Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Kommunikationsnetze

Modeling and Analysis of Software Defined Virtual Mobile Core Networks

Arsany Amir Saber Basta, M.Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Prof. Dr.-Ing. Klaus Diepold
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Wolfgang Kellerer
	2.	Prof. Dr. Stefan Schmid

Die Dissertation wurde am 17.01.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 18.04.2017 angenommen.

Abstract

The mobile core network transports the mobile user traffic between the Radio Access Network (RAN), i.e., base stations (eNBs), and the external packet data networks, e.g., the Internet. In its current fourth generation LTE standard, the mobile core network is populated with several dedicated hardware network functions. Furthermore, the mobile core network relies on a distributed network control that is based on several standard specifications. Due to the tight integration of the mobile core functions in hardware and its inflexible network configuration, mobile operators face several challenges in terms of network deployment cycles, resource provisioning and network operation flexibility.

Recent networking concepts have emerged to address these challenges, namely Network Functions Virtualization (NFV), Network Virtualization (NV) and Software Defined Networking (SDN). NFV leverages the concepts of IT virtualization to network functions, where functions can be implemented in software and deployed on commodity hardware, e.g., data centers. Therefore, NFV offers more flexibility and more efficient resource utilization to mobile operators that is expected to reduce the cost of mobile networks. NV exploits the concept of IT virtualization, however, for the network elements themselves, i.e., nodes and links. NV enables the operation of multiple logical virtual networks, i.e., slices, on the same physical infrastructure. Hence, virtual network slices can provide a fast network deployment as well as efficient resource utilization of the network infrastructure. SDN, on the other hand, decouples the data and control planes of network functions and introduces an open interface between the decoupled planes. In this way, SDN offers a programmable network with a centralized control view, which offers an efficient network control and adds more flexibility to the network configuration.

In this thesis, we analyze the design of the LTE mobile core network architecture based on SDN and NFV. We model the different designs as network optimization problems, which are based on a detailed analysis of the LTE mobile core network functions and include observations from prototype measurements. The proposed models aim at an optimal dimensioning and planning for an SDN and NFV mobile core network with respect to several cost factors, e.g., network cost and data center resources cost. The different involved network parameters, e.g., number of available data centers, required performance metrics, e.g., latency, and objective cost factors are thoroughly evaluated in order to provide guidelines for the optimal core network design. Additionally, in this thesis, we propose several concepts and solutions for an SDN network virtualization layer. The proposed solutions offer flexibility in the allocation of virtualization functions, isolation for the shared resources among the tenants and support for a dynamic operation. Therefore, the virtual SDN mobile core networks can provide a reliable and flexible operation as well as support more network dynamic changes. The developed concepts are evaluated through prototype implementation on an SDN testbed setup.

Kurzfassung

Das Kernnetz eines Mobilfunknetzes ist zuständig für die Übertragung des Nutzerverkehrs zwischen dem Radio Access Network (RAN), d.h. den Basisstationen, und den externen Paketdatennetzen, z.B. dem Internet. Im aktuellen LTE Standard der vierten Generation Mobilfunk besteht das Kernnetz aus mehreren Netzfunktionen, die in Hardware realisiert sind. Das Kernnetz beruht dabei auf einer verteilten Netzsteuerung, die wiederum auf mehreren Standardspezifikationen basiert. Aufgrund der Implementierung der Kernnetzfunktionen in Hardware und der damit einhergehenden unflexiblen Netzkonfiguration sind die Mobilfunkbetreiber mehreren Einschränkungen hinsichtlich des Netzausbaus, der Ressourcenbereitstellung und der Flexibilität des Netzbetriebs ausgesetzt.

Neue Netzkonzepte sind entstanden, um diese Herausforderungen zu bewältigen, wie Network Functions Virtualization (NFV), Network Virtualization (NV) und Software Defined Networking (SDN). NFV wendet die Konzepte der IT-Virtualisierung auf Netzfunktionen an, wobei Funktionen in Software implementiert werden und anschließend auf Standardhardware, z.B. Rechenzentren, laufen können. NFV bietet den Mobilfunkanbietern damit mehr Flexibilität und eine effizientere Ressourcennutzung, welche die Netzkosten reduzieren soll. NV nutzt das Konzept der IT-Virtualisierung für die Netzelemente selbst, d.h. Knoten und Kanten. NV ermöglicht dadurch den Betrieb mehrerer logischer virtueller Netze auf derselben physikalischen Infrastruktur. Demzufolge bieten virtuelle Netze einen schnellen Netzausbau sowie effiziente Ressourcennutzung der Netzinfrastruktur. SDN hingegen entkoppelt die Daten- und Steuerungsebenen von Netzfunktionen und führt eine Programierschnittstelle zwischen den entkoppelten Ebenen ein. Dementsprechend bietet SDN ein programmierbares Netz mit einer zentralisierten Steuerung, die zu einer effizienten Netzsteuerung führt und die Netzkonfiguration flexibler macht.

In dieser Dissertation wird das Design einer SDN- und NFV-basierten LTE Kernnetzarchitektur entworfen und analysiert. Die unterschiedlichen Designs werden als Netzwerkoptimierungsprobleme basierend auf einer detaillierten Analyse der LTE Netzfunktionen und Messungen von Prototypen modelliert. Die Modelle zielen auf eine optimale Dimensionierung und Planung eines SDN- und NFV-Kernnetzes in Bezug auf mehrere Kostenfaktoren, z.B. Netzkosten und Rechenzentrumsressourcen, ab. Um Richtlinien für das optimale Design des Kernnetzes zu liefern, werden unterschiedliche Netzwerkparameter, z.B. Anzahl der verfügbaren Datenzentren, sowie Leistungsmetriken, z.B. Latenzzeit, und Kostenfaktoren ausgewertet. Darüber hinaus werden in dieser Arbeit mehrere Konzepte und Lösungen für SDN Netzvirtualisierung entwickelt. Die vorgeschlagenen Lösungen bieten Flexibilität bei der Zuweisung von Virtualisierungsfunktionen, Isolation der Ressourcen und Unterstützung für den dynamischen Betrieb. Auf diese Weise können die logischen virtuellen Kernnetze einen zuverlässigen und flexiblen Betrieb sowie die zukünftige Netzdynamik bereitstellen. Die entwickelten Konzepte werden durch Implementierung von Prototypen auf einem SDN Testbed ausgewertet.

Contents

Chapter 1

Introduction

Today's mobile network operators require effective ways to cope with the rapid growth of user traffic, service innovation and the persistent necessity to reduce costs [1]. There are several challenges that arise with the current 4th generation LTE mobile core architecture. First, the mobile core comprises of several data and control functions that are tightly coupled to underlying hardware. This makes the current architecture very challenging to upgrade, results in long deployment cycles and induces high cost. Second, the network control plane is distributed among several functions in the mobile core, which results in rigid configurations and lack of flexible and on run time control. These challenges have an impact on mobile operators in terms of low innovation and high cost, while it also reflects on users through limited quality improvements and slow service upgrades [2].

In networking, several concepts have emerged aiming at cost reduction, increase of network scalability and service flexibility, namely Network Functions Virtualization (NFV) [3] and Software Defined Networking (SDN) [4]. NFV proposes the mitigation of the dependency on hardware by running network functions as software instances on commodity servers or data centers. Hence, NFV coupled with cloud computing provides a more flexible mobile core network architecture that can be upgraded and tailored to the user demands. On the other hand, SDN provides a split of network control from data forwarding. Hence, for the mobile core network, it supports a decomposition of the mobile network into control plane and data plane functions. Therefore, SDN provides the control programmability that empowers the operators with dynamic and on run time control.

Our goal is to design and model the next generation mobile core network based on SDN and NFV. The proposed models aim at quantifying the gain and overhead of SDN and NFV to achieve an optimal network dimensioning and resources management. Different architectures for the next generation core network are developed based on the analysis and classification of the mobile core network functions and their inter-dependencies. New network models for the new realization of the core functions based on SDN and NFV are proposed. The network models are developed according to observations taken from prototype implementation and performance evaluation. Furthermore, traffic models are extended to describe and correlate the data and control traffic in the next generation core architecture.

The optimal dimensioning and resource planning models take into account several performance metrics that define the network performance and service quality. An example to such key performance metrics are the data and control plane latency. An example operator strategy is to optimize the network traffic resources and hence reduce the network cost. An alternative strategy is to optimize the server infrastructure resources, which in turn optimizes the server infrastructure cost. In addition, operators could target multiple operation features, e.g., energy savings and coping with dynamic traffic changes. In this work, linear optimization models are proposed for the different key performance metrics, operator strategies and operation features for the next generation network. State-of-the-art network flow models and facility location problems are considered as basis for the proposed problem formulations. The developed models are evaluated through quantitative analysis to the varying performance metrics, network design and setup parameters. Trade-offs are drawn between the different operator strategies, with guidelines and correlations between the input parameters.

Finally, in order to achieve further deployment flexibility to the mobile core, Network Virtualization (NV) [5] is considered for the next generation mobile core network. As a general concept, NV enables a physical network infrastructure to be shared among multiple tenants. In case virtualization is applied to SDN networks, different applications or stake holders would be able to operate isolated and flexible SDN network slices using their own controllers. Our goal is to evaluate state-of-the-art SDN virtualization solutions, which are widely known as SDN hypervisors [6], with respect to their features and capabilities to provide isolated, flexible and dynamic slices. In this work, several features are proposed and implemented for an

SDN virtualization layer to improve the virtual SDN networks' performance. Additionally, multiple procedures are introduced to facilitate the dynamic and flexible operation of virtual next generation core slices. The performance of the introduced SDN virtualization layer is evaluated through measurements on a real testbed. Performance comparison to available SDN virtualization solutions is provided.

1.1 Problem Statement

Mobile network operators opt to migrate their core networks to the new network concepts, namely Software Defined Networking (SDN), Network Functions Virtualization (NFV) and Network Virtualization (NV). With all available solutions, operators need to learn how to apply these concepts where it brings the most benefit and to maximize their gain. For instance, we can observe that each concept, either SDN or NFV, results in different paths for both data and control planes. This results in different latency for data and control planes, that are influenced by the locations of the data centers and, in case of SDN, by the control paths between the SDN controllers and switches, as well. Moreover, each path impacts the traffic distribution over the network, where an NFV deployment adds data plane traffic on the links towards the data centers while SDN adds extra overhead traffic for the SDN control plane. As for the data center resources aspect, the virtual network functions require more computational and networking resources than the SDN controllers, since they involve the processing of high volume data plane traffic. SDN and NFV deployments show trade-offs in terms of latency, network traffic and data center resources. Hence, novel optimization models are required to find optimal planning and dimensioning for a mobile core network, that includes both SDN and NFV deployments, in terms of the network load cost and the data center resources cost, showing trade-offs between both cost factors.

Additionally, in order to achieve the promised gains of these new concepts, operators need to evaluate the performance of the possible solutions in different environments. For instance, the overhead of a decoupled control and data plane in SDN is needed to be performed. For NFV, the data plane performance needs to be benchmarked on commodity hardware with different acceleration tools to get the desired performance in order to meet carrier grade requirements. Since mobile networks require special functionality, such as support for mobility, need for user authentication and accounting, the proposed interfaces in the new architectures are required to be

tested and optimized for the mobile network use case. For this purpose, new software implementations need to be developed to realize the network functions as well as new benchmarking testbeds and tools. The results of performance benchmarks are also needed to formulate more realistic optimization models for the dimensioning and planning tools.

Finally, the next generation network advocates for more sharing of the physical infrastructure through network virtualization, i.e., slicing. Existing network virtualization concepts need to be extended and tested to support the new requirements of the next generation networks. In particular, the support for a reliable operation between the multiple slices is necessary. It is also crucial to achieve a better understanding for the performance of virtual slices compared to non virtual networks, in terms of slice performance as well the performance of the virtualization layer itself. For this purpose, new tools and testbed environments are needed to test the concepts of the proposed slicing solutions and network virtualization concepts. The tests would also show insights to improve the proposed solutions and point out the limitations of network virtualization in the context of sharing the physical infrastructure.

1.2 Main Contributions

In this thesis, we investigate the technology migration of the current LTE mobile core network towards the next generation 5G. The main drivers for the technology migration are the concepts of SDN and NFV. Additionally, network slicing and virtualization are investigated as key features in the next generation 5G. The main contributions of this thesis can be summarized as follows:

- **Analysis of the LTE mobile core network**: a comprehensive analysis to the mobile core network functions is provided. The analysis provides an overview of specific network functions of the mobile core network. In addition, the main procedures, that are needed to operate a mobile network compared to traditional IP networks, i.e., Internet, are analyzed. The suitability and implementation of SDN based or NFV based core network functions is presented.
- **Prototype evaluation:** initial performance evaluation for core network functions deployed under the concepts of SDN and NFV is presented. The performance benchmarks are needed to describe the performance of the new SDN

and NFV based mobile core network functions. Real testbed equipment, i.e., switches and servers, as well as emulation environments are utilized. The initial measurements are used for the modeling phase of the network functions by providing input to the performance parameters to the different network functions involved in the optimization models.

- **Optimization models:** optimization models for the dimensioning and planning of an SDN and NFV based mobile core network are formulated. Several optimization models are proposed and evaluated for a new generation of the mobile core network. Different cost aspects are evaluated, such as network transport cost, data centers cost and energy savings. The optimization models provide telco operators with tools to plan the next generation network with an optimal cost, where each operator can target different optimal cost objectives and network constraints.

- **Design of an SDN virtualization layer:** For the next generation mobile core networks, several solutions are proposed to provide a reliable and dynamic slicing for the mobile core network. Network slicing enables operators to deploy several virtual slices that share the same physical infrastructure. In order to reach these goals, several features need to be provided. A key feature is resource isolation, where different solutions are proposed. Additionally, concepts for dynamic slicing are presented.

- **Performance evaluation of the SDN virtualization layer**: Testbed measurements are carried out to evaluate the performance of the proposed SDN virtualization layer used to slice next generation core network. Several new performance criteria are brought up through slicing, e.g., the performance of the virtual networks of several tenants as they share the same physical infrastructure. Additionally, a key performance factor is the performance of the intermediate virtualization layer itself and its induced overhead.

1.3 Author's Relevant Publications

A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "***A virtual SDN-enabled LTE EPC architecture: a case study for S-/P-gateways functions***," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "***Applying NFV and SDN to LTE mobile core gateways, the functions placement problem***," in *Proceedings of the 4th workshop on All things cellular, ACM SIGCOMM*, 2014, pp. 33–38.

A. Basta, A. Blenk, M. Hoffmann, H. J. Morper, K. Hoffmann, and W. Kellerer, "***SDN and NFV dynamic operation of LTE EPC gateways for time-varying traffic patterns***," in *International Conference on Mobile Networks and Management*, 2014, pp. 63–76.

A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "***Towards a cost optimal design for a 5G mobile core network based on SDN and NFV***," in *IEEE Transactions on Network and Service Management (TNSM)*, to be published, 2017.

A. Blenk, **A. Basta**, and W. Kellerer, "***HyperFLEX: an SDN virtualization architecture with flexible hypervisor function allocation***," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 397–405.

A. Basta, A. Blenk, Y.-T. Lai, and W. Kellerer, "***HyperFLEX: demonstrating control-plane isolation for virtual software-defined networks***," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1163–1164.

A. Basta, A. Blenk, H. B. Hassine, and W. Kellerer, "***Towards a dynamic SDN virtualization layer: control path migration protocol***," in *International Conference on Network and Service Management (CNSM)*, 2015, pp. 354–359.

1.4 Thesis Outline

The remainder of the thesis is structured as follows.

Chapter 2 provides a background to the relevant networking concepts, namely SDN, NFV and NV. It also provides a detailed analysis for the LTE mobile core network functions and their realizations with SDN and NFV.

Chapter 3 presents the different mobile core network designs based on SDN and NFV. The new network designs are modeled as optimization problems that find the optimal placement of the network functions while minimizing the network cost. An evaluation is conducted for several core network design parameters for static as well as time-varying data traffic patterns.

In **Chapter 4**, several dimensioning and planning models are proposed for an SDN and NFV mobile core network with respect to different cost factors, e.g., network and data center resources cost. Pareto optimal solutions that find the balance between the cost factors are introduced. Extensive evaluation for the proposed models is carried out comparing different network parameters and performance metrics.

Chapter 5 introduces concepts and solutions for a flexible, reliable and dynamic SDN virtualization layer. The developed concepts are evaluated through prototype implementations and measurements on a real SDN testbed setup.

Finally, **Chapter 6** provides an overall discussion and conclusions for the work presented in this thesis. Further directions for future work are outlined.

Chapter 2

Background and Analysis of LTE Mobile Core Network Functions

2.1 Background

In this chapter, important background is presented for the concepts and technologies related to this work. The background entails the introduction to Software Defined networking (SDN), Network Function Virtualization (NFV) and Network Virtualization (NV). An introduction is provided to the LTE mobile core network architecture and operation. An analysis of the LTE mobile core network functions is based on our presented work in [7].

2.1.1 Software Defined Networking (SDN)

Software Defined Networking (SDN) [4] is an emerging network paradigm that decouples the network data plane from the control plane as illustrated in Fig. 2.1. In the context of SDN, network applications, e.g., firewall application, can be programmed in software forming the application plane that runs on top of the control plane. SDN introduces an open Application Program Interface (API) which defines an abstraction between the application, control and data planes. A standard open API for network programming results in a vendor agnostic network configuration by removing the dependency and diversity of the heterogeneous APIs from different vendors. This simplifies network management and contributes in developing more network functions and applications. The control plane is realized by SDN controllers that are implemented as programmable software that configures the SDN data plane through the southbound API. SDN controllers form the Networking Op-

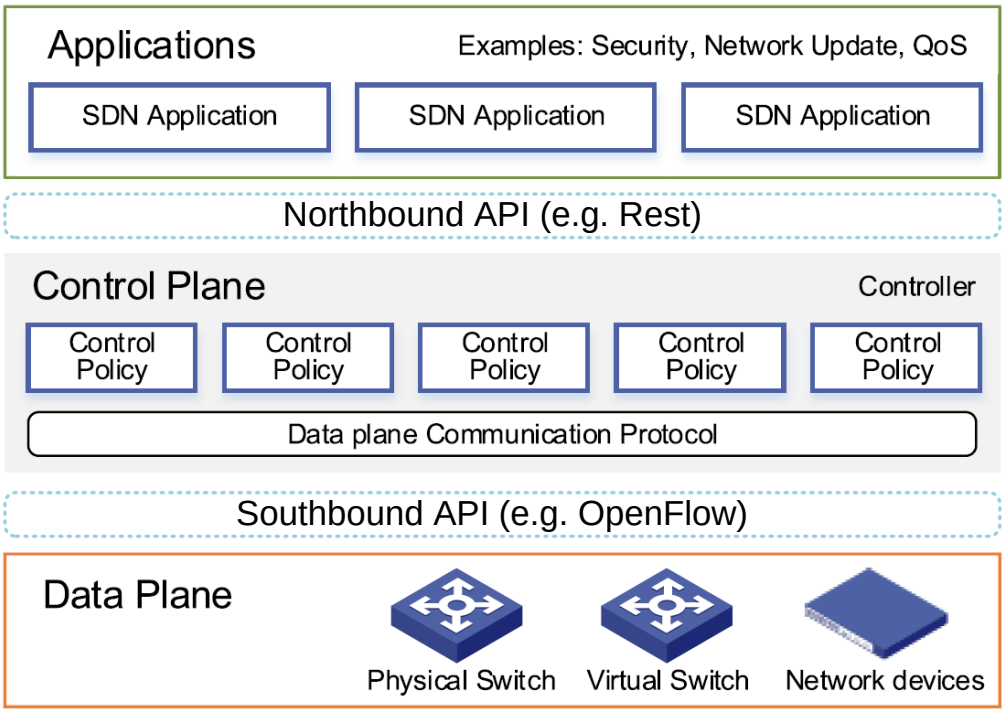


Figure 2.1: Software Defined Networking (SDN) architecture [8].

erating System (NOS) [9]. SDN controllers also offer another API for network applications, i.e., northbound API. In this way, SDN offers a programmable network that leads to increasing the flexibility in network management and control. Furthermore, SDN provides a logically centralized control view, i.e., a control plane that manages several network switches, which provides the operators with the possibility to achieve more efficient network control. Additionally, SDN adds the flexibility to steer the data traffic on run time basis and tailor the data flows inside the network. This leads to an improved service quality and user experience in addition to potential cost savings through a dynamic optimized network operation. However, a distributed control plane can be required, where a network is operated with multiple SDN controllers, in order to avoid single point of failure and to provide network control scalability [10].

Regarding the northbound API, there are several protocols and specifications which describe high level semantics of programming network applications on top of SDN controllers, e.g., REST API [11] or Pyretic [12]. The northbound languages provide high level abstraction for network applications to specify their policies for SDN controllers. As for the southbound API, Openflow (OF) [13] protocol is the current

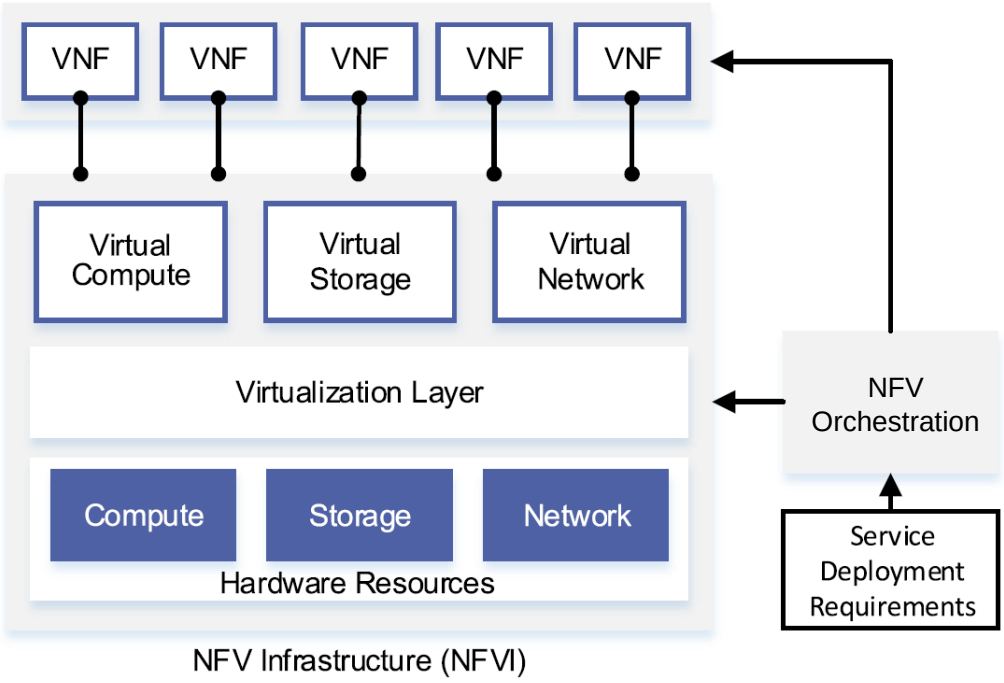


Figure 2.2: Network Functions Virtualization (NFV) architecture [8].

defacto specification for a standard open API for programming the SDN data plane. OpenFlow operates with a match and action paradigm, where a pair of match and action defines a flow rule. SDN controllers can program the network on run time per flow basis. A flow is defined as a set of packets which share a certain pattern, e.g., a header field. While limitations in OpenFlow are being discussed in standardization and new versions are released regularly, these limitations and related performance aspects motivated this work to look into the deployment and split of mobile core network functions from a more general point of view abstracting from a particular OpenFlow realization with respect to deployment of SDN for a mobile network.

2.1.2 Network Functions Virtualization (NFV)

In today's networks the deployment of new network services comes at a high cost for integration and operation as network functions typically come with separate hardware entities. Network Functions Virtualization (NFV) [3] [14] is a recent operators' initiative aiming at addressing this problem. NFV is transforming the way how operators architect their networks towards deploying network services onto virtualized industry standard servers, as depicted in Fig. 2.2. The Virtual Network Functions (VNF) can exploit computing, storage and networking resources of commodity

hardware which can be located for example in data centers. In this way, NFV involves delivering network functions as software that can run as virtualized instances through a virtualization layer, i.e., IT hypervisor such as VMware [15], which assigns virtual resources to virtual network functions. Consequently, VNF instances can be deployed at locations in the network as required, without the need to install hardware equipment for each new service. The management and deployment of VNFs on the available hardware infrastructure is carried out by an NFV orchestration entity, i.e., NFV orchestrator. The orchestrator decides on VNFs deployment based on its view of the assigned resources and capabilities of the hardware infrastructure [16].

With this migration from hardware to software running in a cloud environment, NFV is expected to lower not only equipment cost (capex) but also operational cost (opex). Services are expected to be deployed more flexibly and can be scaled up and down quickly. Operators can initially take their existing integrated network functions and modify the code base to adapt to cloud-based systems, e.g., by enabling it to utilize x86 hardware features where data plane processing on commodity servers can be accelerated by solutions such as Intel DPDK [17]. In this solution, the data and control planes in the network function are still integrated. In this work, we focus on this approach to deploy virtual network functions on commodity hardware.

