pp. 194–204.

[117] R. Bruschi et al. "Model-based Analytics for Profiling Workloads in Virtual Network Functions". In: *Proc. 2017 IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*. Atlanta, GA, May 2017, pp. 916–921.

[118] "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture". In: *IEEE Std 802-2014 (Rev. IEEE Std 802-2001)* (June 2014), pp. 1–74.

[119] "IEEE Standards for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks". In: *IEEE Std 802.1Q-2003* (May 2003).

[120] The 3GPP Association. *System Architecture for the 5G System*. 3GPP TS 23.501, Stage 2, Rel. 15, v.15.2.0. June 2018.

[121] *Network Functions Virtualisation (NFV); Management and Orchestration; Vi-Vnfm Reference Point – Interface and Information Model Specification*. ETSI NFV ISG Spec. 2016. URL: http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/006/02.01.01_60/gs_NFV-IFA006v020101p.pdf.

[122] X. Meng, V. Pappas, and L. Zhang. "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement". In: *Proc. 2010 Int. Conf. Comput. Commun. (INFOCOM)*. DOI: 10.1109/INFCOM.2010.5461930. San Diego, CA, Mar. 2010.

[123] M. F. Bari et al. "Data Center Network Virtualization: A Survey". In: *IEEE Commun. Surveys Tuts.* 15.2 (2013), pp. 909–928.

[124] *Cisco Data Center Spine-and-Leaf Architecture: Design Overview*. Cisco White Paper. 2016. URL: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.pdf.

[125] C. Guo et al. "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers". In: *Proc. 2009 ACM SIGCOMM Conf. (SIGCOMM)*. Barcelona, Spain, Aug. 2009.

[126] R. Bruschi et al. "Evaluating the Impact of Micro-Data Center (μDC) Placement in an Urban Environment". In: *Proc. 2018 IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*. Verona, Italy, Nov. 2018.

[127] R. Bruschi et al. "A Scalable SDN Slicing Scheme for Multi-domain Fog/Cloud Services". In: *Proc. 2017 IEEE Conf. Netw. Softwarization (NetSoft)*. DOI: 10.1109/-NETSOFT.2017.8004244. Bologna, Italy, July 2017.

[128] *OpenStack*. URL: https://www.openstack.org/.

[129] R. Bruschi et al. "OpenStack Extension for Fog-Powered Personal Services Deployment". In: *Proc. 29th Int. Teletraffic Congr. (ITC)*. Vol. 2. Genoa, Italy, Sept. 2017, pp. 19–23.

[130] *OpenVolcano – Open Virtualization Operating Layer for Cloud/fog Advanced NetwOrks*. URL: http://openvolcano.org.

[131] *The Internet Topology Zoo*. URL: http://topology-zoo.org.

[132] T. Wood et al. "CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines". In: *IEEE/ACM Trans. Netw.* 23.5 (Oct. 2015), pp. 1568–1583.

[133] E. G. Coffman Jr., M. R. Garey, and D. S. Johnson. "Approximation Algorithms for Bin Packing: A Survey". In: *Approximation Algorithms for NP-Hard Problems*. Ed. by D. S. Hochbaum. PWS Publishing, Boston, MA, 1996.

[134] J. W. Cohen. "A Two-queue, One-server Model with Priority for the Longer Queue". In: *Queueing Syst. Theory Appl* 2.3 (Nov. 1987), pp. 261–283.

[135] *Best Practices in Core Network Capacity Planning*. Cisco White Paper. 2013. URL: http://www.cisco.com/c/en/us/products/collateral/routers/wan-automation-engine/white_paper_c11-728551.pdf.

[136] Z. Huang and D.H.K. Tsang. "SLA Guaranteed Virtual Machine Consolidation for Computing Clouds". In: *Proc. 2012 IEEE Int. Conf. Commun. (ICC)*. Ottawa, ON, Canada, June 2012, pp. 1314–1319.

[137] *Intel® Xeon® Processor E5-Product Family Datasheet*. URL: http://www.intel.com/content/dam/www/pub-lic/us/en/documents/datasheets/xeon-e5-1600-2600-vol-1-datasheet.pdf.

[138] *Network Function Virtualization: Intel® Data Plane Development Kit vSwitch with Linux Virtualization and Intel® Architecture*. URL: https://networkbuilders.intel.com/docs/Network_Buil-ders_RA_DPDK_vSwitch_Final.pdf.

[139] *Intel Xeon Processor E5-2690: Enterprise Workload Performance while Running Storage Efficiency Tasks*. URL: http://www.intel.fr/content/dam/www/public/us/en/documents/reports/performance-xeon-e5-2690-pt-report.pdf.

[140] *Energy Price Statistics*. URL: http://ec.europa.eu/euro-stat/statistics-explained/index.php/Energy_price_statistics.

[141] Telecom Italia Sparkle S.p.A. *Sparkle / Our World - Telecom Italia Sparkle*. 2017. URL: www.world.tisparkle.com/.

[142] Y. Ho, M. Kastner, and E. Wong. "Teams, Signaling, and Information Theory". In: *IEEE Trans. Autom. Control* 23.2 (Apr. 1978), pp. 305–312.

[143] S. Yüksel and T. Başar. *Stochastic Networked Control Systems: Stabilization and Optimization under Information Constraints*. Birkhäuser Basel, New York, NY, 2013.

[144] R. Bolla et al. "Optimizing the Power-delay Product in Energy-aware Packet Forwarding Engines". In: *Proc. 24th Tyrrhenian Int. Workshop Digit. Commun. (TIWDC) - Green ICT*. Genoa, Italy, Sept. 2013. DOI: 10.1109/TIWDC.2013.6664195.

[145] J. Marshak and R. Radner. *The Economic Theory of Teams*. Yale University Press, New Haven, CT, 1971.