2.1.3 Network Virtualization (NV)

Similar to the concept of NFV, Network Virtualization (NV) [5] extends the notion of virtualization from IT infrastructure to network infrastructure, i.e., nodes and links, as shown in Fig. 2.3. Such concept results in isolated logical slices, where the virtual topology is allocated on a subset of the computational and connectivity resources of the physical network, and is called a Virtual Network (VNet). This approach provides the capability to accommodate multiple independent and programmable network slices or virtual networks, which in addition could coexist and operate over a shared heterogeneous physical substrate which is a critical limitation, for instance, in overlay networks architecture. This means that virtual networks could be deployed over heterogeneous infrastructures operating alongside each other. Unlike the conventional role of Internet Service Providers (ISPs) at the current internet model, network virtualization multi-layer framework provides network connectivity and operation, compared to solely connectivity services as provided by VPNs for instance [18].

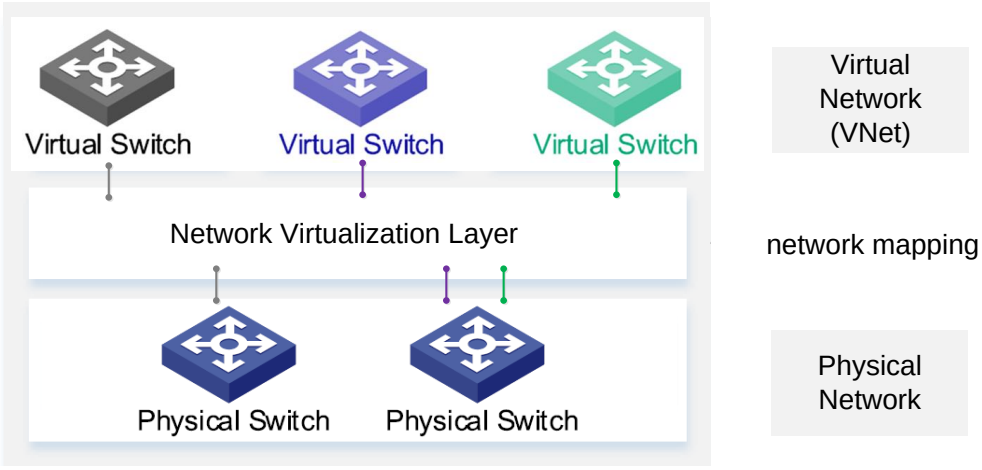


Figure 2.3: Network Virtualization (NV) architecture.

A main design goal of network virtualization is to provide operators and service providers with the flexibility to acquire a virtual network topology, on which they could apply their customized routing protocols, control functions and features independently from the technologies of the underlying physical infrastructures [19]. In addition, the key advantage of network virtualization is to provide programmable virtual networks with customized protocols and features [20]. Another fundamental design goal in operating virtual networks is to introduce the scalability that the Internet needs according to the desired requirements. In this work, we explore the application of network virtualization on Software Defined Networks in order to provide flexible and dynamic virtual mobile core network slices.

2.1.4 LTE Mobile Core Network

The mobile core network, in the latest LTE standard [21] as shown in Fig. 2.4, is responsible to transport the mobile user traffic between the Radio Access Network (RAN) and the Packet Data Networks (PDN), e.g., Internet or operator's services. For this purpose, the mobile core network comprises of several network functions that implement special operations that are needed for a mobile network [22].

The core network functions can be classified into two categories based on their purpose: (a) functions that handle the control plane only, such as the Mobility Management Entity (MME), Policy Charging and Rules Function (PCRF), Online Charging System (OCS) and the Home Subscriber Server (HSS) (b) functions that handle

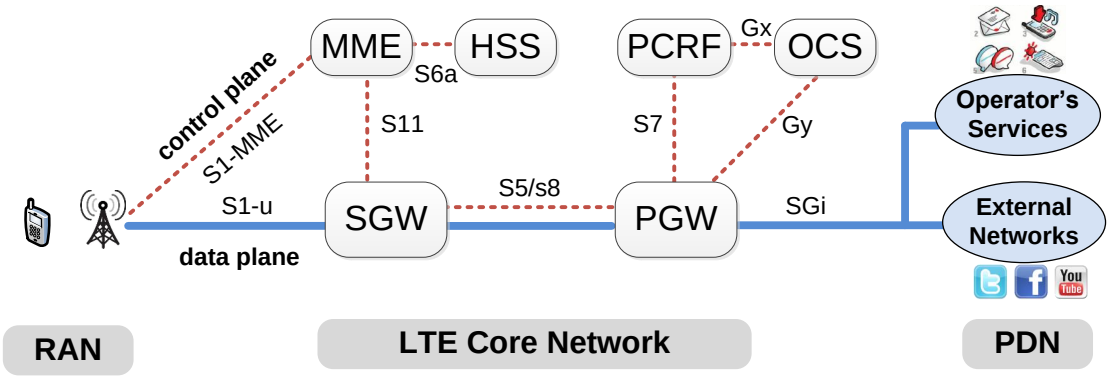


Figure 2.4: LTE mobile core network architecture. The figure shows the names of the interfaces (solid and dashed lines) between the mobile core network functions according to the latest LTE 3GPP standard [23].

both data as well as control planes, such as the Serving Gateway (SGW) and the Packet Data Network Gateway (PGW). The data plane functions implement special purpose processing for mobile networks, i.e., GTP tunneling for the user data in order to differentiate between the users and to be able to provide service quality classes for each user. Other data plane functions include charging and accounting for the user data usage. The control plane functions handle the setup of the user tunnels and mobility management, i.e., tracking area updates and redirection of user tunnels. Additionally, control functions are responsible for handling user authentication, subscription management and access control.

In the current system deployment as standardized in the LTE Evolved Packet Core (EPC), the mobile core network functions are typically deployed by separate dedicated hardware equipment developed by network vendors and deployed by network operators. As it can be observed from this architecture, the SGW and the PGW play a central role in the mobile core network as they are responsible for both the control as well as data plane handling including mobility management, Quality of Service (QoS) bearer handling, traffic policing and statistics. Hence, our analysis in this work for a realization of NFV and SDN is focusing on these two components. This next section is based on our work presented in [7], where we performed a detailed analysis of the LTE mobile core SGW and PGW network functions.

2.2 LTE Mobile Core Network Analysis

The objective of this analysis is to evaluate the SGW and PGW network functions of the LTE mobile core network architecture with respect to their potential deployment according to the NFV and SDN concepts. We decompose and classify the functions according to their impact on control plane and data plane processing. We observe realization challenges with our theoretical study looking into the deployment of the identified functions based on SDN technology using the OpenFlow protocol. Those challenges mainly address the deployment of functions that involve a certain data processing pattern such as tunneling and monitoring being part of SGW and PGW functions, which might demand for potentially new SDN component frameworks.

2.2.1 Network Function Analysis based on LTE Scenarios

We start with a thorough study of the SGW and PGW functionality based on basic EPC scenarios. A scenario defines the steps to be carried out to provide a certain operation for users in a mobile network, from which the main functions of the SGW and PGW are derived.

The *User Equipment (UE) attach* scenario serves the purposes of mutual user-network authentication, user location registration and establishment of a default bearer which has the basic policies and quality parameters [23]. The *UE detach* scenario occurs explicitly when a UE is switched off or implicitly when the network coverage is lost [24].

An EPC bearer is composed of two parts, namely the S1-U bearer and the S5/S8 bearer. The S1-U carries the traffic between the eNodeB and SGW while the S5/S8 carries the traffic between the SGW and PGW. For a better resource utilization in case of user inactivity, the established *S1-U bearer* is *released* while the S5/S8 bearer is preserved to provide the "always-on" feature of LTE [23]. Whenever a user becomes active again, a *service request* would initiate reestablishing the S1-U bearer in case the user utilizes only the *default bearer* [24] [25]. *Service request* is triggered to establish a *dedicated bearer* if additional quality parameters are needed. In this case, new S1-U and S5/S8 bearers are created that support the higher demand service [23] [26].

Addressing users' mobility, a *Tracking Area Update (TAU)* is required to update the tracking area, which is a set of multiple coverage cells, that takes place when an idle user moves to a new TA [23] [26]. The whole bearer is updated by both the

Table 2.1: SGW and PGW functions classification in chosen LTE Scenarios. The following function abbreviations are used: Control Signaling (Sig.), Resources Management Logic (Res.) and Data plane Forwarding (Fwd.).

Scenario	Node	Sig.	Res.	Rules	Fwd.	GTP	Filtering	Charging
UE Attach	SGW	✔	✔	✔	✔	✔	-	-
	PGW	✔	✔	✔	✔	✔	✔	✔
UE Detach	SGW	✔	✔	✔	-	-	-	-
	PGW	✔	✔	✔	-	-	-	-
S1-U Bearer	SGW	✔	✔	✔	-	-	-	-
Release	PGW	-	-	-	-	-	-	-
Request	SGW	✔	✔	✔	✔	✔	-	-
(Default)	PGW	-	-	-	-	-	-	-
Request	SGW	✔	✔	✔	✔	✔	-	-
(Dedicated)	PGW	✔	✔	✔	✔	✔	✔	✔
TAU	SGW	✔	✔	✔	-	-	-	-
	PGW	✔	✔	✔	-	-	-	-
Handover	SGW	✔	✔	✔	✔	✔	-	-
w/wo X2	PGW	-	-	-	-	-	-	-
Handover	SGW	✔	✔	✔	✔	✔	-	-
2G/3G	PGW	✔	✔	✔	✔	✔	✔	✔

SGW and PGW. While a *handover* typically takes place if an active user switches to 2G/3G network or moves to a cell covered by another eNodeB. The active session is transferred without interruption requiring bearer connection updates accordingly [23] [26].

2.2.2 Derived Functions of the SGW and PGW

In each aforementioned EPC scenario, we analyze and extract the main functions that are used in both the SGW and PGW. In addition, we assign a different color for each function block to ease the architecture's visual illustration. We have classified these functions into seven categories. Table 2.1 shows the set of classified functions utilized in each scenario. Each function can be used by different LTE procedure which indicates its deployment requirement in our later analysis.

Control Signaling: receives and triggers signaling messages used to exchange control information, where the corresponding functions are called accordingly. Such signaling function is required by both SGW and PGW.

Resources Management Logic: manages the node's available data plane resources as it assigns the bearer parameters such as the Tunnel End-point IDs (TEID). It also administrates the traffic shaping and users' policy enforcement as it allocates the bearer quality attributes based on users' QoS requirements as the Quality Class Identifier (QCI) or the Guaranteed Bit Rate (GBR) [27] for a dedicated bearer. This function is needed in both SGW and PGW, where the PGW utilizes it additionally to assign IP addresses to users attaching to the network.

Data plane Forwarding Rules: contains the data plane forwarding rules obtained from the resources management logic for all established bearers. The rules include the users' enforced policies on the data plane. In case of the SGW, it is mapping between the S1-U bearer and S5/S8 bearer of each user, while the PGW maps the S5/S8 bearer to the external IP packet data network.

Data plane Forwarding: represents the switching fabric or hardware, e.g., Network Interface Card (NIC) or switch backplane, that handles the data traffic and carries out the data flows processing. It is present in SGW and PGW.

GTP Encapsulation and Decapsulation: Both SGW and PGW use GPRS Tunneling Protocol (GTP), which is an IP tunneling protocol, to encapsulate the signaling and data traffic. GTP serves the purposes of supporting users' IP mobility, security provisioning and aggregating transport traffic. This function handles GTP headers, encapsulation and decapsulation.

Data plane Filtering and Classification : is responsible for identifying the packets based on users' profiles and policies for both uplink and downlink data traffic. This function is typically present at the PGW.

Charging Control: mainly takes place in the PGW only and can be classified into offline and online charging. An offline charging function collects Charging Data Records (CDR) in a post paid fashion while an online charging function allocates charging events on the data plane that are triggered when a condition meets a user's profile [28]. Offline charging can have different models based on data volume or data usage time, while online charging is event-based, e.g., expiry of user's credit. The charging function is generally implemented by the Policy and Charging Enforcement Function (PCEF) located at the PGW.

2.2.3 SGW and PGW Functions Realization via NFV

Conventional deployments of the SGW and PGW mobile network functions are usually based on integrated hardware platforms like ATCA [29]. For network vendors, adopting the NFV concept is an appealing option. The reason is that they can initially take their existing integrated network gateways and modify the code base to adapt to cloud based systems, e.g., by enabling it to leverage x86 hardware features using Intel's DPDK [17]. This reduces the time-to-market for a cloud based solution considerably. This applies to both control, e.g., control signaling, and data plane mobile functions, e.g., GTP tunneling. However, in this solution, the planes in the gateways are still integrated, which impacts deployment flexibility and speed of innovation. Additionally, the SGW and PGW data plane functions require more networking and processing resources on commodity hardware. This is due to the fact that the data plane traffic volume is several folds higher than the control plane traffic and that the data plane performance requirements are also higher than these of the control plane, e.g., in terms of processing latency.

2.2.4 SGW and PGW Functions Realization via SDN

SDN requires the introduction of open interfaces between the planes, as previously introduced in Sec. 2.1.1. This means also to determine which planes are deployed as software and which are still based on hardware equipment. For the interface between control and user plane, i.e., Southbound API, the OpenFlow protocol [13] has been established as a defacto standard. For the interface between application and control plane, i.e., Northbound API, no standard has emerged so far. The realization of the separation based on open interfaces usually requires an extensive re-engineering of the existing integrated solution, but it has the major advantage that subsequent steps on the path are significantly easier to achieve as they consequently require less engineering effort. We evaluate the latest OF switch API capabilities, i.e. OF version 1.5, to provide each function category.

Control Plane Related Functions

Generally for control plane only related functions, it is relatively straightforward to integrate and comply with the OF controller framework. Starting with the control signaling function, a module is needed to be added to the OF controller which performs the signaling management. We could point out that the resources manage-

ment logic could be mapped to the centralized switching module which is already a core function in any OF controller. The existing switching module would need to be slightly modified to include user profiles and policies and to utilize these in the resource management decisions.

Data Plane Related Functions

Intuitively, data plane functions are highly dependent on the data traffic and accordingly would exploit more the OF switch and its traits. First by looking at the data plane Forwarding Rules and data plane Forwarding, both are main operations of a basic OF switch and they impose no additional effort. However, both SGW and PGW use GTP tunneling protocol to transport the signaling and data traffic. An OF switch running even the latest OF specification, version 1.5 [13], is limited to only layer 2 and layer 3 matching, TCP and UDP port fields and generic tunneling protocols such as MPLS and GRE. However, it is not possible currently to realize SGW and PGW data forwarding with GTP header matching.

We use this finding to propose two different frameworks in order to realize functions such as GTP via SDN, illustrated in Fig. 2.5. The first framework as the more convenient proposal would be to enhance the OF switch to include certain additional functions, as in this case a GTP encapsulation and decapsulation function customized in hardware, as shown in Fig. 2.5a. We call it an OF SDN+ switch. This approach is needed for functions that cannot be provided currently by the OF specification or for which an implementation at the switch is beneficial for performance reasons. In this approach a hardware implementation in the SDN+ switch is required, which implies that its flexibility is quite limited.

Fig. 2.5b illustrates the second alternative framework where we propose supplying each switching element with a programmable platform that enables functions to be deployed on it. Such platform should enable the software functions to interact with the data plane and hence extend the OF switch capabilities. This is what we call a programmable OF SDN+ programmable switch, which goes into the direction of bare metal switches that offer the opportunity to run different network operating systems on the same network device [30]. The advantage of such element is the increased flexibility in extending the basic OF switch functionality.

Next to GTP, we have evaluated OF towards QoS support based on the Packet Filtering and Classification function. This would be quite relevant to realize a dedicated

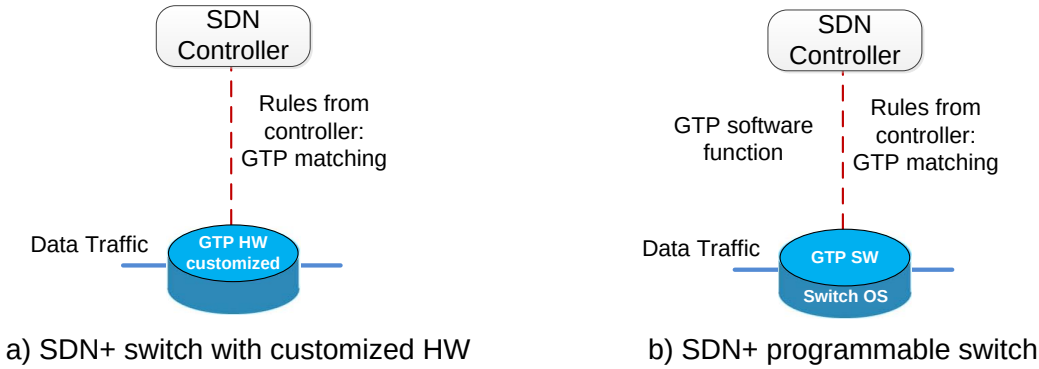


Figure 2.5: SDN frameworks for the GTP tunneling function.

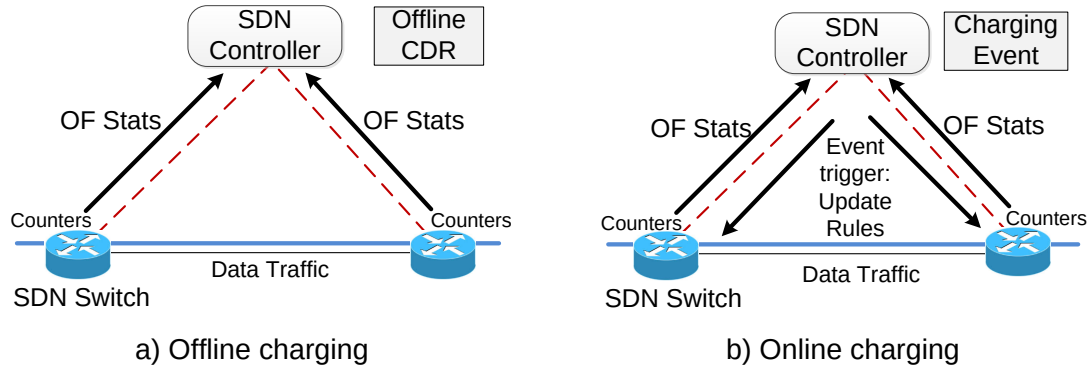


Figure 2.6: SDN framework for the charging control function.

bearer for LTE. Packet filtering can be realized via matching rules which correspond to the service data flow filters specified in [31]. The filters are matched through the IP five tuple: (source IP address, destination IP address, source port, destination port, protocol ID). We could observe that QoS support is provided starting from OF version 1.0. However, quality guarantees such as guaranteed bit rates and priority classes are switch dependent. It depends on the scheduling configurations that the OF switch can offer such as priority queues or weighted fair queuing with guaranteed minimum bit rates.

Regarding the Charging function, we propose using the OF counters and statistics exchanged between the OF switch and controller to realize an SDN charging framework as shown in Fig. 2.6. In case of offline charging, an additional module is needed to collect the Charging Data Records (CDR) based on OF counters and statistics as illustrated in Fig. 2.6a. It becomes more complex for online charging. As the OF switch keeps no state for the forwarded data flows, it is not possible to allocate

charging events on a basic OF switch. Fig. 2.6b illustrates one possible deployment in the OF controller. An alternative solution would be to include the charging function either in a hardware customized SDN+ switch or programmable SDN+ switch.

2.3 Summary and Discussion

The above analysis shows several challenges that need to be addressed by the operator to ensure a successful migration from the legacy mobile core architecture towards the new SDN and NFV concepts. The new concepts introduce additional components and requirements that have to be considered on top of the existing requirements by the current architecture and standard. For instance, the dimensioning of the mobile core network is required to include data centers and SDN switches as well. Further, the operation and management of the mobile core is extended with orchestrators and SDN controllers. These additional challenges induce some extra design and operation effort on the operator. However, the expected potential opportunities of SDN and NFV can be rewarding. That is the reason why it is very important to identify and narrow down the newly emerging challenges.

These new challenges can be addressed differently according to the design requirements and objectives. Operators can decide on various design factors that are of interest, which can differ from one operator to the other. For example, one operator could focus on the cost factors, e.g., decrease the number of data centers or aggregate shared resources to save costs. Another operator could target the network service factor by extending the network capabilities through NFV and SDN, e.g., improve the network support for traffic dynamics, even if it adds additional cost. It becomes compelling to observe the consequences and dependencies of the design choices on the new core network architecture and how SDN and NFV can contribute to the next generation 5G mobile network.

Chapter 3

Design Models for an SDN and NFV Mobile Core Network

In this chapter, we discuss more in detail the next generation mobile core network architecture based on SDN and NFV. Additionally, we analyze the impact of SDN and NFV concepts, introduced in Chapter 2, on the data plane traffic of the mobile core network. The implications of deploying both concepts on the network design is evaluated through network optimization models, which provide mobile operators with insights on how to adopt these new concepts. The proposed models aim at providing operators with an optimal network design, which entails which parts of the mobile core network should be migrated to an SDN deployment or alternatively migrated to an NFV deployment. The models also take performance requirements into consideration, where the output network design operates under constraints on the network performance, e.g., data traffic latency. The models also explore the cost comparison between an SDN or an NFV mobile core network design, e.g., in terms of network traffic load or energy consumption cost. This chapter is based on our work presented in [32] and [33].

3.1 Introduction and Motivation

As previously discussed in Chapter 2, Sec. 2.2, SDN and NFV bring different realizations for the the mobile core network deployment. Mobile operators have the opportunity to exploit both new concepts, namely SDN and NFV, to address the current challenges in their core networks, e.g., traffic changes and high induced cost. SDN introduces a network function split, where control plane functions are appended to

a logically centralized controller that could be deployed in a data center platform, while data plane functions are realized by SDN+ switches at the transport network. On the other hand, NFV proposes the operation of the mobile network functions as software instances on commodity hardware or data centers (DC).

We can observe that having the two deployment options opposes a design challenge for mobile operators which requires a thorough analysis to the advantages as well as overhead of both SDN and NFV core network architectures. For instance, NFV offers network function consolidation in order to reduce the network cost, however, this consolidation might impact the network performance for example in terms of traffic latency. SDN offers decoupled control and data planes in order to increase network programmability and flexibility, however, SDN might also impose an overhead in terms of additional network signaling and control. Hence, we could observe trade-offs between SDN and NFV where it is not yet clear which concept brings more advantages for the migration towards the next generation core network.

First, we present a qualitative comparison between SDN and NFV mobile core network architectures. The cost and overhead of both architectures is analyzed with the focus on the high volume data plane traffic that is typical in mobile core networks. Second, we propose optimization models that provide a quantitative comparison between both network designs in order to provide operators with more insights on how to deploy both SDN and NFV concepts. Since each concept has its own benefits and drawbacks in terms of cost and network performance, a network design based solely on one of the concepts might not be optimal. We argue that within a large and wide spread mobile core network, there is a high potential to combine both deployments, as one network part could adopt the NFV concept, the other part could deploy the SDN concepts, which we call the Function Placement Problem (FPP).

For this purpose, two optimization models are proposed in this chapter that evaluate the functions placement problem. The first optimization model minimizes the network design cost against several parameters such as data plane latency, number and placement of potential data centers as well as SDN control overhead. The second model resolves the function placement problem for time-varying traffic demand patterns, which provides insights on how to achieve cost reduction in terms of energy savings given performance requirements in terms of data plane latency.

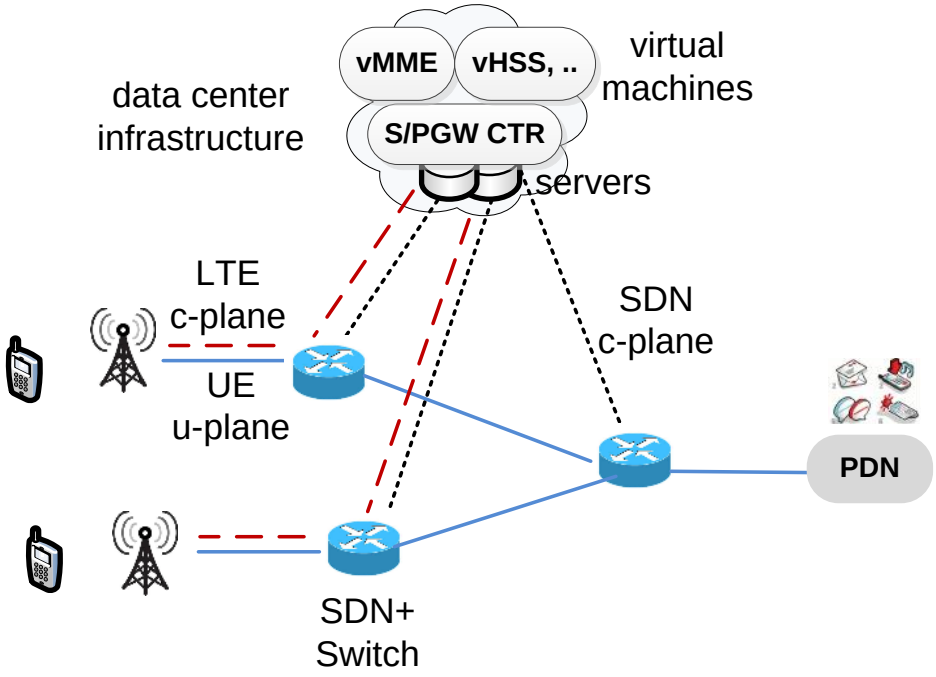


Figure 3.1: SDN mobile core network architecture design.

3.2 SDN and NFV Core Network Architectures

For a successful migration of the legacy mobile networks towards SDN and NFV, there are two key building blocks; the mobile network functions themselves, i.e., their realization as virtual network functions or as SDN controllers and switches, in addition to the network elements needed for their orchestration. A joint orchestration is required for the mobile network functions as well as the infrastructure, in order to ensure a seamless mobile network operation. According to the analysis presented in Chapter 2, the technology migration of the legacy LTE mobile core network architecture can take two paths; either SDN or NFV

3.2.1 SDN Mobile Core Network Functions Design

Considering an SDN based architecture, the control plane mobile core functions, e.g., MME and HSS, run as Virtual Network Functions (VNF) while the gateway functions, i.e., SGW and PGW, are decoupled into SDN controllers (S/PGW CTR) and special purpose SDN+ switches, as shown in Fig. 3.1. The VNFs and controllers

run on virtual machines that are deployed on the physical servers of the data center infrastructure. The SDN controllers, deployed at the data center infrastructure, configure the SDN+ switches which handle the data plane user traffic (UE u-plane). The controllers implement the control plane of the core network gateway functions. Thus, the SDN controllers are required to handle the LTE control plane signaling procedures (LTE c-plane) which are defined by the 3GPP standard, i.e., exchange of signaling messages with the radio access network in order to support the user's attachment to the mobile network or user's mobility. According to the signaling procedures, the controllers are responsible to configure the data plane, i.e., SDN+ switches, via the SDN API (SDN c-plane) used by the operator.

Additionally, the controllers need to implement the specific functions of mobile core gateways as discussed in detail in Sec. 2.2. Controllers are required to collect the data usage of each user from the data plane switches for the purpose of charging and accounting. On the other hand, the SDN+ switches implement the gateway data plane functions. One important data plane function needed at the SDN+ switches is GTP tunneling which is used to identify data plane traffic of users. The SDN+ switches monitor the data plane statistics for charging and accounting. Additionally, the SDN+ switches need to support the configuration of quality of service classes that can be assigned to users.

A key advantage for an SDN based architecture is that it allows more flexibility and offers a higher degree of freedom for the data plane as it is deployed on distributed elements. Keeping the data traffic on distributed dedicated nodes would have a major influence on the performance. In addition, keeping the centralized control plane maintains the global network view, which contributes to better network control in the mobile core network. Nevertheless, data plane rules exchange on the SDN control plane could introduce an additional overhead that needs to be minimized. Additionally, it could also create a bottleneck at the centralized control which would have a major influence on the whole core performance. It has to be restricted to relatively strict delay requirements to meet LTE core network standards.

3.2.2 NFV Mobile Core Network Functions Design

In case of an NFV based deployment, as illustrated in Fig. 3.2, the control plane mobile core functions as well as the gateway functions, i.e, SGW and PGW, run as VNFs (vS/PGW) on commodity hardware at data centers. This means that the gateway's

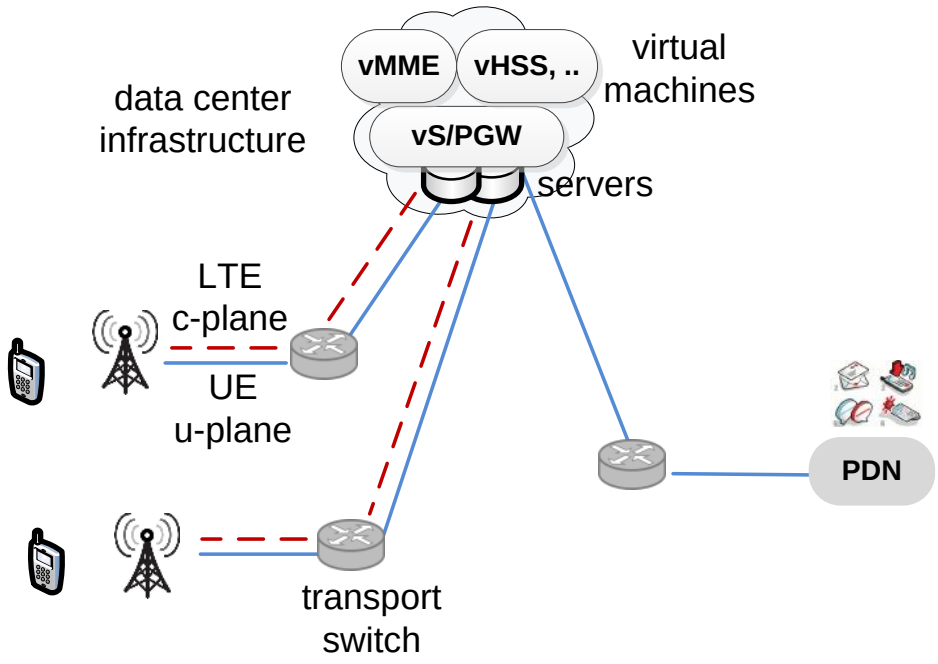


Figure 3.2: NFV mobile core network architecture design.

control plane as well as the data plane processing is running on commodity servers in the cloud. The data plane processing on commodity servers can be accelerated by solutions such as Intel DPDK [17]. Hence, the legacy core network hardware would be replaced by simple forwarding switches, i.e., transport switches, that forward both the data plane and control plane traffic between the radio access network, the data centers and the external network, as illustrated in Fig. 3.2. Note that in this architecture, all mobile core network functions are migrated to software running on commodity servers and are fully independent from hardware, i.e., functions which handle control plane only, e.g, MME, and functions that handle both data as well as control plane, e.g., SGW and PGW. This implies that there is no processing, i.e., function, implemented on the forwarding switches of the mobile core network.

The high volume cloud infrastructure brings several advantages to traditional mobile networks. First, the cost savings are quite notable compared to proprietary hardware components. A second advantage is the flexible dimensioning of mobile core functions, where a dynamic migration of the virtual components allows for better resource utilization and is a key factor in case of failures or disasters. Another

advantage is flexible upgrades by allocating more resources or adding a new functionality to the mobile core network.

However, such advantages are bounded by the cloud infrastructure domain and a larger scale flexibility covering the whole core network is not yet exploited in this architecture. Considering a use case of a service that requires a core node to be deployed in a rather nearer geographical location towards the users, this requirement could rise due to performance constraints such as the end-to end delay or due to on-demand or time-based services. The cloud infrastructure performance is another critical factor taking into account the high frequency of the control plane operations taking place at the SGW and PGW in addition to the large volume data traffic needed to be managed, especially at the PGW.

3.2.3 Orchestration Elements

There are multiple orchestration and control elements that are needed to manage and coordinate all of the platforms and network functions, i.e., data centers infrastructure, virtual network functions, SDN controllers as well as switches, introduced by the new architectures. According to ETSI NFV-MANO [3] shown in Fig. 3.3, a Virtual Infrastructure Manager (VIM) is required to operate the cloud infrastructure. The VIM is responsible for assigning sufficient resources to the virtual function instances from the cloud platform's pool of resources, which include computing, storage and networking. The virtual function instances need to run on resources so they can achieve a performance comparable to conventional hardware integrated solutions. Additionally, the VIM handles the synchronization and migration of the virtual functions in order to adapt to network requirements or dynamics.

The VIM is also required to setup the internal virtual connectivity within the data center to connect the virtual network functions. A virtual network function can be instantiated on multiple physical hosts. Nevertheless, it is abstracted as a single VNF. Alternatively, a decoupled network function can be distributed among different physical hosts and the appropriate connectivity, i.e., function chain, needs to be established. Such network mapping has to be established and maintained by the VIM such that the network function's purpose is preserved. Needless to say that the orchestration of the virtual infrastructure adds an additional dimension of complexity to operation and management. Additionally, a Virtual Network Function Manager (VNFM), is needed to maintain the life cycle management of the VNF instances.

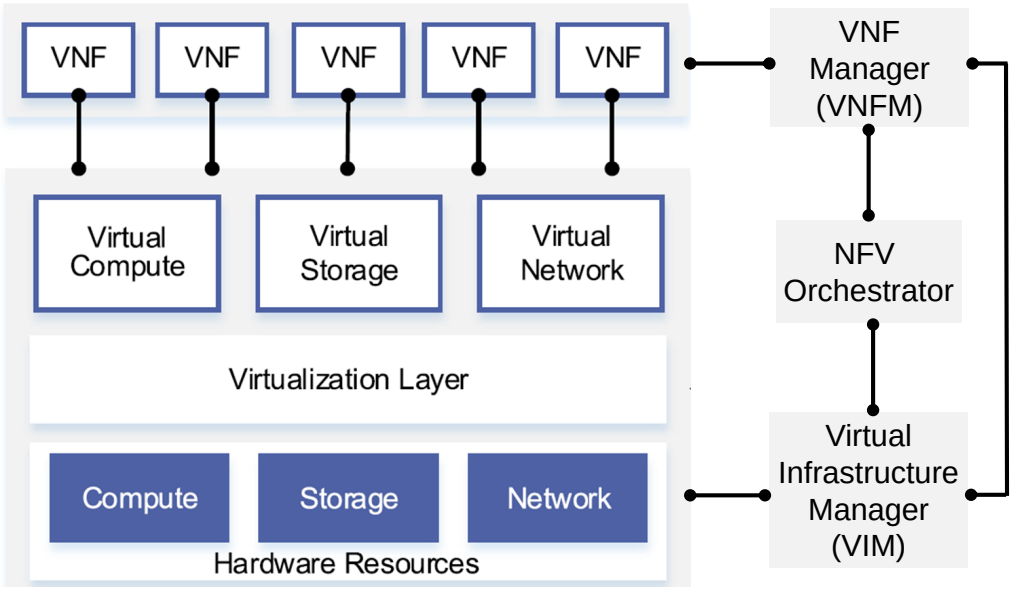


Figure 3.3: ETSI NFV MANO orchestration elements [3].

The VNFM is responsible to instantiate, configure and terminate VNF instances. It is also required to monitor the state and performance of the VNF instances, where it should handle error or fault events and perform software updates accordingly. In summary, the VNFM plays the role of overall coordination and configuration of the virtual function instances.

The NFV orchestrator located at the Network Operation Center of the operator can act as the interface between the non cloud infrastructure and VNF managers. It is required to delegate the network changes needed by each manager to keep an intact network state, e.g., in case the VIM migrates a cloud-based gateway to a new data center, the VNFM has to be notified to keep its management role of the VNF instances. The network operation center also contains the central logic which is responsible for dimensioning the network and deciding on network changes. Network changes are required for example for adding new sites or network elements, balancing the network load or saving energy costs by shutting down parts of the network including data centers. The network operation center delivers the operator requirements which are translated and enforced by both management components.

These result of the proposed optimization models in this chapter act as an input to the above described orchestration elements. The resulting network design, including the resources and locations of the mobile core VNFs and SDN controllers,

is derived and passed to the operator's orchestrator, which consequently communicates with the VIM and VNFM elements to start instantiating and setting up the virtual network functions as well as the controllers.

3.3 SDN and NFV Mobile Core Network Architecture State-of-the-Art

The application of SDN and NFV for the LTE mobile core network architecture has been in the focus of several investigations in the state-of-the-art. In this review, we summarize related work that considers deployment architectures or implementation oriented solutions for SDN or NFV mobile core networks.

Considering SDN, Softcell [34], MobileFlow [35], SAMA [36] and SoftMoW [37] apply the concept of SDN on the mobile core network by replacing the network functions with SDN controllers and switches that are used to interconnect between the RAN and external packet networks. The above proposals fold out the architecture of mobile core networks with centralized SDN control. [38] presents a qualitative discussion to the advantages and drawbacks of using SDN for mobile networks. This work highlights the advantages of having centralized network control in terms of traffic shaping and resilience. The authors in [39] present an SDN core network architecture with the focus on extensions to the OpenFlow protocol to implement GTP tunneling protocol to encapsulate users' traffic in the core network. SDMA [40] and TrafficJam [41] are proposals for a core network architecture based on SDN with a focus on user mobility management using OpenFlow. Both argue that an SDN mobility management can improve the core network support for mobile users due to the programmability provided by SDN on run time. Another direction is presented in [42] where the authors focus on the state, e.g., user data tunnels and charging profiles, that needs to be collected and exchanged between SDN controllers that implement control functions of the mobile core network.

A second group of proposals has investigated an NFV architecture for the mobile core network. The authors of [43] and [44] discuss a core network architecture that is fully comprised of virtual network functions and deployed on a cloud infrastructure, i.e., data centers. These proposal present the advantages of operating a virtual mobile core network in terms of resources efficiency and scalability. The work in [45] presents the concept of Software as a Service (SaaS) for a virtual core network, where

mobile virtual network functions are deployed as a service on-demand. The authors of [46] exploit the concepts of NFV and cloud computing to present a virtual core network that follows mobile users as they move. This work argues that this solution can improve the mobile network latency experienced by the users. Furthermore, the work in MOCA [47] and SMORE [48] presents an NFV core architecture that runs alongside a standard legacy core network. The NFV core network in these proposals is used for offloading purposes in case the legacy core network is overloaded. These solutions aim at providing a secondary virtual core network that is responsible for additional data traffic in overloaded situations.

We can observe that all proposed mobile core network architectures in the state-of-the-art literature consider either a deployment solely based on SDN or NFV. However, as we have presented in our previous work in [7], an SDN architecture can induce a higher cost due to the additional SDN control plane, while an NFV architecture can violate the network latency requirements due to the consolidation of VNFs in data centers. In this thesis, we explore an architecture that includes both SDN and NFV, where part of the network is selectively operated with SDN and the other part is comprised of VNFs. A combined deployment of SDN and NFV can exploit the advantages from both concepts and address their limitations.

3.4 Data Plane Analysis based on SDN and NFV

In this section, the impact and performance of both SDN and NFV on the mobile core data plane is analyzed and measured through prototype implementations.

3.4.1 Data Plane Paths

The data plane path within the mobile core depends on the operator's decision for the realization of both the SGW and PGW functions. In case of using SDN, as shown in Fig. 3.4, the legacy hardware functions would be replaced with the SDN+ switches which are controlled by the controllers residing in the cloud. This means that the data plane itself would follow the same function chains as the legacy network, i.e, between the SDN+ switches. It also means that the data plane latency depends only on the locations of the SDN+ switches and is decoupled from the location of the data center infrastructure. The data plane traffic in mobile networks can be modeled as uni directional function chains, i.e., uplink or downlink.

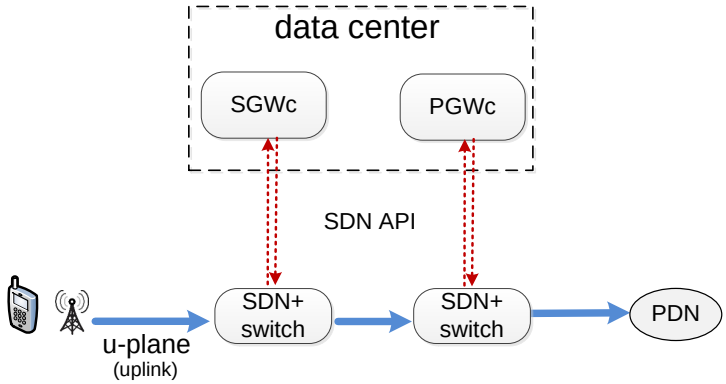


Figure 3.4: SDN data plane traffic as well as as SDN control plane paths.

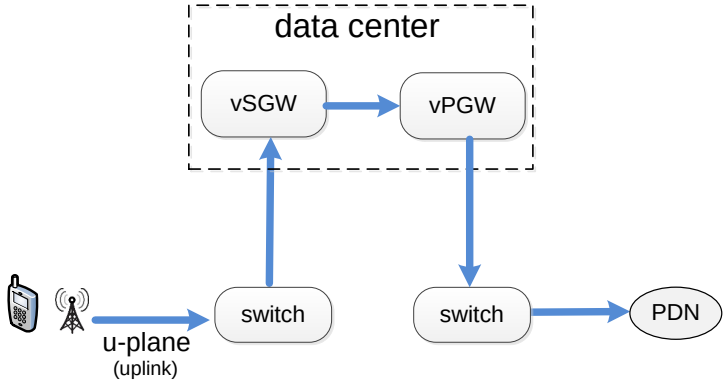


Figure 3.5: NFV data plane traffic path.

On the other hand, following the concept of NFV, the SGW and PGW functions are moved to the cloud. The legacy functions are replaced by simple forwarding transport switches, as shown in Fig. 3.5, which transport the data plane traffic towards the data center infrastructure where the data plane processing is carried out by the software gateway functions. This means that the NFV architecture has an impact on the data plane latency as it changes the data plane function chains. The data plane function chains are extended by the links carrying the traffic back and forth between the transport switches and the data centers. Hence, the data plane latency becomes dependent on the data center locations.

Whereas introducing NFV to the mobile network has lots of advantages as previously mentioned, it also requires transporting the whole network data traffic to the operator's data centers which imposes additional load on the transport network. Furthermore, an additional data plane delay is expected, which requires a thorough planning of the data centers location within the mobile network. In the same way,

introducing SDN adds an extra control plane which is not present in the standard architecture, that similarly imposes additional transport network load overhead, depending on the SDN control volume. In order to investigate such additional overhead, we focus on the two main functions of the LTE mobile core network which involve both control and data plane functions; the SGW and PGW. Our main aim is to investigate the influence of virtualizing or decomposing those two central functions on the data plane delay as well as the transport network load cost.

Our objective is to find the optimal realization and placement for these functions as well as the optimal placement for data centers that host the virtual network functions and SDN controllers. In this chapter, this model is formulated as an optimization problem which we call the Function Placement Problem (FPP).

3.4.2 Data Plane Latency Measurements

The data plane user traffic latency within the mobile core, i.e., traffic forwarded through the gateways between the access network and IP domains, could be simply defined as the sum of packet propagation delay T_{prop} on each link in addition to the packet processing delay T_{proc} at each node a packet traverses. The propagation delays depend on the link distances and transport technology between the gateways. In this section we aim at providing a quantitative comparison between the fully virtual and SDN gateways by measuring the data plane packet processing delays.

For our measurement, the first step was to develop prototype implementations, abstracted versions from the 3GPP standard, of the previously classified SGW and PGW functions, most importantly the GTP packet processing function. For the virtual gateway, we have developed a GTP packet processor in java. The virtual gateway has been hosted on one of our server nodes and was allocated 20 Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz virtual CPU cores with 128 GB RAM.

Regarding the SDN gateway evaluation, the SDN+ switch has been realized using an NEC PF5420 OpenFlow switch from our SDN testbed, running OF 1.3 firmware [49] and supporting a rate of 1 Gbps per physical port. It should be noted that since GTP headers are not included in the OF match fields specification, we have emulated the SDN+ switch packet processing as header bits modification using the OF Modify action. Additionally, the open source java floodlight controller [50] has been modified and extended to represent the gateway's control plane.

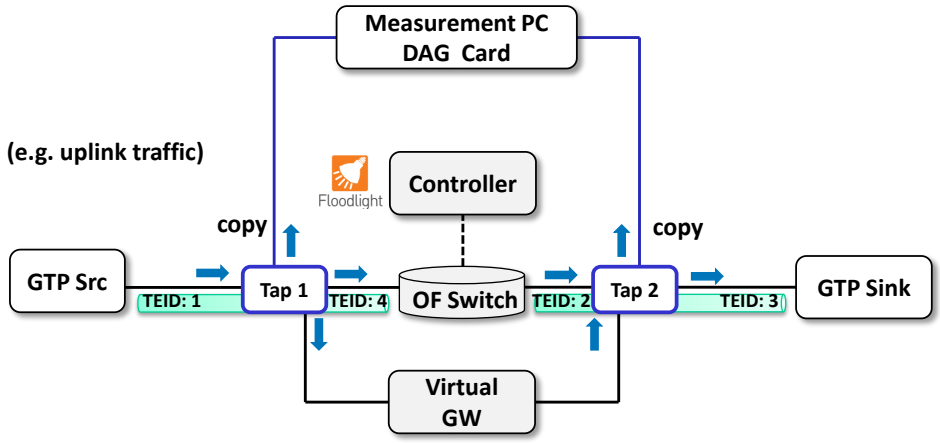


Figure 3.6: SDN and NFV data plane processing latency measurement setup.

Table 3.1: Measured mean packet processing latency.

Number of Tunnels	10	100	1 K	10 K
Data rate	1 Mbps	10 Mbps	100 Mbps	1 Gbps
Packet Rate	83 pckt/sec	830 pckt/sec	8300 pckt/sec	83000 pckt/sec
Virtual GW T_{proc}	62 μs	83 μs	109 μs	132 μs
SDN GW T_{proc}	15 μs	15 μs	15 μs	15 μs

Our measurement setup, illustrated in Fig. 3.6, consisted of a measurement PC with a 4 ports DAG card which has a time stamping precision in $nanoseconds$. Network taps were used to capture the traffic and forward duplicate copies to the measurement PC which then calculates the packet processing delay. The taps were directly connected within a small vicinity to the SDN switch and the server node running the software packet processor, to eliminate any propagation delay.

The packet processing delay, of both the virtual software function and SDN switch, has been measured against different values of data rate, number of packets per second and number of established tunnels. Each measurement run lasted for a duration of $10seconds$. The mean delay is calculated for each run, where the overall mean, shown in Table 3.1, is calculated with 95% confidence and at least 100 runs. A fixed packet size was considered of 1500 Bytes. Note that the SDN signaling as well as the tunnel establishment delay are not considered in this case as the main focus is on the packet processing delays. Hence, the delay measurements start only after all tunnels are established.

The SDN gateway, emulated by an SDN switch, shows a stable processing delay of $15\mu s$ over the range of parameters. It can be seen that the virtual gateway, running in software, experiences higher processing delays compared to the SDN gateway with a maximum of $132\mu s$ in case of 1 Gbps line rate. This is in line with common observations in switch design where packet processing in hardware is typically faster than software processing, certainly dependent on the available resources. We expect that the aforementioned processing delay difference between the SDN and virtual gateways would be minimized and improved by carrier-grade equipment. However, it can be noted that the data plane processing latency is in the order of microseconds. In case of a defined budget of several milliseconds to the data plane latency within the mobile core network, this would imply that a big part of the core network latency is coming from propagation rather than processing latency.

3.5 The Function Placement Problem (FPP)

The objective of the Function Placement Problem (FPP) is to define, model and solve the problem of applying virtualization, SDN decomposition or a combination of both concepts on the mobile core gateways. In other words, for each mobile core gateway, this problem resolves whether to virtualize and migrate all gateway functions to a data center or decompose this gateway as a controller residing in a data center and an SDN+ switch at the transport network, we call this the functions placement problem. It also addresses the optimal placement of the data centers, which host the virtual gateways as well as the SDN controllers. The controller placement problem has been previously addressed in [51] and [52], which focused on minimizing the SDN control delay and resilience provisioning, receptively. In our study, the optimal functions placement aims at minimizing the network load and satisfying the data plane delay budget. Note that this novel problem formulation could also be adapted to other network functions.

3.5.1 Problem Formulation

In this section, we formulate a model that evaluates the different gateway deployments in a mobile core network, where the aforementioned processing delay measurements are taken as input parameters. As previously discussed in Chapter 2, the core gateway network function could be either virtual or SDN based. The virtual gateway is moved to a data center where the gateway is replaced with a transport

switch which steers the traffic towards the data center. Alternatively, the gateway function could be realized as an enhanced SDN+ switch and an extended controller, where the controller is hosted by a data center. In our study, we target a first step migration scenario, so we keep the gateways' geographical locations unchanged. Additionally, we assume that data centers would be placed in a location where the operator has already an existing site to reduce the floor space cost, i.e., the data center is placed in a location where an operator has a gateway.

The problem is formulated as a path-flow model considering transport data plane demands between SGWs and PGWs. For each data plane traffic demand, there are two possible paths depending on the selected realization of the SGW and PGW functions either as SDN or NFV. Hence, for each traffic demand, the selected path results in the SGW and PGW functions placement and data center location. Demands are considered to be uniform, bi-directional and non-splittable.

Network Load Cost Objective

We define the network load as the bandwidth-latency product. For each data plane traffic path p for demand d with the location of data center c, the network load is pre-computed as the requested bandwidth by the demand r_d multiplied by the length of the path $lenPath_{c,p,d}$ as defined in (3.1). In this way, we could optimize the network resource allocation, i.e., bandwidth, in addition to the performance, i.e., latency in terms of path length, which would provide performance gains to the experience of the mobile users.

$$N_{c,p,d} = r_d lenPath_{c,p,d} \tag{3.1}$$

The problem's objective is to determine each gateway function placement and data center location, which minimizes the total transport network load:

$$\min \sum_{c \in C} \sum_{d \in D} \sum_{p \in P} \delta_{c,p,d} N_{c,p,d} \tag{3.2}$$

where the set C includes all possible locations of the K data centers. $\delta_{c,p,d}$ is a binary variable which denotes that a path p is chosen for demand d, with the location of data center c. The parameter $N_{c,p,d}$ is the pre-calculated load resulted from a combination of data center location, path and demand. The constraints of the function placement problem are given by:

Number of DCs

$$\sum_{c \in C} \delta_c = K \tag{3.3}$$

where (3.12) ensures that K data centers are chosen, and a binary variable δ_c denotes each selected data center c.

DC Selected Flag

$$\sum_{p \in P} \delta_{c,p,d} \leq \delta_c \quad \forall d \in D, c \in C \tag{3.4}$$

(3.13) reflects that if a data center c is chosen, a path p for demand d could be chosen using this data center c.

Data Path Selection

$$\sum_{c \in C} \sum_{p \in P} \delta_{c,p,d} = 1 \quad \forall d \in D \tag{3.5}$$

In case of K selected data centers, only one path using one data center for each demand is chosen, guaranteed by (3.14).

Data Latency Budget

$$\sum_{p \in P} \delta_{c,p,d} L_{c,p,d} \leq L_{budget} \quad \forall d \in D, c \in C \tag{3.6}$$

Finally, (3.15) ensures that the chosen path satisfies the data plane delay budget where parameter $L_{c,p,d}$ is the pre-calculated latency resulted from a combination of data center location, path and demand.

3.5.2 Results and Evaluation

For evaluation, a java framework was developed and we used Gurobi [53] as the optimization problem solver. It is known that it is quite difficult to set hands on a real deployment topology of an operator's mobile core. Therefore, we presumed a core gateways topology for the US as shown in Fig. 3.7, based on LTE coverage map in [54]. The topology consists of 4 clusters with 4 PGWs and 18 SGWs, resulting in 18

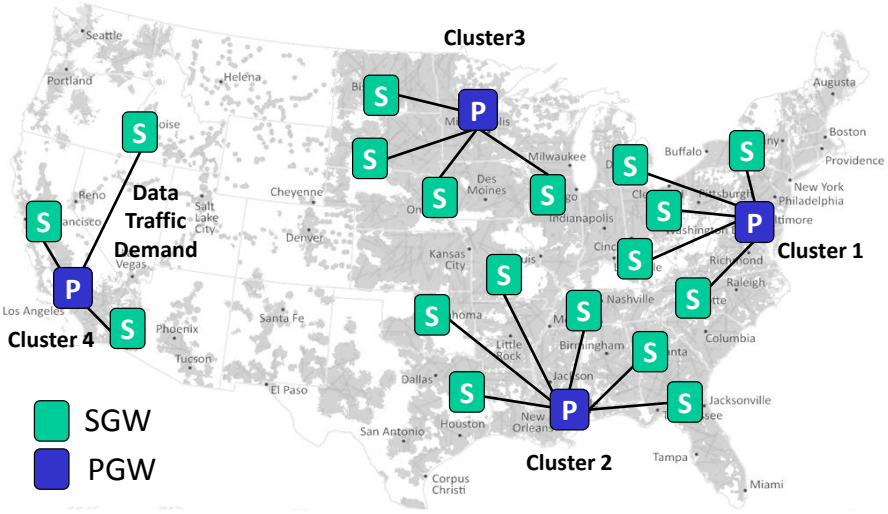


Figure 3.7: Mobile core network topology for USA, with respect to SGWs and PGWs, based on LTE coverage map in [54].

demands in total. A fully meshed infrastructure is assumed to be available between the gateways, i.e., a link could be established from any gateway to any other.

The data plane delay budget is defined for the different SGW-PGW interface realizations, either going through the data center or between SDN+ switch equipment. Each data center hosts the virtual gateways and the control functions of the SDN gateways, i.e., extended SDN controller, respectively. The processing delay input is taken from our previous measurement as a relative average of 10 μs for the gateway SDN+ switch and 100 μs for the virtual gateway transport switch. An underlying optical transport network has been considered which results in propagation delays depending on the distances between the functions' location. In this chapter, constraints on the data center available resources or restricted locations are not considered, hence, the focus is to study the impact on the high volume data plane traffic of the mobile core network which induces an important part of the core network operation.

There are two sources of traffic that comprise the network load depending on the functions placement. First is the data traffic to be transported between the SGW and PGW. Second, the SDN control traffic that is added in case of SDN deployment. This SDN control volume is pretty much dependent on the protocol adopted by the operator and the customization added to implement the mobile core functions as for example tunneling or charging. The SDN control volume depends on the LTE

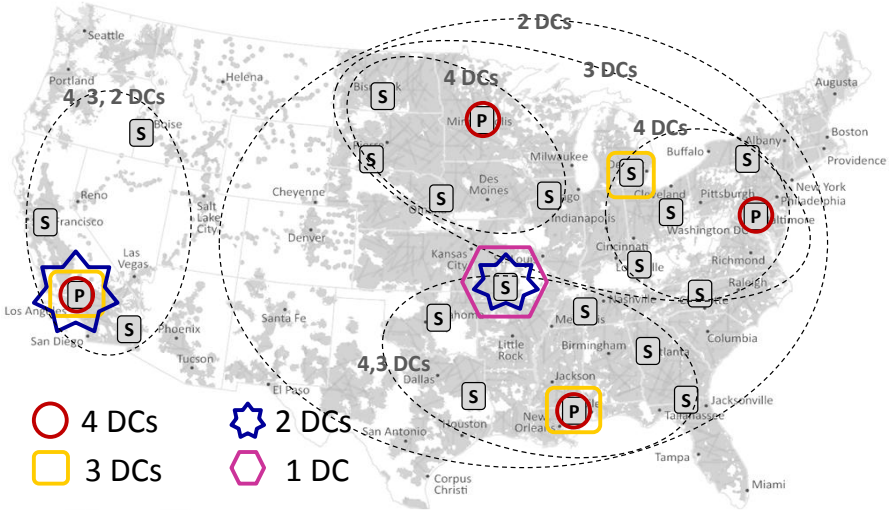


Figure 3.8: Data centers location at number of available data centers $K = 1, 2, 3, 4$.

signaling that an operator sees within its core network as well. Hence, we have investigated the impact of having different SDN control profiles and we denote it as a percentage of the data volume transported for each gateway. For example, an SDN control profile of 10% means SDN loads the link from the data center to the gateway SDN+ switch by a volume that is one tenth the data volume transported by such gateway. It would be the task of the operator to identify the SDN control volume in its network.

The optimization problem is a deterministic problem, therefore multiple runs are not needed. The network load $N_{c,p,d}$ and path latency $L_{c,p,d}$ were pre-calculated for all possible combinations of data center, path and demand, which decreases the complexity and makes the run time quite fast, i.e., in the resolution of $milliseconds$ per iteration.

3.5.2.1 Full Virtual (NFV) Deployment

The maximum propagation delay between an SGW and its respective PGW in the presumed US core topology is $5.128ms$, see Fig. 3.7 in cluster 4 between Idaho and Los Angeles. Considering an additional processing delay of $100\mu s$, the data plane delay budget has been set to $5.3ms$. First question; *how many data centers are needed to fully virtualize all gateways and move them to data centers for this given topology under the* $5.3ms$ *data plane delay budget?*

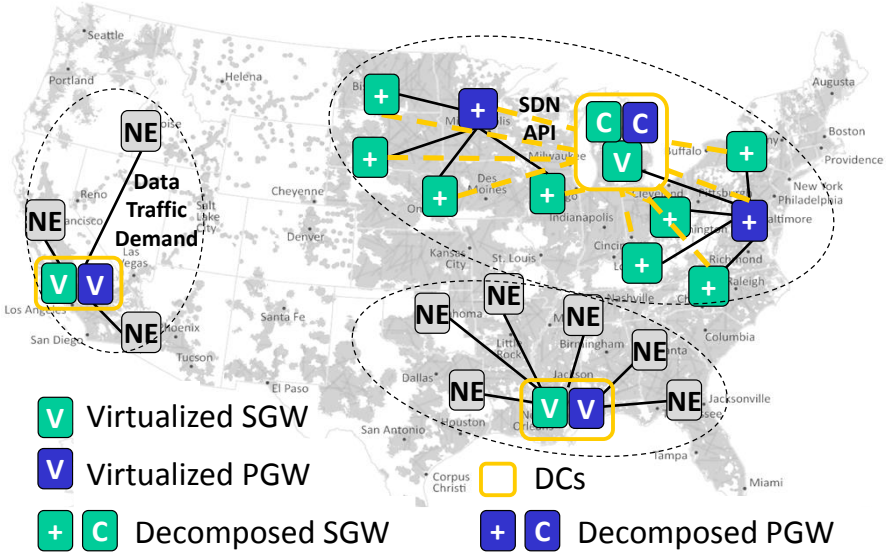


Figure 3.9: Functions placement at 3 data centers, with SDN control traffic = 10% of data plane traffic and under 5.3 ms delay budget.

The problem's solution determines that at least 4 data centers are required to support a full migration to virtual gateways under a data plane delay budget of $5.3ms$, where a data center is located at the PGW of each cluster as shown in Fig. 3.8. The total network load of the original presumed gateway topology has been calculated to be used as a reference for the load overhead. In this case, the total network load with 4 data centers is equal to the total network load in the original presumed gateway topology. In such tree structure at each cluster, placing the gateway functions at the tree root node would diminish any extra traffic transport overhead. It has also been noted that with a fewer number of data centers, it is infeasible to fully virtualize all gateways under such data plane delay constraint.

3.5.2.2 Combined SDN and NFV Deployment

Another conclusion that can be drawn from the above results is that, if we want to operate less than four data centers in this given topology and under a strict delay budget, SDN gateways are needed to keep a subset of the data traffic at the transport network. Hence, we solve again our functions placement problem with the possibility of having virtual or SDN gateways, which minimizes the total network load under the same data plane delay budget of $5.3ms$. The resulting data center placement with the respective network partitioning is illustrated in Fig. 3.8, ranging from

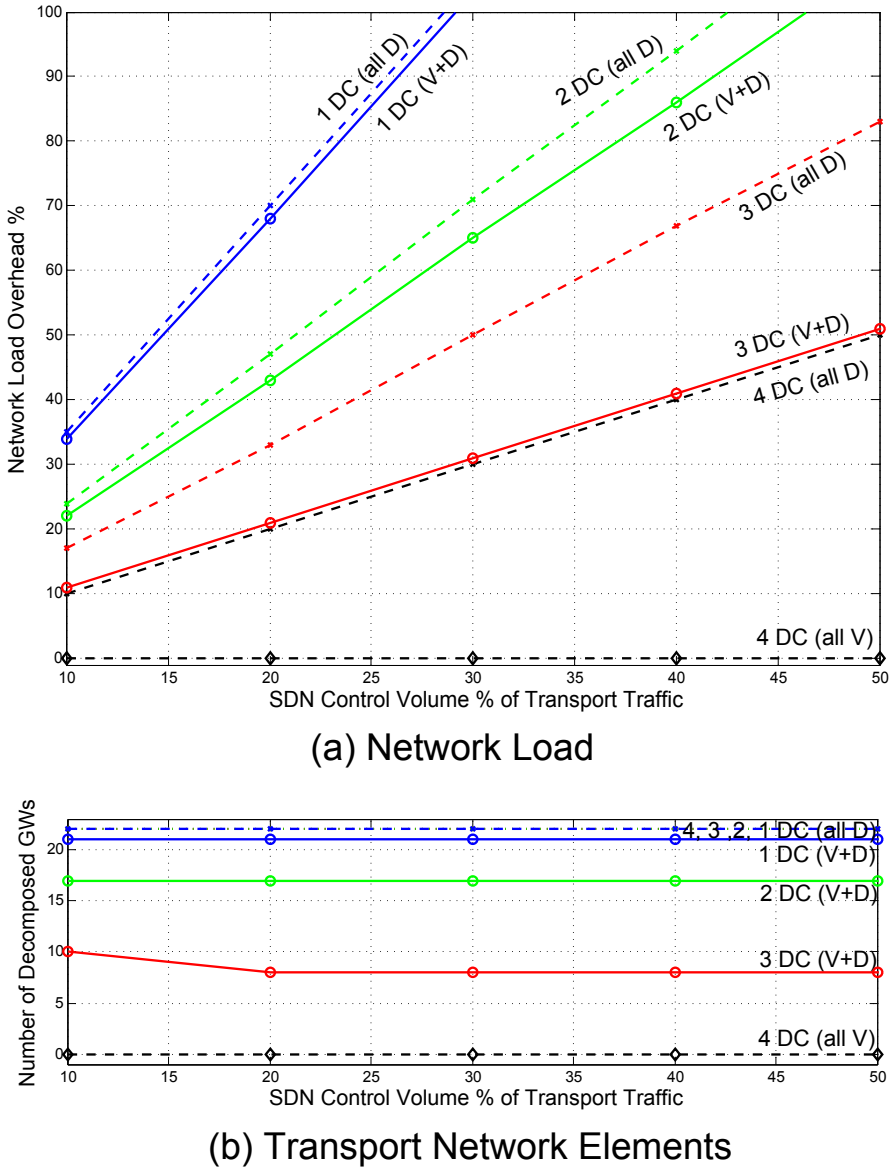


Figure 3.10: Network load overhead and number of SDN gateways, compared to number of available data centers K and SDN control volume, under data plane delay budget of 5.3 ms.

3 to 1 available data centers. A snapshot of the functions placement with a combined virtual and SDN deployment is shown in Fig. 3.9, having 3 data centers and an SDN control volume of 10% under $5.3ms$ delay budget. We also solve the problem for a full SDN gateway deployment, i.e., all gateways are SDN based, in order to have a comparison with the combined virtual and SDN gateway deployment. The total network load of the legacy core topology is considered as a reference for the resulting network load overhead.

3.5.2.3 Network Load Overhead

It is shown in Fig. 3.10a that the number of data centers represents an important factor in determining the network load overhead as the total network load increases significantly in case fewer data centers are deployed. For instance, at a 10% SDN control volume, the combined virtual and SDN gateways deployment with 3, 2 and 1 data centers show 11%, 21% and 34% load overhead, respectively. A single data center is showing a significant load overhead with a steeper incline as the SDN control volume increases, doubling the traffic inside the core network at an SDN control volume of 29%. This is due to the extra SDN control plane and the high number of SDN+ switches needed shown at Fig. 3.10b for a single data center.

Fig. 3.10a shows also that with the considered topology and input parameters, a combined virtual and SDN gateway deployment outperforms the deployment with only SDN gateways in all cases of K data centers, concerning network load overhead. Furthermore, the load overhead resulted by a combined deployment using 3 data centers approaches the full SDN gateways deployment using 4 data centers.

Additionally, the network load overhead gap observed between the combined deployment and the full SDN deployment is seen to be closing down as the number of available data centers decreases. This could be explained by Fig. 3.10b which shows that the number of needed SDN gateways is increasing while fewer data centers are used, approaching a full SDN deployment. In case of a single data center, 21 out of 22 gateways are realized through SDN.

3.5.2.4 Functions Placement

Fig. 3.10b shows that the number of SDN gateways, realized by SDN+ switches at the transport network, increases significantly with the availability of fewer data centers. This is to avoid the extra delays of transporting the traffic through the data center under such data plane delay budget of $5.3ms$. For instance at an SDN control volume of 10%, if an operator decides to deploy a combination of virtual and SDN gateways using 3 data centers instead of a full virtual deployment with 4 data centers, 45% of the transport network elements are required to be SDN gateways or enhanced SDN+ switches. In this case, a cost analysis is essential for the operator to evaluate the cost difference between the deployment of a data center or deploying enhanced SDN+ switches, which will be then the decisive factor.

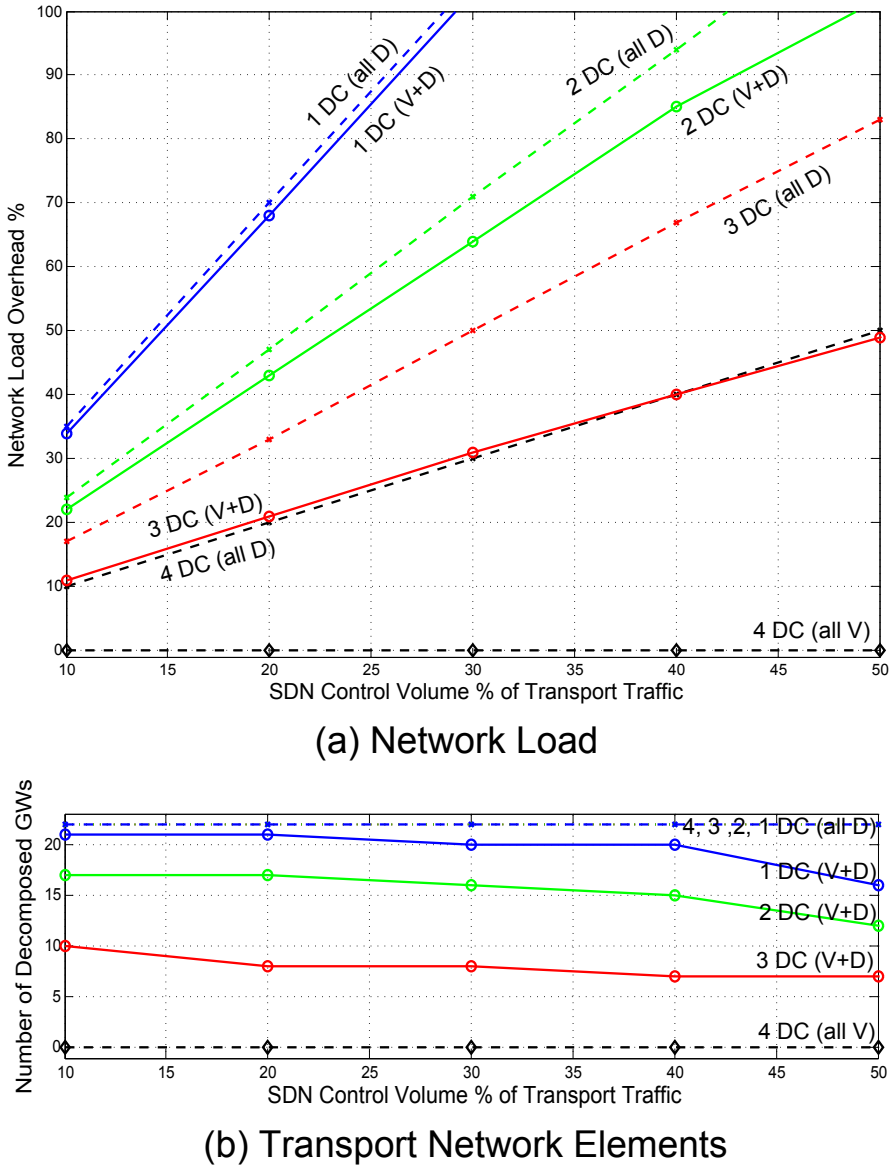


Figure 3.11: Network load overhead and number of SDN gateways, compared to number of available data centers K and SDN control volume, under data plane delay budget of 10 ms.

Regarding data centers placement, it is observed that each data center is placed in a central location towards the respective network elements within its selected network partition, as in the case of 3 data centers and an SDN control volume of 10%, shown in Fig. 3.9. The centrality of each data center in its respective network partition is an indication for the transport network load balancing and the responsiveness of the SDN control as well.

3.5.2.5 Delay Budget Relaxation

In the previous evaluation, the gateway functions have been placed under a strict data plane delay budget of $5.3ms$. We solve the problem again after relaxing the data plane delay budget to $10ms$ to observe the impact of relaxing the data plane delay constraint on particularly the functions placement and total network load. Note that higher delay budgets could be tolerated by some services, yet nowadays mobile networks attempt to offer less delays to improve the user satisfaction.

For a full virtual gateways deployment, it is still infeasible with fewer than 4 data centers, which again points out to the necessity of a combined virtual and SDN deployment for the case of using less than 4 data centers.

The network load overhead after relaxing the delay-budget to $10ms$ is quite similar to $5.3ms$ as shown in Fig. 3.11a. However, with a higher data plane delay tolerated, it is affordable to deploy more virtual gateways in case the SDN control volume increases, in order to keep a minimum network load, as illustrated in Fig. 3.11b.

3.6 Time-varying Function Placement Problem

Today's network operators are faced with the steadily increasing network traffic given the limited flexibility provided by currently deployed architectures. This lack of flexibility leads to an inefficient use of the available resources, which in turn leads to decreasing revenues for operators [55], when the changing dynamics of the user demands cannot be fully considered. This means that network operators let their network regularly undergo periods of over- and under-utilization. In this problem, we investigate how operators can exploit SDN and NFV to operate their network resources in a more fine granular way [56] and to apply dynamic network changes over time. Both concepts allow the operators to exploit more gains from operating the mobile network considering time-varying traffic patterns to achieve more efficient load balancing or energy savings. The objective of this problem is to find the optimal data center placement, hosting the virtual gateway components, which achieves a minimum transport network load while considering time-varying traffic pattern under a data plane delay budget. We provide further approaches to maximize power savings by adapting the data centers operation according to the traffic patterns and data center resources.

3.6.1 Traffic Pattern Observations and Generation

We first summarize recent related work showing measurements taken from mobile and wired networks. Based on observations of these papers, the traffic patterns used for the evaluation of our architecture are explained subsequently.

3.6.1.1 Traffic Measurements from Today's Networks

Many measurement papers [57, 58, 59, 60, 61] show traffic behavior that is correlated in time and space as well. The measurements are taken from 2G/3G/4G networks, for different vantage points, for up to four major U.S. carriers including UMTS networks, and for mega events.

In [57], an event based analysis of cellular traffic for the Super Bowl is provided, which shows that there are traffic peaks related to events, e.g., the Half-time Show or a Power Outage. In [58], the authors investigate the traffic proportion of companies, e.g. Google, Facebook, Akamai and Limelight. The traffic shows a time-dependent behavior, where Google has the highest proportion. [59] analyzed user activity at different locations in order to investigate users' mobility. [60] shows that traffic patterns for wired and wireless networks are affecting each other where more users utilize the mobile network during commuting time. Further, different points in space are showing a different but timely correlated traffic intensity. In [61], a correlation even for different applications can be concluded.

3.6.1.2 Traffic Pattern Generation

To quantify the effectiveness of our approach, traffic patterns for the mobile core demands have to be created. A demand is defined as the traffic flow between each sgw and its respective pgw, which is considered to be time-varying, bi-directional and non-splittable. Hence, traffic demand patterns could integrate a correlation in time and location of the corresponding gateway $sgw \in SGW$.

First, we explain how the traffic pattern for each city, whose demand is forwarded to one sgw, is determined. For this, the population and location of cities C, and the time-dependent behavior of the traffic intensity is taken into account. The intensity function

$$i(t) = intensity_t \tag{3.7}$$

takes the traffic intensity $intensity_t$ from a look-up table (as introduced in Table 3.2) according to the current time t. The current time t of a city corresponds to its current local time. For calculating the traffic based on the population for the current time t, the following function is used:

$$f(c,t) = i(t) \cdot p(c) \tag{3.8}$$

where $p(c)$ provides the population of city c. Using these functions, the traffic patterns integrate a correlation in time and location for each city.

The traffic at each sgw is the sum of all cities C connected to it. Thus, for each $sgw \in SGW$ the function

$$TR_{sgw}(t) = \sum_{C} f(c, time_{c,sgw}(t)) \cdot b_{c,sgw} \tag{3.9}$$

calculates the aggregated traffic at time t, where $time_{c,sgw}(t)$ is the local time of city c calculated depending on the local time of sgw, and $b_{c,sgw}$ is an indicator whether city c is connected to sgw or not. Note again that cities and gateways may belong to different time-zones, thus, the traffic intensity of a city c always depends on c's local time that has been calculated depending on the current time, i.e., the time zone of the sgw.

3.6.2 DC Placement and DC Power Saving Models

As previously discussed in Sec. 3.6.1, time-varying traffic patterns could be observed in today's mobile networks. Such variation in network traffic could be considered for dynamic network dimensioning to achieve higher gains for operators in terms of network utilization or power saving. We split the whole time frame into multiple time slots, each with varying traffic demands.

We introduce three models, where the first model aims at finding the optimal data center placement that minimizes the total transport network load under a data plane delay budget. By exploiting the traffic variation, the second and third models aim at achieving power savings by allowing fewer data centers in operation within the resulted available data center resources or using additional resources.

The network load is considered as a significant objective cost metric for mobile networks dimensioning as it reflects on traffic delay and cost imposed on operators. We define the total network load as:

$$\sum_{t \in T} \sum_{d \in D} r_{d,t} lenPath_{d,t} \tag{3.10}$$

where d is a demand between each SGW and PGW, which are considered to be time-varying, bi-directional and non-splittable. $r_{d,t}$ is the traffic volume of demand d at a time slot t, while $lenPath_{d,t}$ is the length of the path taken for a demand d at time slot t. A demand path within the core network is defined between an SGW transport switch, a data center and a PGW transport switch. Hence, chosen paths in fact determine the location of data centers and assigned demand to each DC, while data plane delay is defined as propagation delay on the path for each demand.

Note that data centers are assumed to be placed in a location where an operator already has an existing site to reduce the floor space cost, i.e., the data center is placed in a location where an operator has gateways. We also keep the gateways' geographical locations unchanged, i.e., replace conventional gateway with transport or SDN+ switches. All models are formulated as path-flow models.

3.6.3 DC Placement for All Time Slots (DCP-ATS)

Network Load Cost Objective over All Time Slots

The objective of this first model is to find the optimal DC placement with minimum transport network load under a given data plane delay budget, considering the traffic demands over all time slots as follows:

$$\min \sum_{c \in C} \sum_{d \in D} \sum_{t \in T} \delta_{c,d,t} N_{c,d,t} \tag{3.11}$$

where the set C includes all possible locations of the K data centers. The binary variable $\delta_{c,d,t}$ denotes that a data center c is chosen for demand d at time slot t. The parameter $N_{c,d,t}$ is the pre-calculated load resulted from a combination of data center location, demand and time slot. The constraints are given by:

Number of DCs

$$\sum_{c \in C} \delta_c = K \tag{3.12}$$

where (3.12) ensures that K data centers are chosen as δ_c is a binary variable to indicate a chosen data center.

DC Selected Flag at Time Slot T

$$\delta_{c,d,t} \leq \delta_c \quad \forall d \in D, c \in C, t \in T \tag{3.13}$$

Constraint (3.13) reflects that if a data center c is chosen, demand d at time slot t could be assigned to this data center c.

Data Path Selection at Time Slot T

$$\sum_{c \in C} \delta_{c,d,t} = 1 \quad \forall d \in D, t \in T \tag{3.14}$$

In case of K selected data centers, only one data center can be assigned for each demand, guaranteed by (3.14).

Data Latency Budget at Time Slot T

$$\delta_{c,d,t} L_{c,d,t} \leq L_{budget} \quad \forall d \in D, c \in C, t \in T \tag{3.15}$$

Finally, (3.15) ensures that the chosen path satisfies the data plane delay budget where parameter $L_{c,d,t}$ is the pre-calculated latency resulted from a combination of data center location, demand and time slot.

3.6.4 Power Saving at Each Time Slot within DC Resources (PS-ETS)

This model acts as a next step after solving DCP-ATS in Sec. 3.6.3 as it takes the resulted chosen DCs and their available resources as an input from the solution of the model 3.6.3. The objective of such model is again to minimize the total transport network load, while having the degree of freedom of operating a number of DCs less than K at each time slot, from the set of previously chosen DCs and within the available DC resources. This implies that among the deployed DCs and considering the traffic characteristics, operators would be able to minimize the power consumption of unutilized DCs, resulting in power and cost savings.

Network Load Cost Objective over Time Slot T

In this model, the objective is updated to find the optimal solution for each time slot as follows:

$$\min \sum_{c \in C_s} \sum_{d \in D} \delta_{c,d,t} N_{c,d,t} \quad \forall t \in T \tag{3.16}$$

DC Power Saving at Time Slot T

Additionally, constraint (3.12) is replaced with constraints 3.17, which allows for operating fewer DCs and (3.18) that ensures that one DC is at least in operation.

$$\sum_{c \in C_s} \delta_c \leq K \tag{3.17}$$

$$\sum_{c \in C_s} \delta_c \geq 1 \tag{3.18}$$

DC Available Resources

Note that for all expression the set of data centers C is replaced by C_s which contains the chosen DCs by model DCP-ATS in Sec. 3.6.3. Additional constraint (3.19) is needed to ensure that the assigned resources at each DC do not exceed the available resources $R_{d,t}$.

$$\sum_{d \in D} \sum_{t \in T} \delta_{c,d,t} R_{d,t} \leq R_c \quad \forall c \in C_s \tag{3.19}$$

3.6.5 Power Saving at Each Time Slot with Additional DC Resources (PS-ETS-AR)

This models is an extension for the previous PS-ETS model, where it provides a room of having additional resources at each DC which should not exceed the available resources multiplied by a factor P. It also considers the set of DCs as an input from the solution of model DCP-ATS in addition to applying constraints (3.17) and (3.18) as in the previous model. However, constraint (3.19) is substituted by (3.20) to allow additional resources under the boundary provided by $R_c * P$.

DC Additional Resources

$$\sum_{d \in D} \sum_{t \in T} \delta_{c,d,t} R_{d,t} \leq R_c * P \quad \forall c \in C_s \tag{3.20}$$

3.6.6 Results and Evaluation

For simulation, a java framework has been implemented with GUROBI [53] used as the optimization solver. We created a US mobile core gateway network shown in Fig. 4.5, based on LTE coverage map in [54] which correlates with the US population distribution [62]. The core gateways network consists of 4 PGWs and 18 SGWs, where a traffic demand exists between each SGW and the corresponding PGW in its cluster, resulting in a total of 18 demands. The core network is assumed to be fully meshed which implies that any gateway location could be chosen to deploy a DC with an available link to all other locations. The network load $N_{c,d,t}$ and path latency $L_{c,d,t}$ parameters are pre-calculated for all combinations of DC locations, demands and time slots. This decreases the optimization solving complexity and run time.

3.6.6.1 Traffic Patterns

Based on the definition in (3.9), a traffic pattern for each gateway is determined. Here, the indicator $b_{c,sgw}$ is set to 1 if a gateway sgw is the closest to a city c. The population and the geo-location information of all U.S. cities is taken from [63, 64]. Table 3.2 contains the daytime and the corresponding traffic intensity according to [60].

Fig. 3.12 shows the traffic patterns of 18 gateways, which consider the population and intensity based on the time during a day. The time zone is set to EDT (New York's local time). As SGW 2 is located at the east coast, and, thus is connected to cities such as New York, it has the highest traffic demand. SGW 16 is located at the west coast and serves cities such as Los Angeles, therefore, having the second highest demand. Between both gateways, we can also see the time shift according to the different time zones of the cities both gateways are serving. As SGW 12 is the most northern gateway that connects only cities with a small population, it has the lowest demand over time.

The daily time frame was split into 4 time slots according to the traffic distribution, where hours with minor variation traffic patterns were grouped into a time slot. The evaluation is done based on the averaged traffic patterns defined in time slots that are shown in Fig. 3.13.

Table 3.2: Look-up table that contains daytime and corresponding traffic intensity.

Time	0:00	1:00	2:00	3:00	4:00	5:00	6:00	7:00
Intensity	0.65	0.55	0.35	0.25	0.2	0.16	0.16	0.2
Time	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Intensity	0.79	0.79	0.85	0.86	0.4	0.55	0.65	0.75
Time	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00
Intensity	0.85	0.83	0.8	0.79	0.76	0.76	0.69	0.6

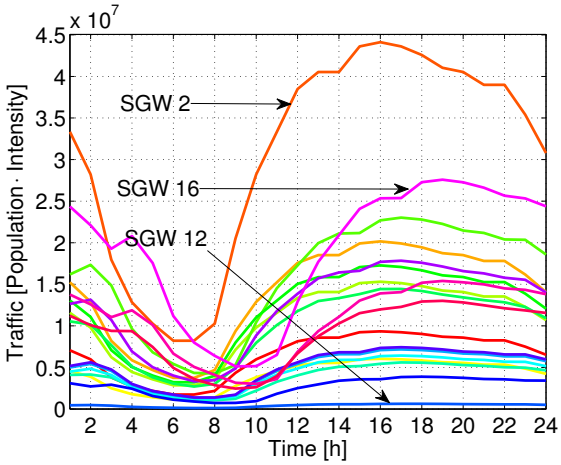


Figure 3.12: Daily traffic pattern.

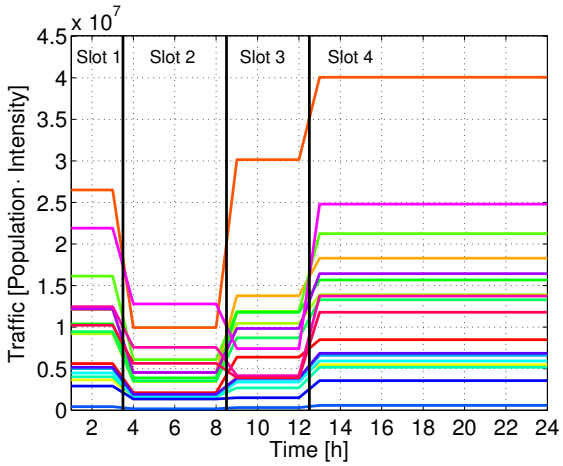


Figure 3.13: Pattern split in 4 time slots.

Table 3.3: Evaluation parameters and use-cases.

Model	DCP-ATS	PS-ETS	PS-ETS-AR	DCP-ATS
K DCs	3	3	3	2

3.6.6.2 Evaluation Parameters and Cases

In this evaluation, the transport network data plane delay budget L_{budget} has been relaxed to $15ms$. An underlying optical transport network has been considered which is used to calculate the data plane propagation delay for each path. The DC resources are depicted in traffic demand units. For evaluation, we have conducted a comparison between four cases shown in Table 3.3, in terms of the resulting DC required resources, DC utilization, transport network load and DC power consumption.

First, the problem has been solved for the DCP-ATS model which considers all 4 time slots in a day, with 3 DCs. The chosen DC locations and their resources have been taken as an input to solve the DC power saving models, namely PS-ETS and PS-ETS-AR, which exploit the operation of fewer data centers than 3 for each time slot, to achieve operational power savings. For the PS-ETS model, it is solved with the resulting available DC resources from the optimization of data center placement

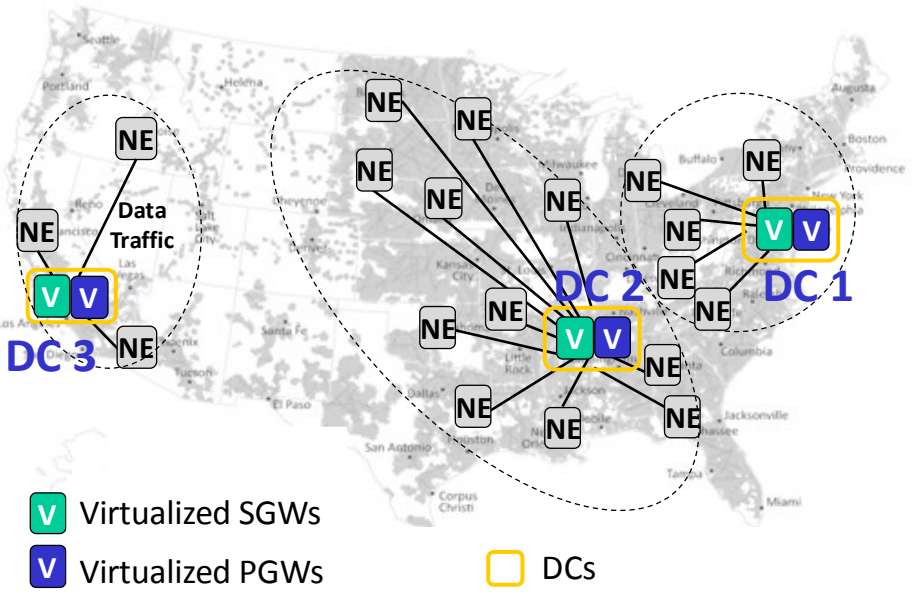


Figure 3.14: Function placement at number of available data centers $K = 3$.

over all 4 time slots (DCP-ATS). While for PS-ETS-AR model, it allows additional resources for each DC up to factor P, which has been set to 2, i.e., the additional acquired resources at each DC should not exceed the value of the existing resources or in other words that the total resources should not be doubled. The case with $K - 1$ data centers has been considered to compare the gains and drawbacks of the two proposed power saving approaches.

3.6.6.3 Data Center Location

The DC locations which achieve the minimum transport network load under the data plane delay budget of $15ms$ and with 3 DCs can be seen in Fig. 3.14. It could be noted that cluster 4 required its own DC to be able to handle its traffic under the $15ms$ data plane delay budget due to its remoteness from the other three clusters. However, cluster 1, 2 and 3 which are more adjacent could be handled by only 2 DCs. Another property to note is that in case a DC is handling only a single cluster, the DC is placed at the location of the PGW of such cluster as for example DC1 and DC3, as this achieves the minimum transport network load. On the other hand, the centrality of the DC is observed in case it handles more than one cluster as for example in the case of DC2.

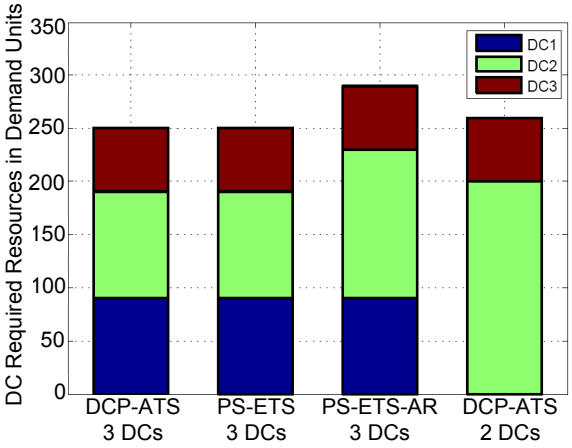


Figure 3.15: Required data center resources for the evaluation use-cases.

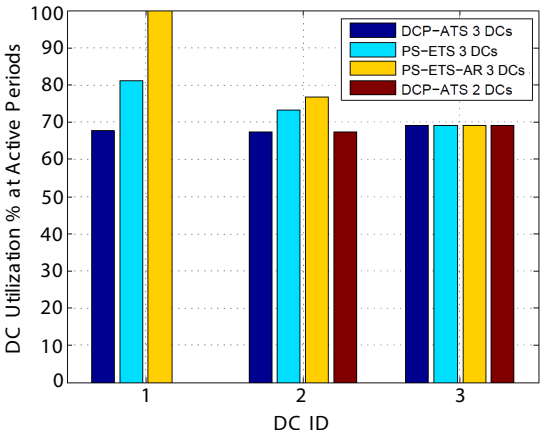


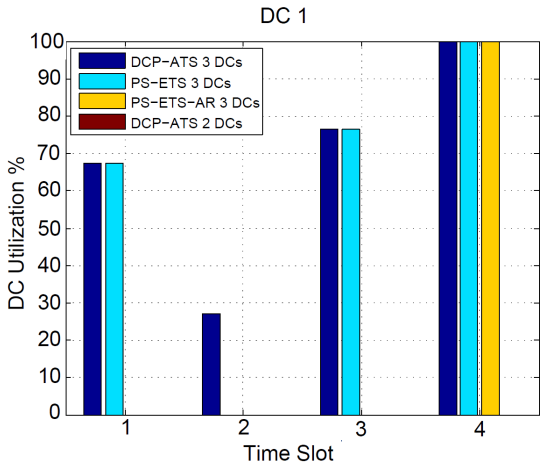
Figure 3.16: Data center utilization at daily active periods for the evaluation use-cases.

3.6.6.4 Data Center Resources

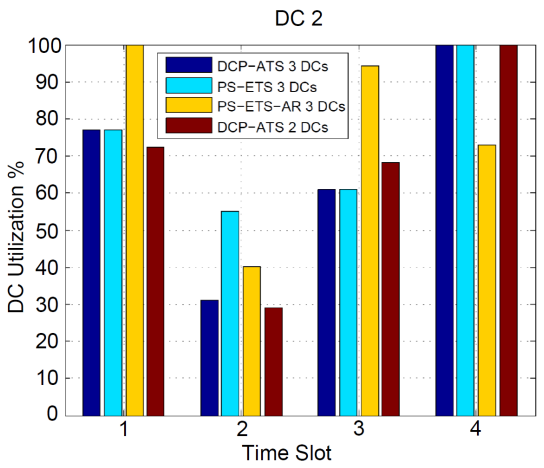
Fig. 3.15 shows the required resources at each DC for the 4 evaluation cases. Models DCP-ATS with 3 DCs and PS-ETS have equal required DC resources since PS-ETS is achieving power savings within the available DC resources from DCP-ATS. On the other hand, model PS-ETS-AR which allows for additional resources results in increasing the acquired resources of DC2 by 40% compared to models DCP-ATS with 3 DCs and PS-ETS. Additionally, it can be seen that in case DCP-ATS with 2 deployed DCs only, DC2 and DC3 are chosen due to the remoteness of cluster 4 and the centrality of DC2 towards clusters 1, 2 and 3. The resources required by DC2 in this case compared to DCP-ATS with 3 DCs are 100% more. In this case, it is the operator's decision whether to get an extra data center or increase the resources of a data center. This depends on the difference between the cost induced for deploying the DC itself compared to adding resources to it.

3.6.6.5 Daily Data Center Utilization

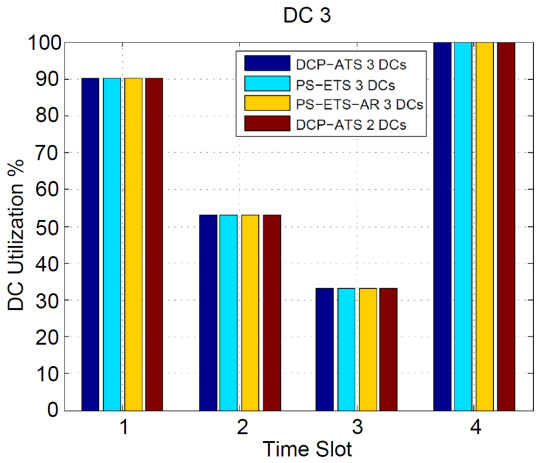
The data centers daily utilization considering the active operation periods is shown for each DC in Fig. 3.16. It shows the efficiency of the DC resources utilization which is impacted by the dynamic operation through considering the daily active operation periods. It can be noted that the model PS-ETS improves the DC utilization and operation efficiency while the model PS-ETS-AR shows the most efficient utilization of the DC resources.



(a) Utilization of data center 1.



(b) Utilization of data center 2.



(c) Utilization of data center 3.

Figure 3.17: Data center utilization at each time slot for the evaluation use-cases.

3.6.6.6 Data Center Utilization at Each Time Slot

The utilization percentage at each time slot is shown in Fig. 3.17, respectively. The figure also shows the active operation periods at each time slot for each DC. It is shown that model PS-ETS is able to divert the allocated demands from DC1 to DC2 at time slot 2, which means that the operator would operate two DCs only at time slot 2 instead of three. Hence, this leads to power saving for one time slot each day. This can be noted due to the constraint on the already existing resources at DC2. The dynamic allocation is supported by the cloud orchestration, which synchronizes the state of DC1 and DC2, while SDN provides the dynamic network steering of the traffic to DC2 for time slot 2 and back to DC1 for the other time slots. Again it is not

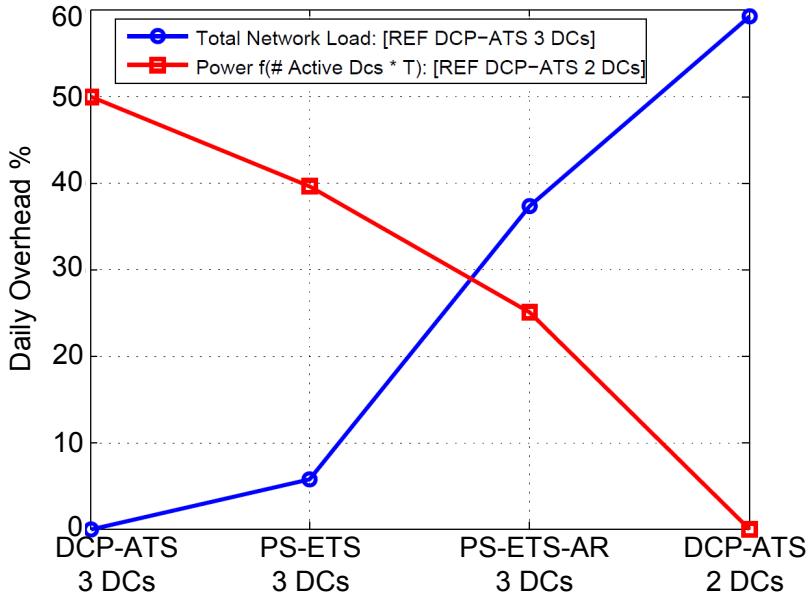


Figure 3.18: Daily total network load compared to data centers power consumption.

possible to re-allocate the demands of DC3 at any time slot due to its remoteness and the data plane delay budget of $15ms$.

Moreover, model PS-ETS-AR is able to divert the traffic demands of DC1 to DC2 for time slots 1, 2 and 3, due to the acquired additional resources at DC2 which leads to more daily power savings. However, note that the utilization % at DC2 can be observed to be lower compared to the other models for time slots 2 and 4, due to the increased amount of available resources and the traffic demands at these time slots.

3.6.6.7 Transport Network Load vs. DC Power Consumption

Fig. 3.18 shows the resulted total transport network load compared to the DC power consumption for all 4 cases. The power consumption is defined as number of active DCs multiplied by active time daily. The comparison is shown as daily overhead %, where the transport load has the reference of DCP-ATS with 3 DCs as it achieves the minimum transport load while for power consumption has the reference of DCP-ATS with 2 DCs as it intuitively results in the minimum power consumption according to the aforementioned definition.

The trade-off between the resulting network load overhead compared to the power consumption overhead can be noted, as model PS-ETS with 3 DCs results in 6% load overhead while it offers power savings of 10%, compared to DCP-ATS

with 3 DCs, respectively. Regarding model PS-ETS-AR with 3 DCs, it shows an increase in the transport load of 37% while it offers in return power savings of 25%, compared to DCP-ATS with 3 DCs, respectively. Hence, an operator would adopt either of the 4 cases depending on the cost resulted from increasing the transport network load compared to costs endured due to power consumption.

3.7 Summary and Discussion

In this chapter, we discuss the SDN and NFV mobile core network architectures. Through detailed analysis, we show the trade-offs between both architectures, in terms of data plane latency and data plane traffic cost. We model the SDN and NFV core network as an optimization problem, what we call the Function Placement Problem (FPP). The problem aims at finding the optimal network design in terms of data traffic cost while satisfying a data plane latency budget. The model offers a possible network design where SDN and NFV are both used in the mobile core network. The problem is evaluated against different number of available data centers, data plane delay budgets and SDN control volume. The solutions of the function placement problem show that a combined deployment of SDN and NFV is most suitable with respect to the core network load cost and data plane latency constraints.

Additionally, we extend the function placement problem to find the optimal core network design under time-varying traffic patterns that shows a trade-off between the network load cost and energy savings. In order to exploit the dynamic flexibility offered by virtualization and SDN as well as the variation of the traffic over time, the time frame is split into time slots in which data centers can shut down. Several models are formulated which minimize the transport load, however, with the possibility of operating fewer number of DCs at each time slot for power saving purposes, namely power saving at each time slot (PS-ETS) and power saving at each time slot with additional DC resources (PS-ETS-AR). The evaluation of the energy saving models shows the trade-off between transport network load cost and power consumption which demonstrates the advantage of considering the time-varying property of the traffic for network dimensioning.

Chapter 4

Dimensioning and Planning of an SDN and NFV Mobile Core Network

In this chapter, we leverage the insights observed in Chapter 3, considering the function placement, SDN and NFV combined deployment, data plane latency requirements and network load cost, to develop models for the data centers resources cost and multi-objective pareto optimal dimensioning and planning. The proposed models in this chapter aim at network planning and dimensioning under variable network traffic demands, in order to provide operators with concrete network designs under changing demands. Furthermore, the proposed models in this chapter are extended with LTE control plane, in addition to the data plane and SDN control plane, in order to evaluate the impact of the LTE control plane requirements on the mobile core network design. This chapter is based on our work in [65].

4.1 Introduction and Motivation

In the previous chapter, we demonstrated the trade-offs between an SDN and an NFV mobile core network architecture, which have been modeled as the function placement problem for static as well as time-varying traffic demands. The problem results have shown that there is quite a benefit from a combined SDN and NFV deployment instead of a sole deployment of either SDN and NFV. In this chapter, we extend the optimization models to solve the planning and dimensioning of the SDN and NFV mobile core network architecture. The focus in the previous chapter is on the evaluation of SDN and NFV on the data plane traffic and the addition of SDN control to the mobile core network. In this chapter, we extend our analysis

and modeling to the LTE control plane in order to evaluate its impact compared to the data plane and SDN control. Additionally, in this chapter, we show a comparison between the different cost factors that are induced by SDN and NFV as well as exploring multi-objective pareto optimal solutions for the core network design.

The optimization models are required to consider the new realization of the mobile core functions as well as the new mobile core network infrastructure. Such infrastructure comprises of a mix of networking forwarding elements, i.e., switches, as well as cloud infrastructure, i.e., data centers. The models should also incorporate new traffic models for the data as well as control planes, e.g., additional SDN control plane traffic. There are different cost factors that can be optimized in the new core network design based on SDN and NFV. The first cost factor is the network load cost, which represents the cost of the network resources needed to support the data and control plane traffic of the mobile core network. The second cost factor is the data center resources cost, which is introduced by the new concepts of NFV and SDN where data centers are needed to host the VNFs and SDN controllers.

In the previous chapter, we have introduced the optimization model that incorporates both SDN and NFV core network functions. However, we have only considered the optimization of the network load cost. We also focused only on data plane paths and data plane latency requirements. Hence, in this chapter, we extend the network load cost optimization model to include control plane paths and control latency requirements to provide a more comprehensive overall model for a mobile core network. We also propose a new optimization model for the data center resources cost to analyze the trade-offs between the network load and data center resources cost factors. In order to find pareto optimal cost solutions considering both the network as well as data center resources cost, a multi-objective model is proposed. We use prior inference, based on the single objective solutions, to pre-select candidate data center locations for the pareto optimal multi-objective model in order to improve its run time. All three proposed models take into account the data and control plane latency as key performance metrics, as well as the number of data centers that are used for deployment.

The objective of the proposed optimization models is to find the optimal design for a mobile core network based on SDN and NFV. These models provide optimal cost solutions with respect to the following aspects: a) the optimal placement of the data centers, which host the mobile VNFs and mobile SDN controllers, b) the opti-

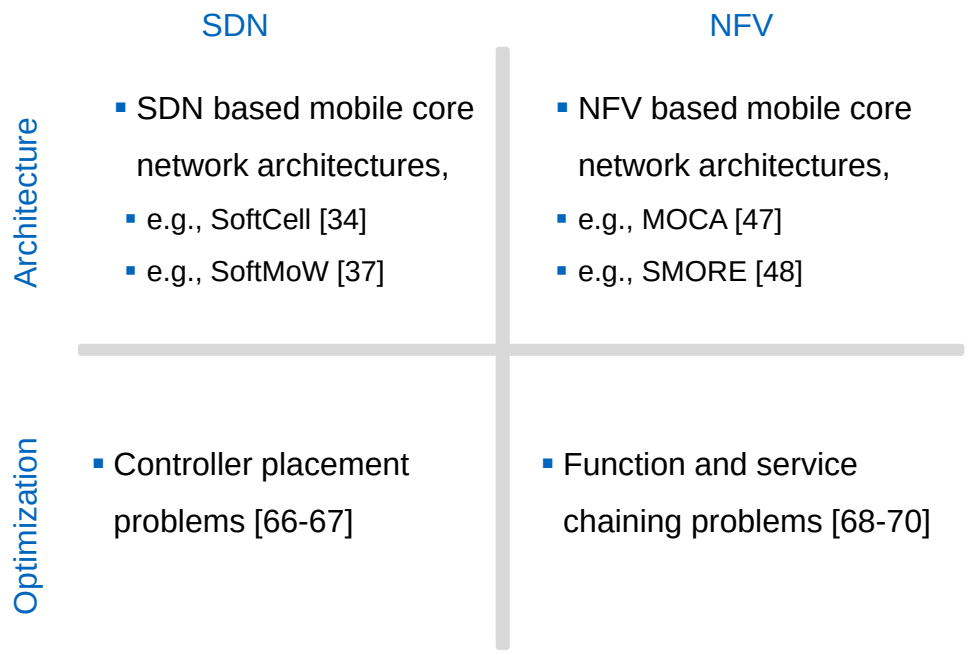


Figure 4.1: SDN and NFV mobile core network state-of-the-art which illustrates that joint architectures and models for both SDN and NFV concepts are missing.

mal mapping of VNFs and controllers to each data center, and c) the number and placement of the mobile special purpose SDN+ switches. All models consider the mobile core network data plane, control plane as well as SDN control traffic.

4.2 SDN and NFV Mobile Core Network Models State-of-the-Art

The state-of-the-art literature can be classified into two areas as illustrated in Fig. 4.1. The first area (upper part of the figure) is concerned with the architecture and implementation designs for SDN or NFV mobile core networks, as discussed in Chapter 3. The second area (lower part of the figure) considers the modeling and optimization of SDN or NFV networks, for mobile networks and for traditional IP networks. In both areas, we could observe a clear split of the work into either SDN or NFV related.

There are two main areas of modeling and optimization related to the use of SDN and NFV in the mobile core network: (a) placement of SDN controllers and switches and (b) resource allocation and placement of VNFs.

The dimensioning and placement of SDN controllers and switches is known as the controller placement problem. This problem has been introduced in [51] which uses a brute force approach to find the placement of K number of controllers and the assignment of switches to each controller targeting a minimum control plane latency. A controller placement based on a simulated annealing heuristic has been proposed in [66] with a focus on control plane latency and resilience aspects. The authors in [67] provide a mathematical formulation for an optimal controller placement that considers both control latency and controllers load. However, all of these problems concerning the placement of SDN controllers only focus on the SDN control plane and fall short in modeling the specific LTE control plane procedures. The existing controller placement problems also focus more on the SDN control latency as well as the controllers load, while less attention is to given to the SDN control traffic overhead induced by an SDN realization.

Considering the resource allocation and placement of VNFs, the authors in [68] demonstrate an optimal placement for virtual core gateways that handle sudden traffic increase in case of large crowd events. [69] presents a mathematical formulation for an optimal placement of virtual function chains. They consider constraints on the network capacity as well as requested latency for a function chain. In [70], the authors present two algorithms to embed service chains with a target of minimizing the embedding cost. The authors in [71] use machine learning techniques to find an optimal placement for VNFs given data center resources. For mobile networks, an optimization for the network resources, i.e., link and node capacity, has been proposed in [72] for the embedding of virtual mobile core network functions. However, the above proposed solutions do not capture the mobile core network models, with respect to the data or control plane traffic. Additionally, most virtual network function models focus more on the data center resources, i.e., the host infrastructure for the virtual network functions. Most proposals neglect the transport network resources, which is in our case the mobile core network between the radio access networking and the data centers themselves.

Reviewing the existing related literature on modeling and optimization, we can observe that models that jointly consider SDN and NFV are missing. Additionally, only a few proposals incorporate the detailed functions, operations and requirements of the mobile core network as we aim at in our work. Furthermore, there are only a few proposals that investigate the impact of the data plane as well as the

control plane latency requirements. There is no existing work, to our knowledge, that is tailored for mobile core functions and considers the optimization of both the network load cost as well as data center resources cost factors. The focus of our work is to develop joint SDN and NFV optimization models while grasping the details of the mobile core network, i.e., combine the architectures and problems shown on the east and west parts of Fig. 4.1.

4.3 LTE Control Plane Analysis

The LTE control plane procedures in the mobile core network consist of multiple sequential iterations, i.e., request and response rounds, between the network functions. For instance, the *ATTACH* procedure, refer to the 3GPP standard [21] as illustrated in Fig. 4.2, involves mainly the MME, SGW and PGW for the setup of a user GTP tunnel. The *ATTACH* procedure defines the control messages exchanged in order to attach a user to the mobile network and setup its data plane GTP tunnel. It includes three control iterations between the RAN and the MME, two control iterations between the MME and the SGW and two control iterations between the SGW and PGW, respectively. Hence, the control plane is required to be modeled differently from how the control plane is modeled in traditional IP networks. Existing work, as discussed in Sec. 4.2, e.g., [69], models the control plane paths as uni directional demands. This does not accurately model the control at the LTE mobile core network where sequential control iterations are required.

Considering an SDN deployment for the mobile core gateway functions, the control plane paths would be mapped on the path between the RAN, i.e., eNBs, and the data centers which run the virtual control functions, i.e., vMME and the SDN controllers. This makes the control plane latency dependent on the location of the data centers. The control paths are also extended by the control path between the SDN controllers and their respective SDN+ switches. For an SDN deployment, the data and LTE control as well as SDN paths are shown in Fig. 4.3.

Alternatively, an NFV deployment for the mobile core gateway functions means that the mobile core VNFs are all consolidated in data centers. Hence, the control plane paths are mapped on the path between the RAN and the VNFs that are deployed at the data centers infrastructure. Therefore, the latency of the control plane paths becomes dependent only on the locations of the data centers. For an NFV

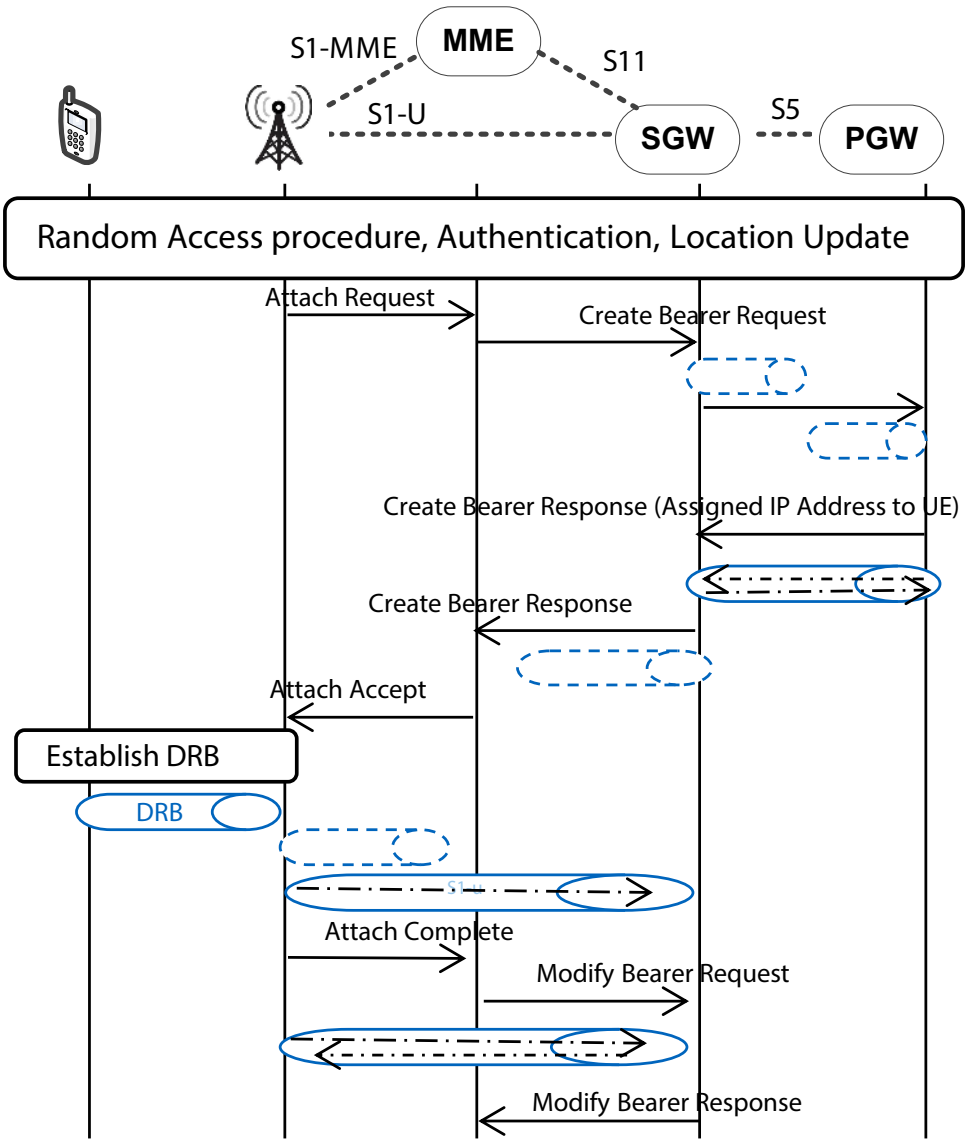


Figure 4.2: 3GPP LTE standard UE Attach procedure [21].

deployment, the data and LTE control paths are illustrated in Fig. 4.4.

From the analysis, we could observe that each concept, either SDN or NFV, results in different paths for both data and control planes. This results in different latency for the data and control planes, that are influenced by the locations of the

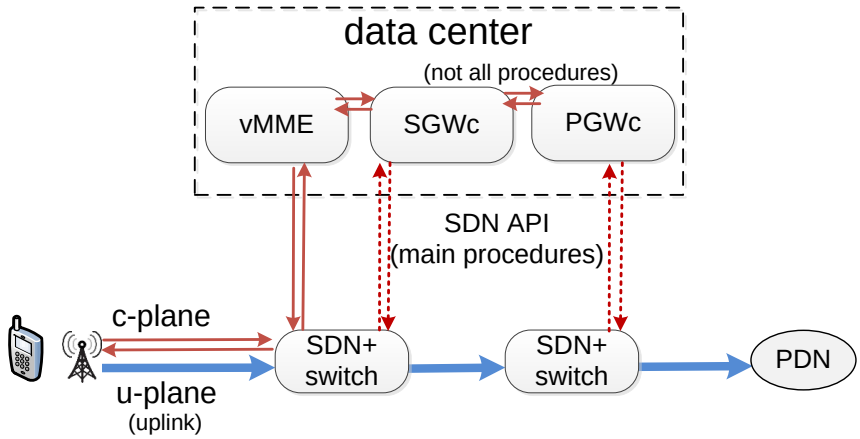


Figure 4.3: SDN data and control plane traffic paths.

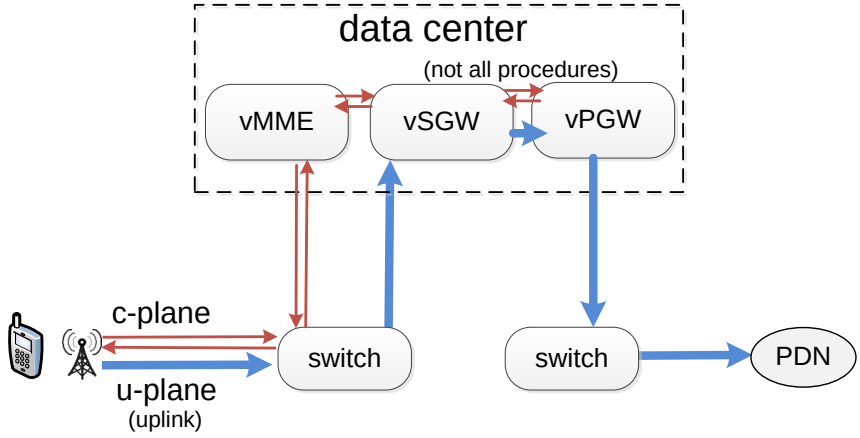


Figure 4.4: NFV data and control plane traffic paths.

data centers and, in case of SDN, by the control paths between the SDN controllers and switches. Moreover, each path impacts the traffic distribution over the network, where an NFV deployment adds data plane traffic on the links towards the data centers while SDN adds extra overhead traffic for the SDN control plane. As for the data center resources aspect, the VNF gateway functions require more computational and networking resources than the SDN controllers, since they involve the processing of high volume data plane traffic.

SDN and NFV deployments show trade-offs in terms of data and control plane latency, network traffic and data center resources. Hence, novel optimization models are required to find optimal planning and dimensioning for a mobile core network, that includes both SDN and NFV deployments, in terms of the network load cost and the data center resources cost.

4.4 SDN and NFV based Mobile Core Network Dimensioning Models

In this section, three optimization models are proposed for the optimal cost dimensioning of the mobile core network based on both SDN and NFV concepts. We introduce the mathematical formulation for the models and the used notations for each of the proposed models. The optimization models are formulated as Mixed Integer Linear Programs (MILP). In general, the aim of the proposed models is to find the optimal dimensioning and resource allocation of the core network that satisfies data plane and control plane latency requirements given a core network topology and number of data centers. The models are used to solve a) the optimal placement of the data centers, which host the mobile VNFs and SDN controllers, b) the optimal mapping of VNFs and controllers to each data center and c) the number and placement of the special purpose SDN+ switches that implement the data plane functions of the core network. The first model targets the optimal network load cost, the second model optimizes the data center resources cost, while the third model is a pareto optimal multi-objective model that results in pareto optimal cost for the network load and data center resources.

Graph Model and Notation

A core network graph $G(V, E)$ is considered with a set of nodes V and edges E. The core nodes are classified as SGW nodes $v^s \in V^s \subset V$ and PGW nodes $v^p \in V^p \subset V$. We assume a brownfield scenario where an operator would select a location to deploy a data center (DC) where it already has a deployed node, thus, data center nodes, i.e., locations, $C \subseteq V$. The set D contains flow demands in the core network, where a flow demand $d = (v^s, v^p) \in D$ represents the requested bidirectional and non-splittable data plane traffic flow, i.e., uplink and downlink, between an SGW node v^s and PGW node v^p. The data and control planes of each demand can be realized as SDN or NFV paths, respectively. For each demand, the set $F^d(c, d)$ contains the SDN and NFV data plane paths of a demand $d \in D$ using a data center $c \in C$. Similarly, the set $F^c(c, d)$ contains the SDN and NFV control paths of a demand d using a data center $c \in C$.

Regarding the NFV realization of a demand $d = (v^s, v^p)$, the data plane path is defined as the path traversing SGW node v^s, the VNFs deployed at the data center nodes c and the PGW node v^p , while the control plane is defined as three times

Table 4.1: Dimensioning and planning problem sets.

Notation	Description
$G(V, E)$	core network graph
C	set of nodes (locations) for data centers $C \subseteq V$
V^s	set of SGW nodes (locations) $V^s \subset V$
V^p	set of PGW nodes (locations) $V^p \subset V$
E	set of physical network edges
D	set of traffic demands $d = (v^s, v^p) \in D$
$F^d(c, d)$	set of data paths for demand $d \in D$, DC $c \in C$
$F^c(c, d)$	set of control paths for demand $d \in D$, DC $c \in C$

the path between the SGW node v^s and VNF deployed at the data center $c \in C$, as explained for the ATTACH procedure in Sec. 4.3. As for the SDN realization, the data plane path represents the path between the SDN+ switches, instead of the SGW node v^s and PGW node v^p. The control plane path is defined as three times the path between the the SGW node v^s and the SDN controller deployed at data center node c in addition to the maximum of the two paths between the controller and switches at v^s and v^p, respectively.

All combinations of data and control paths with data center locations in the sets $F^d(c, d)$ and $F^c(c, d)$ are pre-calculated for each demand, i.e., calculated and provided as input to the optimization problem in order to simplify the problem and improve the solving time. The end-to-end latency of each path is additionally pre-calculated. Assuming an underlying optical transport layer in the mobile core network, the latency $\ell(e)$ of an edge e is calculated as the geographic distance in $kilometers$ between any two connected nodes divided by the speed of light 2^8 m/s in optical fiber. The latency of a path is the sum of latencies $\sum_e \ell(e)$ on the edges e that belong to a data path $f^d(c, d) \in F^d(c, d)$ or a control path $f^c(c, d) \in F^c(c, d)$. Note that the processing latency of VNFs and carrier-grade SDN+ switches are assumed to be insignificant compared to the network propagation latency of a wide spread core network topology, according to our measurements and observations in Chapter 3.

4.4.1 Network Load Cost Optimization Model

In this section, we show the extensions done to the formulation of the function placement problem presented in Sec. 3.5.1. This model aims at optimizing the network cost, i.e., it finds the dimensioning and resource allocation that provides an optimal network cost. The model's cost function, what we call the network traffic load or

Table 4.2: Dimensioning and planning problem parameters.

Notation	Description
K	number of data centers
L^d	data plane latency requirement
L^c	control plane latency requirement
$r(d)$	requested data bandwidth by a demand $d \in D$
$\alpha(d)$	control percentage of $r(d)$ for demand $d \in D$
$l^d(c, f^d, d)$	data plane latency of demand $d \in D$ as a data path $f^d \in F^d$ using DC $c \in C$, 0 otherwise
$l^c(c, f^c, d)$	control plane latency of demand $d \in D$ as a control path $f^c \in F^c$ using DC $c \in C$, 0 otherwise
$n^d(c, f^d, d)$	data plane load of demand $d \in D$ as a data path $f^d \in F^d$ using DC $c \in C$, 0 otherwise
$n^c(c, f^c, d)$	control plane load of demand $d \in D$ as a control path $f^c \in F^c$ using DC $c \in C$, 0 otherwise
$r^d(c, f^d, d)$	DC CPU resources for demand $d \in D$ as a data path $f^d \in F^d$ using DC $c \in C$, 0 otherwise
$r^c(c, f^c, d)$	DC CPU resources for demand $d \in D$ as a control path $f^c \in F^c$ using DC $c \in C$, 0 otherwise
s_{cores}	number of cores in a data center server
p^d_{vnf}	number of cores used by a VNF for data plane
p^c_{vnf}	number of cores needed by a VNF for control plane
p^c_{ctr}	number of cores needed by an SDN controller

shortly network load, is defined in the same way as in the function placement problem. However, the network load cost objective in this section considers both the data as well as control planes of the mobile core network. For each path f using a data center c for each demand, the network load is pre-computed as the requested bandwidth by the demand $r(d)$ multiplied by the latency on the path. Hence, the load for the data path is defined as $n^d(c, f^d, d) = r(d).l^d(c, f^d, d)$, while the load of a control path is defined as $n^c(c, f^c, d) = \alpha(d).r(d).l^c(c, f^c, d)$, where $\alpha(d)$ denotes the control bandwidth percentage of requested data plane bandwidth for this demand. For SDN paths, we consider that the percentage of the mobile control traffic, i.e., signaling, can be assumed to be comparably similar to the traffic resulting from SDN control. The constraints used in this model are defined as follows:

Path and DC Selection

This constraint ensures that for every demand $d \in D$ there is one path selected, i.e., either NFV or SDN, denoted by the binary variables $\delta^d(c, f^d, d)$ and $\delta^c(c, f^c, d)$ for data and control plane, respectively. This path must use at most one data center

c, i.e., place the VNF at this data center location for an NFV path or use this data center to host the controller for the SDN path of this demand.

$$\sum_{c \in C} \sum_{f \in F^d} \delta^d(c, f^c, d) = 1 \quad \forall d \in D \tag{4.1}$$

$$\sum_{c \in C} \sum_{f \in F^c} \delta^c(c, f^d, d) = 1 \quad \forall d \in D \tag{4.2}$$

Path Match

This constraints makes sure that the control path $f^c \in F^c(c, d)$ matches the selected data plane path $f^d \in F^d(c, d)$ for each demand $d \in D$ using a data center location $c \in C$, e.g., if an SDN data plane path is selected for a demand, then the control path of this demand must be SDN. A function $\pi(f^d, f^c)$ returns the path type, i.e., SDN or NFV.

$$\delta^d(c, f^d, d) \leq \delta^c(c, f^c, d) \quad \forall d \in D, c \in C, f^d \in F^d, f^c \in F^c \tag{4.3}$$

DC Selected Flag

A binary variable $\delta(c)$ is utilized in this constraint to flag that this data center location has been selected in case at least one path of one demand has selected the data center c to place the VNF or controller.

$$\sum_{f^d \in F^d} \delta^d(c, f^d, d) \leq \delta(c) \quad \forall d \in D, c \in C \tag{4.4}$$

$$\sum_{f^c \in F^c} \delta^c(c, f^c, d) \leq \delta(c) \quad \forall d \in D, c \in C \tag{4.5}$$

Number of DCs

This constraint defines the maximum number of allowed data center locations to be used. It ensures that the sum of the binary variable $\delta(c)$ overall locations is equal to a pre-defined input parameter K.

$$\sum_{c \in C} \delta(c) = K \tag{4.6}$$

Data and Control Latency

For mobile networks, it is very important to meet the latency performance requirements for both data and control planes, the next two constraints ensure that a selected path using a data center $c \in C$ for a demand $d \in D$ satisfies the upper bound for allowed data and control latency.

$$\sum_{f^d \in F^d} \delta^d(c, f^d, d) l^d(c, f^d, d) \leq L^d \quad \forall d \in D, c \in C \tag{4.7}$$

$$\sum_{f^c \in F^c} \delta^c(c, f, d) l^c(c, f^c, d) \leq L^c \quad \forall d \in D, c \in C \tag{4.8}$$

Network Load Cost Objective

The model's objective is to minimize the network load cost which is defined by the the product of carried traffic and the path length. The network load is the sum of the load of both data and control paths for all demands $d \in D$.

$$\begin{aligned} C_{net} = \min & \sum_{c \in C} \sum_{f^d \in F^d} \sum_{d \in D} \delta^d(c, f^d, d) n^d(c, f^d, d) \\ & + \sum_{c \in C} \sum_{f^c \in F^c} \sum_{d \in D} \delta^c(c, f^c, d) n^c(c, f^c, d) \end{aligned} \tag{4.9}$$

Solving this objective results in finding the optimal locations of K data centers. It also finds the optimal paths for each demand, i.e., either SDN or NFV, based on the selected data center locations in addition to optimally assign the paths to the data centers such that the resulting total network load, i.e., data and control traffic, is minimized.

4.4.2 Data Center Resources Cost Optimization Model

This model aims at optimizing the data center infrastructure cost needed to operate a core network given a set of demands and latency requirements. This model reflects the dimensioning of the data centers independently from the network cost, e.g., in case an operator does not control or does not have access to the inter-data center network. As an initial assumption, we only consider the infrastructure cost as the servers cost. The number of servers is proportional to the number of computational resources, i.e., CPU cores, that are needed for the NFV paths, i.e., virtual gateways,

Table 4.3: Dimensioning and planning problem variables.

Notation	Description
$\delta(c)$	binary variable =1 if DC is located at $c \in C$, 0 otherwise
$\delta^d(c, f^d, d)$	binary variable = 1 if data plane of demand $d \in D$ is selected as a path $f^d \in F^d$, either SDN or NFV, using DC $c \in C$, 0 otherwise
$\delta^c(c, f^c, d)$	binary variable = 1 if control plane of demand $d \in D$ is selected as a path $f^c \in F^c$, either SDN or NFV, using DC $c \in C$, 0 otherwise
$\sigma^d(c)$	integer variable denoting number of servers required for data plane function chains at DC $c \in C$
$\sigma^c(c)$	integer variable denoting number of servers required for control plane function chains at DC $c \in C$
$\mu(c)$	integer variable denoting the total number of servers required for both data and control planes at DC $c \in C$

or SDN paths, i.e., controllers. For NFV paths, the CPU resources needed are precomputed as the requested bandwidth by a demand multiplied by the number of cores required by a virtual gateway per unit demand $r^d(c, f^d, d) = r(d).p^d_{vnf}$ while the control plane resources $r^c(c, f^c, d) = \alpha(d).r(d).p^c_{vnf}$. As for the SDN paths, the number of cores needed for the SDN controllers are $r^c(c, f^c, d) = \alpha(d).r(d).p^c_{ctr}$, while there are no resources needed for the data plane, i.e., $r^d(c, f^d, d) = 0$. This model additionally aims at achieving a resource balancing among the data centers, in case the number of data centers $K > 1$, by minimizing the maximum number of servers allocated at one data center location. This model uses all previous defined constraints, i.e., eqs (4.1)-(4.8). It requires additional constraints for the data centers as follows:

DC Number of Servers

The number of servers, for data and control planes, at each data center $c \in C$ is calculated by adding the resources $r^d(c, f^d, d)$ or $r^c(c, f^c, d)$ used by paths of all demands that use this data center. This gives the total number of CPU cores required at this data center, which is divided by the number of cores per server, what we call the server consolidation factor $\frac{1}{s_{cores}}$.

$$\sum_{d \in D} \sum_{f^d \in F^d} \frac{1}{s_{cores}} \Big(\delta^d(c, f^d, d) r^d(c, f^d, d) \Big) \leq \sigma^d(c) \quad \forall c \in C \tag{4.10}$$

$$\sum_{d \in D} \sum_{f^c \in F^c} \frac{1}{s_{cores}} \Big(\delta^c(c, f^c, d) r^c(c, f^c, d) \Big) \leq \sigma^c(c) \quad \forall c \in C \tag{4.11}$$

Maximum Number of Servers at a DC

The maximum number of servers at a data center is represented by the integer variable $\mu(c)$ which is lower bounded by the data center that has the maximum number of allocated servers, according to this constraint.

$$\sigma^d(c) + \sigma^c(c) \leq \mu(c) \quad \forall c \in C \tag{4.12}$$

DC Resources Cost Objective

This model's objective is to minimize the data center resources cost in terms of the total number of servers required at the deployed data centers. Additionally, it aims at minimizing the maximum number of servers allocated at a single data center location in order to achieve a balanced resource distribution.

$$C_{dc} = \min \sum_{c \in C} \Big(\sigma^d(c) + \sigma^c(c) \Big) + \mu(c) \tag{4.13}$$

solving this objective results in finding the optimal locations of K data centers. It also finds the optimal paths for each demand, i.e., either SDN or NFV, based on the selected data center locations in addition to optimally assign the paths to the data centers such that the resulting total data center resources, i.e., number of required servers infrastructure, is minimized.

4.4.3 Multi-objective Pareto Optimal Model

This model results in pareto optimal solutions between the network load cost and data center resource cost objectives to enable operators to choose the right balance between the two objectives. The multi-objective optimization model includes all constraints from the previous two models, i.e., eqs (4.1)-(4.8) and eqs (4.10)-(4.12). The multi-objective function incorporates both cost functions of the previous two models, where ω_{net} denotes the weight factor for the network load cost objective, while ω_{dc} defines the weight for the data center cost objective. It is formally defined as follows:

$$C_{multi} = \min \quad \omega_{net} C_{net} + \omega_{dc} C_{dc} \tag{4.14}$$

In order to get pareto solutions for the multi-objective problem, i.e., trade-offs between the optimality of the two objectives, the weight ω can be defined as $\omega = \lambda_i.norm_{factor}$. The parameter λ is a variable that goes from 1 to 0, in order to iterate from the optimality of one objective to the other. Since the two objectives, namely network load cost and data center cost, represent different network metrics and have different units, the normalization factor $norm_{factor}$ is used to normalize the two objectives such that they both have the same units and thus contribute similarly to the multi-objective function. This is called in optimization literature as the weighted sum method for pareto optimal multi-objective optimization. This method is one of the most widely used approaches in multi-objective optimization due to its efficiency and simplicity, as it can find the entire pareto frontier through the set of multi-objectives that are scalarized into a single objective [73]. The details of the proposed model are represented as follows:

Multi-objective Pareto Optimal Optimization with Pre-selection Feature for Data Center Locations

Input: no. of DCs K, DC locations $C \subseteq V$,
data and control latency requirements L^d, L^c

1: $min\, C_{net}$, $out\, C_{dc}$, $loc_{net} \leftarrow$ min. network cost C_{net}
2: $out\, C_{net}$, $min\, C_{dc}$, $loc_{dc} \leftarrow$ min. data center cost C_{dc}
3: **(locations pre-selection feature)**
$\{C \leftarrow (loc_{net}, loc_{dc}) || C| = K\}$
4: **for** $\lambda_i = 0 : 0.1 : 1$ **do**
5: $\omega_{net,i} \leftarrow \lambda_i / (out\, C_{net} - min\, C_{net})$
6: $\omega_{dc,i} \leftarrow (1 - \lambda_i) / (out\, C_{dc} - min\, C_{dc})$
7: minimize $C_{multi,i} = \omega_{net,i} C_{net} + \omega_{dc,i} C_{dc}$
8: $C_{net,i} \leftarrow$ post calculation from $C_{multi,i}$
9: $C_{dc,i} \leftarrow$ post calculation from $C_{multi,i}$
10: **end for**

Output: Pareto optimal solutions $(C_{net,i}, C_{dc,i}) \quad \forall \lambda_i = [0, 1]$

In order to get the normalization factors, the single objectives are solved first given a number of data centers K and data as well as control plane latency requirements. Each objective results in an optimal solution for its target and results in an out-turn value for the other target. For instance, solving for the network load objective, it results in the optimal network load cost $min\, C_{net}$ and we could calculate the resulting out-turn data center cost $out\, C_{dc}$. Similarly, we solve the data center cost objective and obtain the optimal data centers cost $min\, C_{dc}$ as well as the resulting out-turn network cost $out\, C_{net}$. The normalization factor for each objective is defined

as the difference between the maximum value for the objective and its optimal solution. The multi-objective function is solved while iterating over λ_i that ranges from 0 to 1, with a step parameter of 0.1. Each solution from each iteration is unnormalized in order to get the set of pareto solutions for the network load and data center resources cost, respectively.

Data Center Location Pre-selection Feature for the Multi-objective Resource Allocation Model

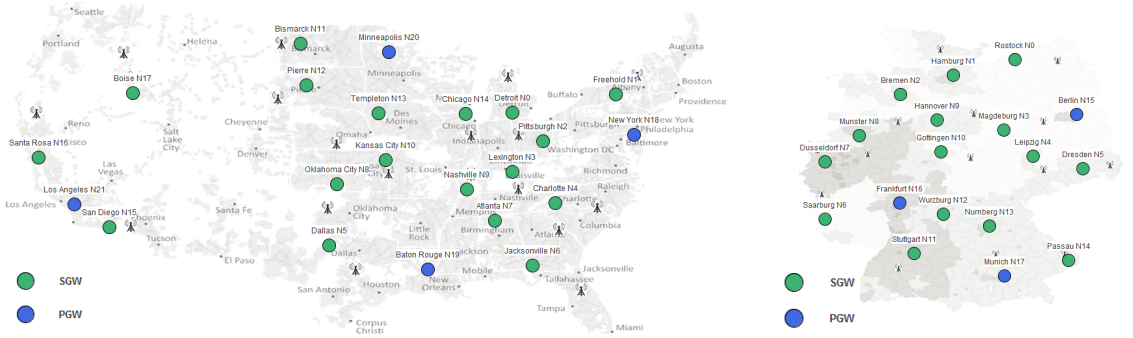
According to previous work and preliminary results, we could observe that each objective function can influence the data center and path placement, i.e., locations. Hence, in order to improve the run time of the multi-objective optimization, we propose a pre-selection for candidate data center locations on the given core network graph from solving the individual objectives, done in steps (1) and (2) of the multi-objective model. The number of pre-selected data center locations is equal to the maximum number of available data centers to be deployed, i.e., size of locations set $|C| = K$.

4.5 Evaluation for the Objective Trade-offs

In this section, an extensive evaluation and sensitivity analysis to the proposed dimensioning and planning models are presented.

4.5.1 Framework

For evaluation, a java framework has been developed that implements the three proposed optimization models in Sec. 4.4. The framework is initialized by reading the graph topology and creating the data plane traffic demands. It also creates the different SDN and NFV paths and computes their associated parameters, i.e., network load, data center resources, data as well as control plane latency. The framework implements the optimization models, where Gurobi is used as the linear optimization solver. Finally, the framework is used to calculate the different parameters and attributes of the solution, from which the framework derives the resulting SDN and NFV mobile core network.



(a) Mobile Core topology (USA)

(b) Mobile Core topology (Germany)

Figure 4.5: Mobile core topologies for both USA and Germany based on LTE coverage and user population. The figure shows the locations of the SGWs (green) and PGWs (blue). The coverage is depicted by the intensity (grey) on the map background.

4.5.2 Mobile Core Topologies

For evaluation, we use a mobile core network topology for the USA based on the LTE coverage map in [74], which correlates with the population distribution as well as Internet Exchange Points (IxP) [75], illustrated in Fig. 4.5a. The US topology consists of 18 SGWs and 4 PGWs with a total of 22 nodes, i.e., potential data center locations. For comparison, we use another mobile core network topology for Germany that has 15 SGWs and 3 PGWs with a total of 18 nodes, shown in Fig. 4.5b. In both topologies, each SGW node is associated to its geographically nearest PGW node, respectively.

4.5.3 Data and Control Plane Traffic Demands

In order to evaluate the mobile network dimensioning cost with respect to the expected traffic increase in the next years and the traffic dynamics introduced by SDN and NFV, we consider random traffic requests for each data plane demand. The demands are uniformly distributed between 10 and 50 Gbps. As for the control as well as the SDN traffic, we have considered a random control traffic ratio between 10 and 30 % of the data traffic request for each demand. The traffic demands are data plane traffic flows between each SGW and its nearest PGW node. The traffic assumptions are projections from the predicted data plane and control plane traffic increase in the next generation 5G network [1] [2]. In order to get statistically evident results, the optimization models are solved for multiple runs until a 95% confidence is reached for the optimization solution or at least for 30 runs.

Table 4.4: Evaluation parameters.

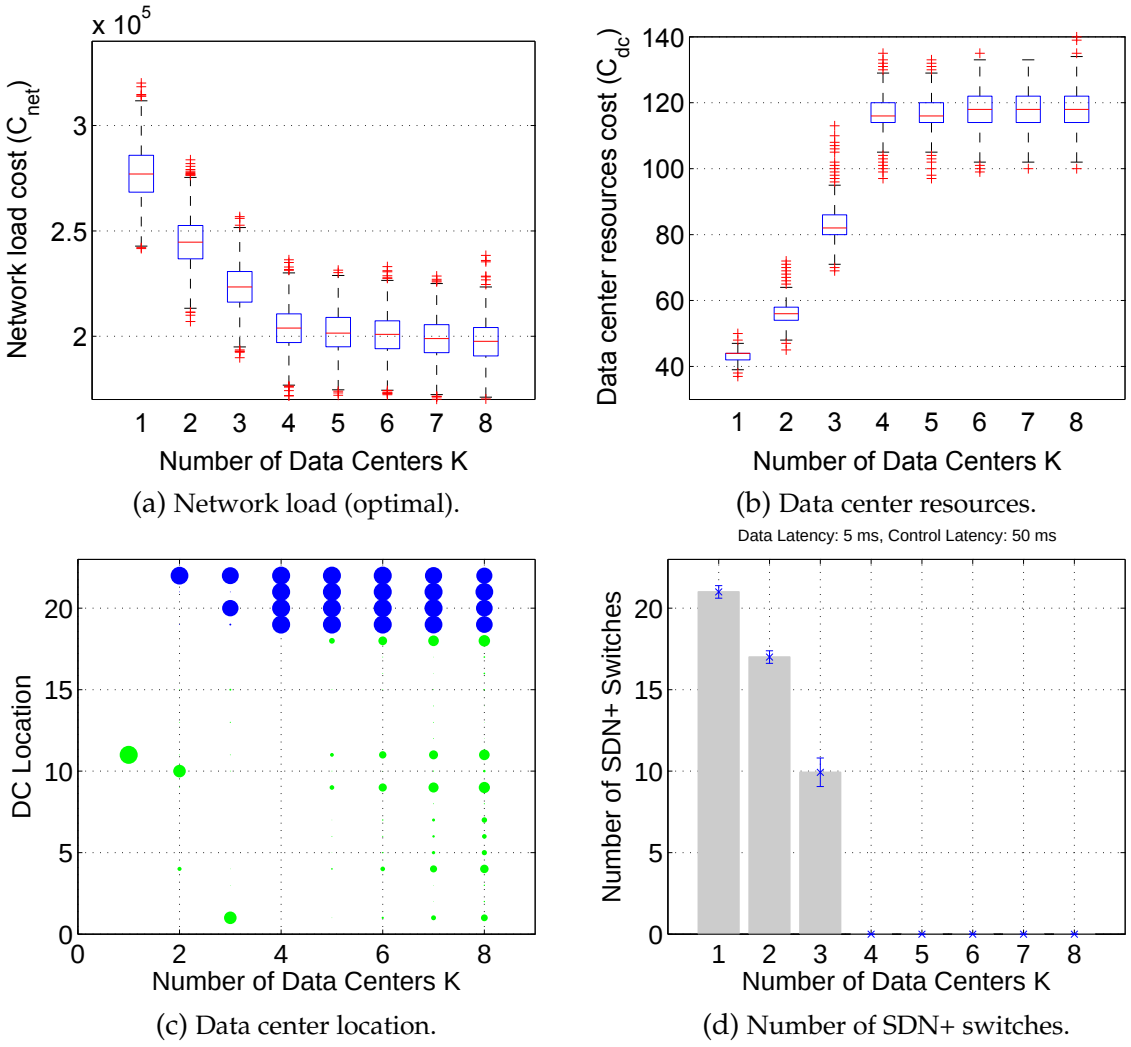
Parameter	Description
Topology	USA (18 SGWs, 4 PGWs) Germany (15 SGWs, 3 PGWs)
Data Traffic demands (Gbps)	uniform distribution [10 - 50]
Control and SDN traffic ratio α	uniform distribution [10-30]%
Data plane latency requirement	data plane uni directional 5 ms
Control plane latency requirement	control plane procedure 50 ms
Number of DC locations K	1 - 8 data centers
CPU cores per unit demand (1 Gbps)	p^d_{vnf}=18 cores, p^c_{vnf}=2 cores p^c_{ctr} = 6 cores
Number of cores per server s_{cores}	48 cores per server

4.5.4 Data and Control Plane Latency Requirements

Moving towards the next generation 5G, data and control latency requirements are critical performance metrics that need to be improved in order to leverage the user experience and performance. Hence, we consider the lowest latency that can be achieved by both considered mobile core networks, US and Germany. As for the data plane latency requirement, we consider a budget of 5 ms for the mobile core network as a uni directional latency, either for uplink or downlink. We have used previous observations from our previous evaluation in Chapter 3. As for the control latency budget, a 50 ms is considered for the control plane, including SDN control for SDN paths, according to 3GPP LTE standards [76, 77]. This control latency covers the end-to-end latency to complete the control iterations of the UE ATTACH procedure.

4.5.5 Data Center Resources

It is intuitive to assume that a VNF that handles both data and control planes would need more computational and processing power than an SDN controller that handles the control plane only. Therefore, according to our measurements in [32], we assign 20 cores for the VNF for the processing of 1 unit data traffic demand, i.e., 1 Gbps, with a distribution of 18 cores for data plane p^d_{vnf} and 2 cores for control plane p^c_{vnf}. As for the SDN controller, 6 cores are allocated for the processing of the control plane p^c_{ctr} that corresponds to a unit data plane traffic demand. As for the consolidation factor s_{cores} that defines the number of cores per server, we assume server sizes of 48 cores that can be typical in current data center deployments [78]. A summary of the evaluation parameters is presented in Table 4.4.



(a) Network load (optimal).

(b) Data center resources.

(c) Data center location.

(d) Number of SDN+ switches.

Figure 4.6: Trade-offs solving for **network load cost** objective for **US** topology, data latency = 5 ms and control latency = 50 ms.

4.5.6 Trade-offs between the Network Load and Data Center Resources Cost Objectives

First, we present an evaluation for the trade-offs between the two proposed optimization models, i.e., the network load cost objective compared to the data center resources cost objective. We also investigate the impact of the data center deployment by going from a single centralized data center, i.e., $K = 1$, up to a distributed data center deployment with $K = 8$. We start by presenting the results for the US topology considering a data plane latency requirement of 5 ms and a control plane latency requirement of 50 ms. The results of optimizing for the network load cost objective are illustrated in Fig 4.6, while the results for the data center resources cost objective are illustrated in Fig. 4.7. For each objective, the results focus on four eval-

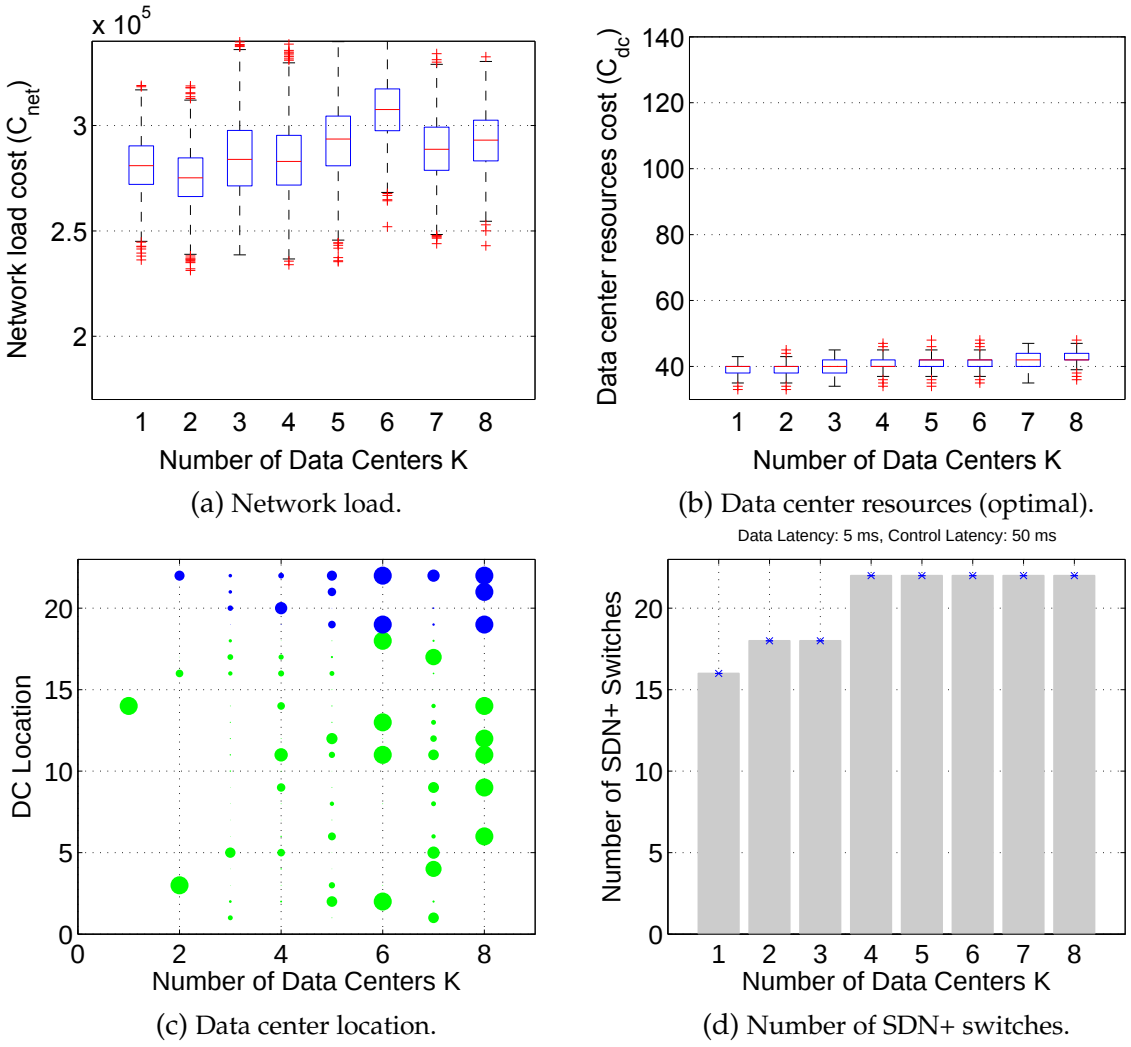
uated criteria which are the network load cost, data center resources cost, data center locations and the number of required SDN+ switches.

4.5.6.1 Network Load Cost Objective, US Topology

Considering the network load cost objective, Fig. 4.6a shows that the optimal network load cost is impacted by the data center deployment choice, i.e., the number of data centers. We could observe that the optimal network load cost could be significantly improved by distributing the data center infrastructure, up to 75% at 8 data centers. The reason for this improvement is that, with more available data centers, more VNFs could be deployed under the given latency requirements, refer to Fig. 4.6d, in order to decrease the additional SDN control traffic and thus decreasing the total network load cost. Additionally, since the network load cost metric considers both the traffic bandwidth and the length of the paths, deploying distributed data centers can decrease the length of the paths across the network. Moreover, we can observe that adding more data centers at $K > 4$ does not bring significant improvements to the optimal network load cost.

Considering the resulting data center resources cost, i.e., the number of servers required, as shown in Fig. 4.6b, a trade-off between the optimal network load and the resulting data center resources cost while increasing the number of data centers K can be observed. The resulting number of servers required with 8 distributed data centers is 275% higher than with a single centralized data center. This is again due to the deployment of more VNFs while increasing the number of available data centers, refer to Fig. 4.6d, which requires more computational CPU cores at the data centers and hence more servers. We can conclude that adding more data centers could optimize and decrease the network load cost further on the expense of needing more servers and increasing the cost for the data centers infrastructure.

Throughout the repeated runs of solving the optimization model given random data and control traffic demands, we could observe several trends in the placement of the data centers, i.e., their locations, as shown in Fig. 4.6c. The frequency of selecting a location for the data centers among the repeated runs is represented by the density of the plotted point, i.e., location, on the figure. The green locations represent the locations of SGWs, while blue locations represent those of PGWs. For instance, at a single data center $K = 1$ and optimizing for the network load cost, we could observe that there is one dominant location (node 11: Kansas City) that is always



(a) Network load.

(b) Data center resources (optimal).

(c) Data center location.

(d) Number of SDN+ switches.

Figure 4.7: Trade-offs solving for **data center resources cost** objective for **US** topology, data latency = 5 ms and control latency = 50 ms.

selected even with varying random demands. This is due to the geographic centrality of this location, which balances the traffic in the network and optimizes the load cost and could satisfy the data as well as control latency constraints with a centralized data center. The other trend that we could observe is that by increasing the number of data centers from $K = [2 - 8]$, the locations of PGWs get more dominant, i.e., they are more frequently selected while varying the input traffic demands. This is because the locations of the PGWs could serve aggregated traffic demands from multiple SGWs, which decreases the distance of transporting the traffic to a different location. Hence, with more than one data center, i.e., distributed deployment, data centers are favored to be placed at the location of PGWs for the network load cost optimization.

Finally, the number of needed SDN+ switches with respect to the number of data centers is illustrated in Fig. 4.6d. As mentioned before, the network load cost optimization attempts at decreasing the additional control traffic induced by SDN and thus aims at deploying more VNFs. However, according to the data center locations, the data and control latency requirements might not be satisfied for all demands with only VNFs, therefore the need for SDN+ switches. The number of SDN+ switches decreases while increasing the number of data centers K, going from a single centralized data center up to 3 distributed data centers. A network that comprises only of virtual functions is possible starting from 4 data centers.

4.5.6.2 Data Center Resources Cost Objective, US Topology

Here, the same four evaluation metrics as before are used, however, while solving for the optimal data center resources cost, in terms of the total number of servers. The results are shown in Fig. 4.7. First, we start by discussing the target of this optimization model, i.e., data center resources cost, illustrated in Fig. 4.7b. We observe that the optimal solutions are less impacted by the available number of data centers. This can be explained by observing the number of SDN+ switches shown in Fig. 4.7d. Since SDN controllers require less computational cores at the data centers, the model's solution results in almost a full SDN deployment given the data and control latency requirements. This results in decoupling the optimal data center resources cost from their deployment design, i.e., centralized or distributed. Additionally, the optimal data center resources cost increases slightly while increasing the number of data centers from a centralized $K = 1$ to distributed $K = 8$ deployment. This is due to the possibility of consolidating more cores on servers with centralized data centers which decreases the total number of required servers. With a distributed data center infrastructure, servers are needed at each location without the full utilization of their computational cores. However, the optimal data center resources cost in Fig. 4.7b is much lower than the resulting data center resources cost while optimizing for the network load cost objective in Fig. 4.6b, e.g., at $K = 8$, savings of 260% in terms of number of servers can be achieved.

As for the resulting network load cost with the data center resources objective, shown in Fig. 4.7a, we could observe fluctuations in the resulting load cost varying with the number of data centers. This shows the trade-off between the network load cost and data center resources cost, where optimizing the data center resources only

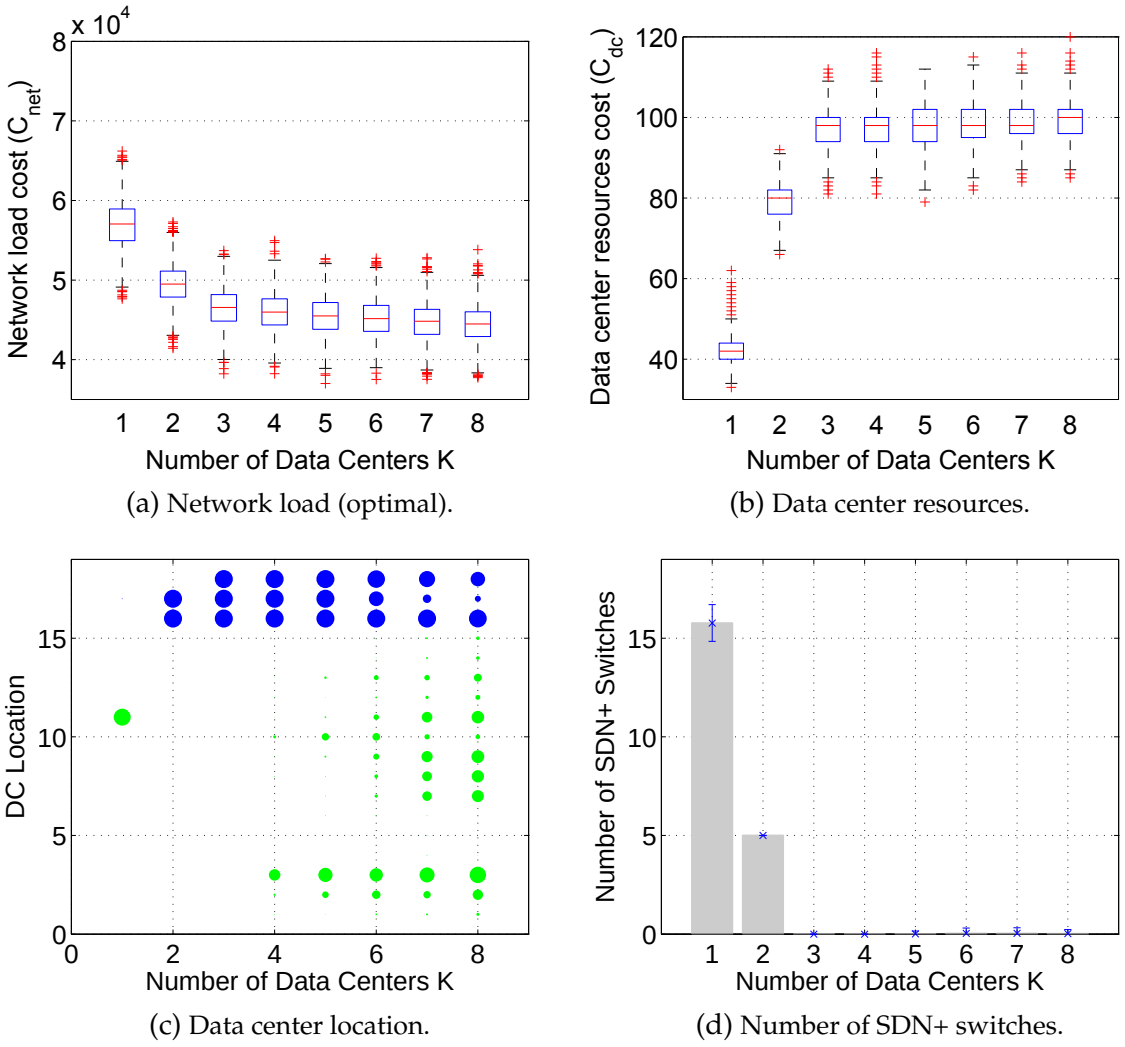
as an objective results in a quite high network load cost in return. In fact, this points out to the necessity of our third model, i.e., multi-objective pareto optimization, such that the operator can find pareto solutions that balance between the network load cost and data center resources cost.

Regarding the locations of the data centers selected throughout the repeated runs with varying random traffic demands, Fig. 4.7c shows that the data center locations are more biased towards the locations of SGWs, i.e., towards the network edge, while using the data center resources cost objective. We could also observe that the selected locations are more sparse and diverse depending on the traffic demands. These trends are different from what has been observed before with the network load objective, refer to Fig. 4.6c. The data center placement in this case is biased with the control plane latency requirement. Since this model attempts to use more SDN controllers to save on the data center resources, the data centers are placed more towards the edge in order to enable more SDN controllers to satisfy the control plane latency requirement.

Finally, Fig. 4.7d, shows the number of SDN+ switches needed for the data center resources cost objective compared to the number of data centers. We could observe that more SDN+ switches are used in this objective compared to the network load objective. Additionally, the network turns to a full SDN deployment starting at $K = 4$ data centers, which is the same K for the network load objective where the network turns to a full NFV deployment.

4.5.7 Trends with Different Topologies

In this section, we investigate whether the previously observed trends for the two cost optimization models can also be observed with different topologies. Therefore, we have repeated the previous evaluation for the German topology. Results are illustrated in Fig. 4.8 and Fig. 4.9. The results demonstrate similar trends for the German topology as for the US topology for both network load and data center resources cost objectives. Hence, the repetition of the trends for the evaluated topologies can support our proposed pre-selection of the data center locations for the multi-objective optimization model from the resulting locations in the single objective models.



(a) Network load (optimal).

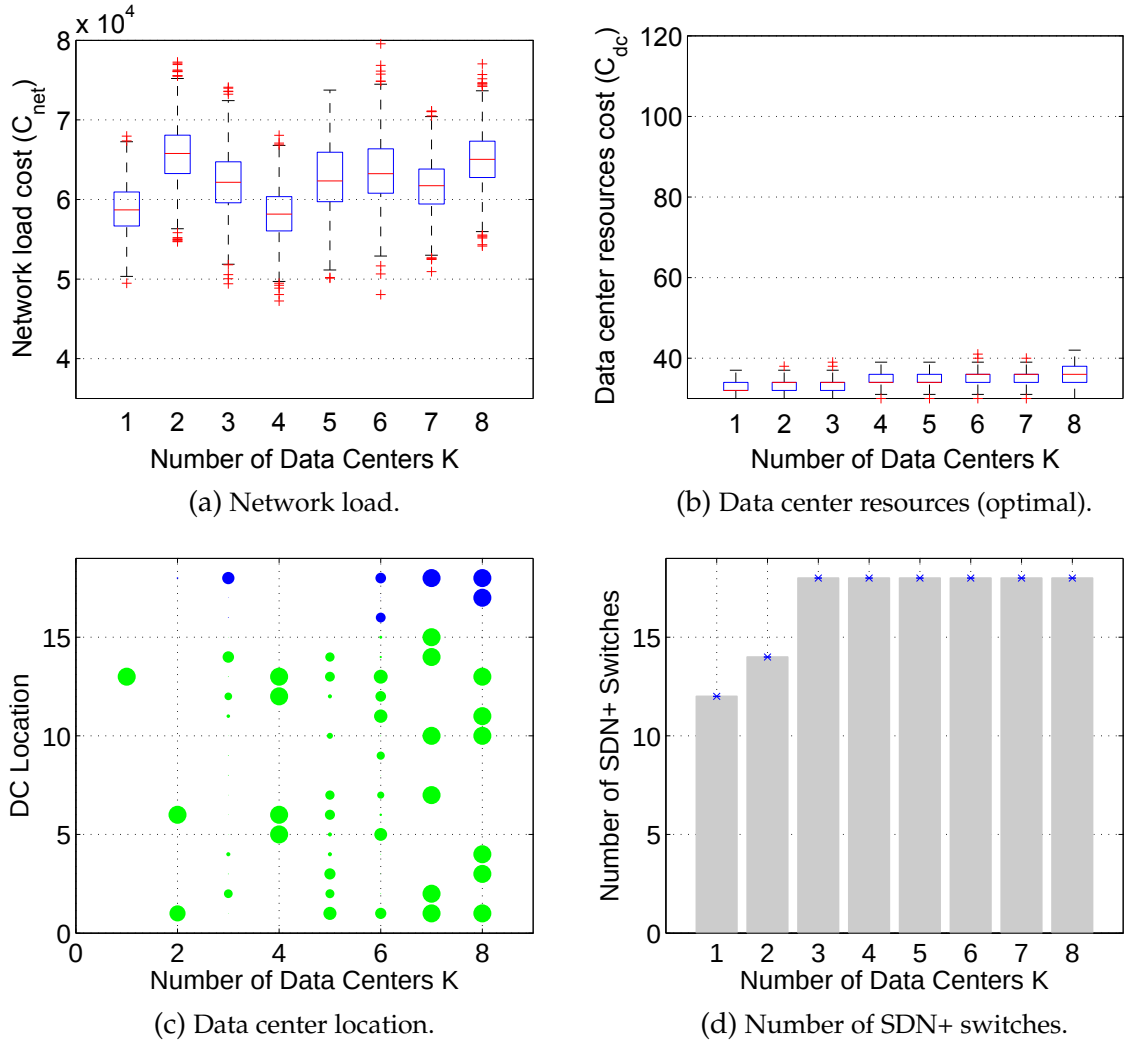
(b) Data center resources.

(c) Data center location.

(d) Number of SDN+ switches.

Figure 4.8: Trade-offs solving for **network load cost** objective for **German** topology, data latency = 5 ms and control latency = 50 ms.

Note that the number of data centers K at which a full NFV deployment with the network load cost objective, or a full deployment of SDN+ switches with the data resources cost objective, differs between the two topologies. For the German topology, it is possible starting from $K = 3$ compared to $K = 4$ for the US topology. As previously explained, this is influenced by the number of PGWs that the topology contains, where the German topology contains 3 PGWs, compared to 4 PGWs at the US topology. This can be remarked as a trend observation, where a full deployment, either SDN or NFV depending on the cost objective, is possible starting from K data centers equal to the number of PGWs.



(a) Network load.

(b) Data center resources (optimal).

(c) Data center location.

(d) Number of SDN+ switches.

Figure 4.9: Trade-offs solving for **data center resources cost** objective for **German** topology, data latency = 5 ms and control latency = 50 ms.

4.6 Evaluation for the Multi-objective Model

4.6.1 Gain from Pareto Optimal Multi-objective Model

First, we investigate the results of the pareto optimal multi-objective model without data center locations pre-selection and we compare it to the results of the single cost objective models. As explained in Sec. 4.4.3, the multi-objective method iterates over different weight parameters λ for each objective ranging between [0,1]. In other words, it explores the solution space starting by solving one single objective, then moving to solve both objectives simultaneously, and stops after solving the other single objective, thus it produces the pareto frontier between the two objectives. For each weight parameter λ, the setup is again repeated with random varying traffic

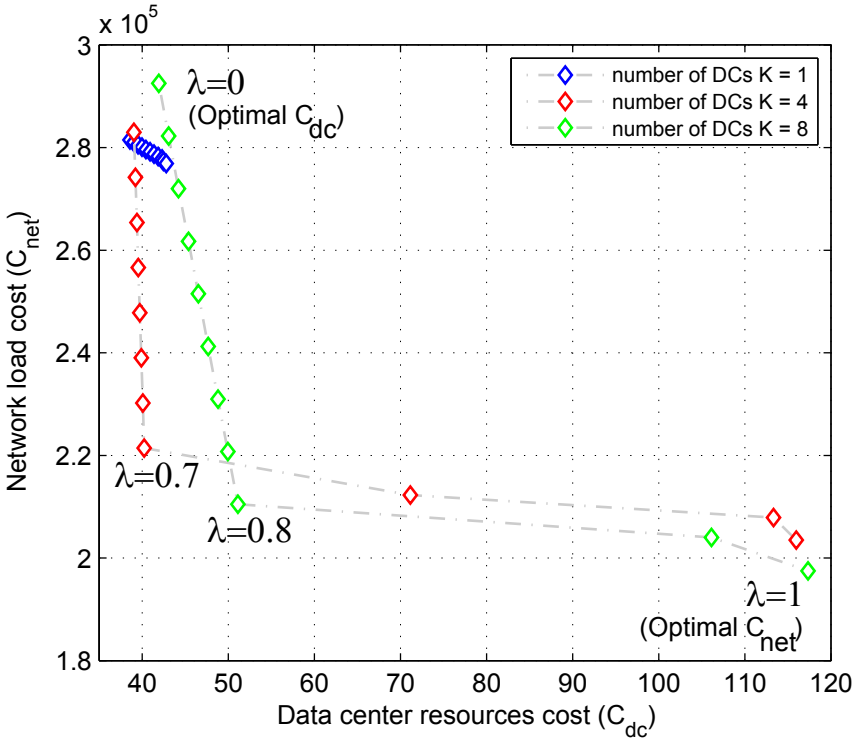


Figure 4.10: Pareto frontier for the network load cost (C_{net}) and data center resources cost (C_{dc}), solving the pareto optimal multi-objective model at number of data centers $K = (1, 4, 8)$ for the **US** topology.

demands till a 95% confidence is reached or at least with 30 runs.

Fig. 4.10 illustrates the pareto frontier between the network load cost and the data center resources cost, for the US topology and given a data latency requirement of 5 ms and a control latency requirement of 50 ms. The evaluation is demonstrated for a number of data centers $K = (1, 4, 8)$. We could observe that for a single centralized data center $K = 1$, there is not enough degree of freedom to explore the solution space and provide a balance or trade-off between the network load cost and the data center resources cost. The reason is that the data center locations, which satisfy both the data as well as the control plane latency requirements with a centralized data center, are quite limited.

Considering a distributed data center infrastructure with $K = 4$, more pareto solutions offering trade-offs between the two objectives can be observed. For instance the pareto solution at $\lambda = 0.7$, the network load cost has an overhead of only 3% compared to its optimal solution at $\lambda = 1$, while the data center resources cost results in an overhead of 4% compared to its optimal solution at $\lambda = 0$. Considering more distributed data centers at $K = 8$, we could observe that there could be more

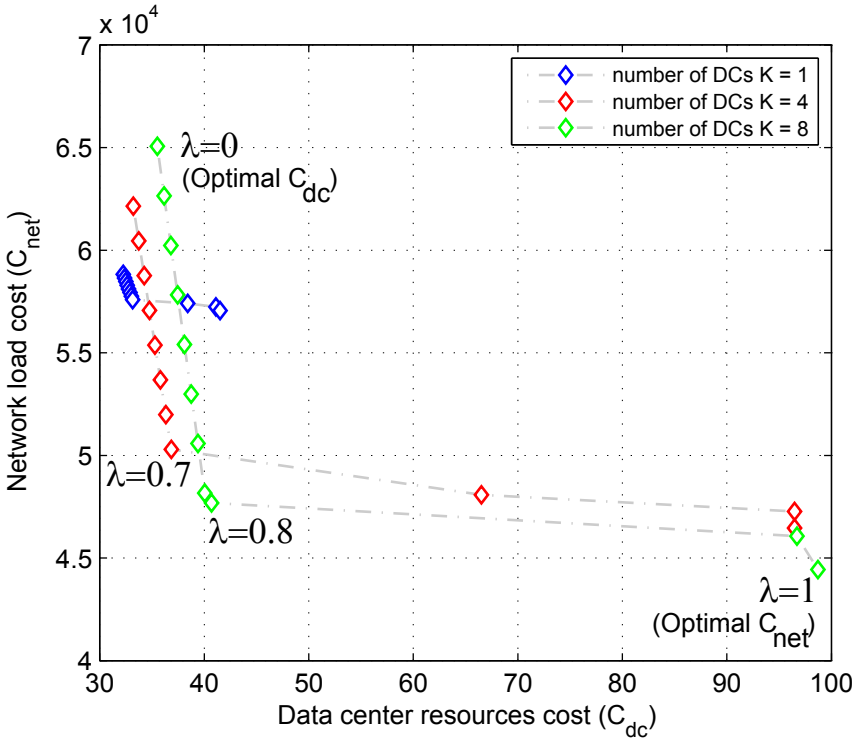
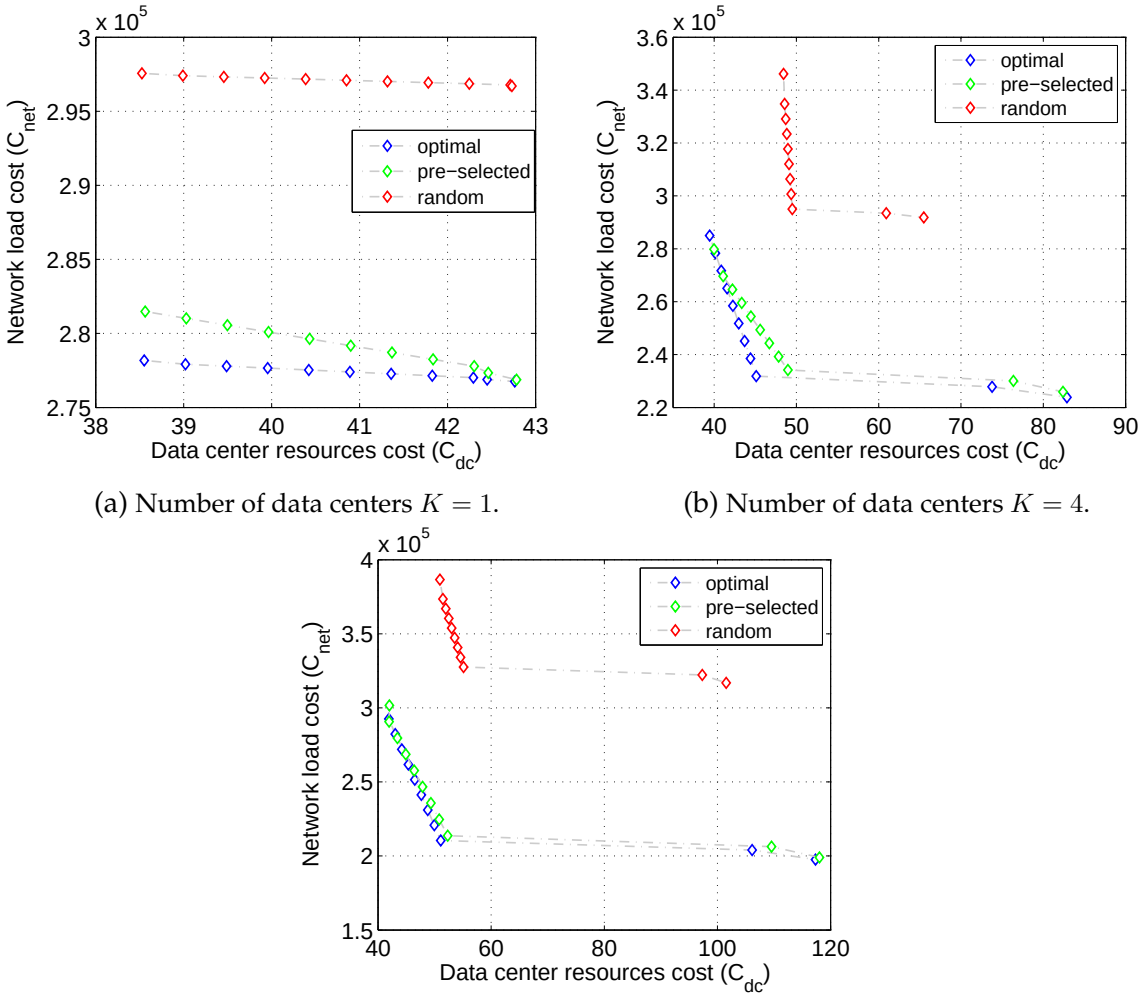


Figure 4.11: Pareto frontier for the network load cost (C_{net}) and data center resources cost (C_{dc}), solving the pareto optimal multi-objective model at number of data centers $K = (1, 4, 8)$ for the **German** topology.

degree of freedom to cover a larger solution space. For instance, considering the pareto solution at $\lambda = 0.8$, the network load cost has an overhead of 5% compared to its optimal solution, while the data center resources witness an increase of 21% compared to its optimal solution. It is worth mentioning that an operator could go for a different pareto solution depending on the cost values for each of the network traffic load and the data center resources. In general, the pareto frontier shows the advantage of finding solutions that could not be easily found through arbitrary weights to each objective in the multi-objective function. This provides operators with the possibility to find the optimal network that balances between the network load cost and the data center resources cost.

The evaluation for the pareto optimal multi-objective model for the German topology is shown in Fig. 4.11. We demonstrate the pareto frontier evaluation for the number of data centers $K = (1, 4, 8)$. Similar trends for the pareto frontiers could be observed as in the US topology. However, more pareto optimal solutions could be obtained with a centralized single data center at $K = 1$. Since the German topology is geographically smaller than the US, this provides more locations to the single data



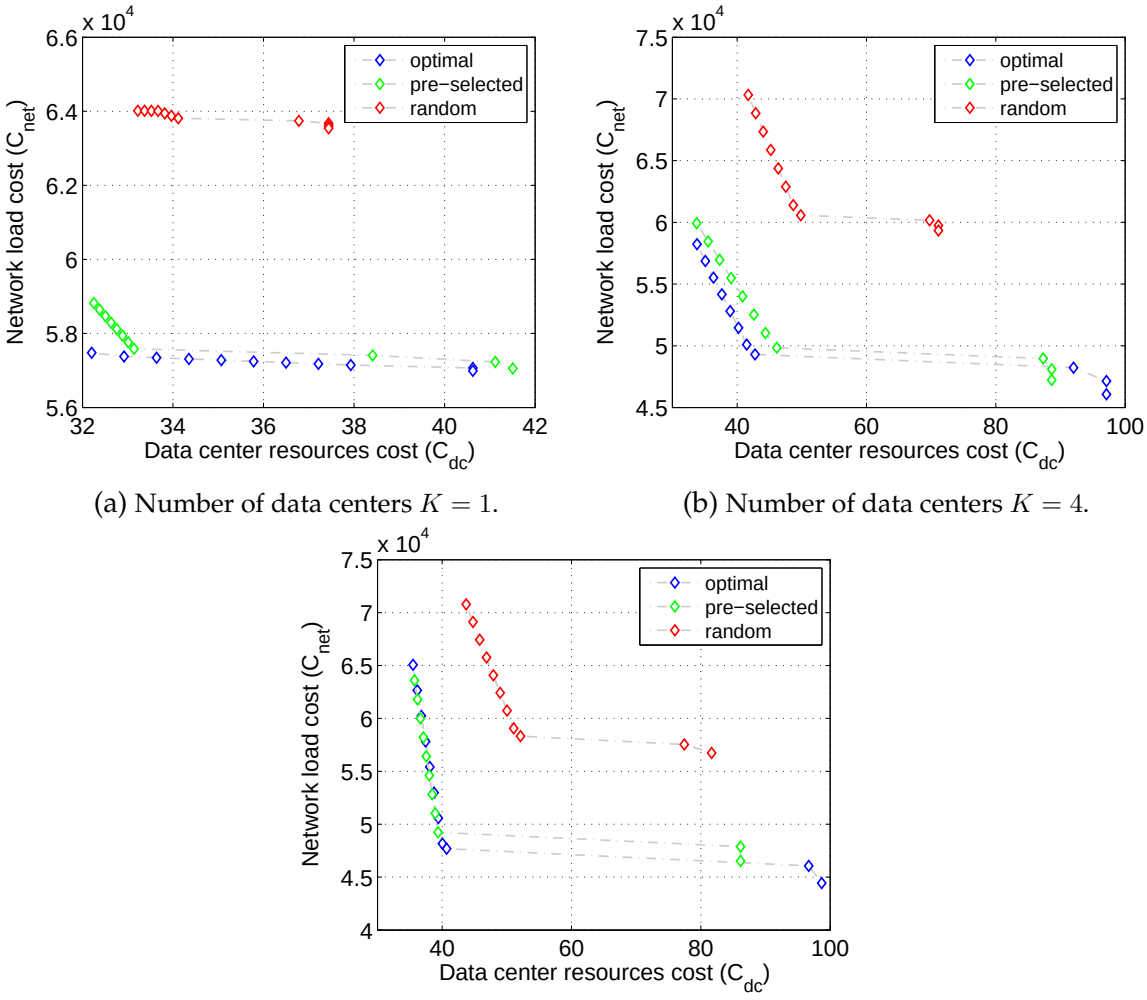
(a) Number of data centers $K = 1$.

(b) Number of data centers $K = 4$.

(c) Number of data centers $K = 8$.

Figure 4.12: Pareto frontier for the network load cost (C_{net}) and data center resources cost (C_{dc}) for the **US** topology, comparing the solutions of the optimal multi-objective model with data center locations **pre-selection** and with **random** data center locations.

center that could satisfy the data and control latency requirements, thus, find more pareto solutions for the network load cost and data center resources cost. With a distributed data center at $K = 4$, the pareto solution with $\lambda = 0.7$ provides an overhead of 6% to the optimal network load cost at $\lambda = 1$ and an overhead of 11% compared to the optimal solution for the data center resources cost at $\lambda = 0$. Considering more distributed data centers with $K = 8$, the pareto solution at $\lambda = 0.8$ offers a trade-off of 11% increase in the optimal network load cost while an increase of 14% in terms of the data center resources cost is offered.



(a) Number of data centers $K = 1$.

(b) Number of data centers $K = 4$.

(c) Number of data centers $K = 8$.

Figure 4.13: Pareto frontier for the network load cost (C_{net}) and data center resources cost (C_{dc}) for the **German** topology, comparing the solutions of the optimal multi-objective model with data center locations **pre-selection** and with **random** data center locations.

4.6.2 Gain from Data Center Locations Pre-selection for the Multi-objective Model

We discuss the evaluation of our proposal of data center locations pre-selection for the multi-objective model as explained in Sec. 4.4.3. The pre-selected data center locations are a combination of the resulting locations from the solutions of the single objective models, i.e., network load cost model and data center resources cost model. Let us consider an example with a number of data centers $K = 4$. The solution of the network load cost model, with 4 data centers, gives 4 optimal data center locations that minimize the network load cost. Similarly, 4 optimal data center locations are given by solving the data center resources cost model with 4 data centers. Two

data center locations are selected arbitrarily from the given solutions of each single objective, respectively. The pre-selected data center locations form the input set to the multi-objective optimization model.

Note that in case of a centralized data center $K = 1$, an arbitrary location among the two resulting data center locations from the solution of the two single objectives is pre-selected. The evaluation focuses on the solution optimality and how much it is impacted by the pre-selection, since the size of the input data center locations set would be $|C| = K$ instead of $|C| = |V|$, i.e., all graph nodes. The evaluation also focuses on how much does the pre-selection improve the run time of the multi-objective model.

4.6.2.1 Optimality Gap with Pre-selection

Fig. 4.12 and Fig. 4.13 illustrate the pareto frontier evaluation for the optimal multi-objective model compared to the multi-objective model with pre-selection, for the US and German topology. We also evaluate our proposed pre-selection, based on the solutions of the single objectives, to a random pre-selection. The random pre-selection represents the case where an operator already has fixed locations for the data centers and is solving the multi-objective model for the given locations. The optimality gap is the difference between the three evaluation cases at each pareto solution. We demonstrate the optimality gap at a number of data centers $K = (1, 4, 8)$ in order to investigate the impact of centralizing or distributing the data center infrastructure.

For both topologies, we could observe that the proposed pre-selection results in pareto optimal solutions with a minimal gap compared to the optimal solutions for the evaluated number of data centers $K = (1, 4, 8)$. For instance, at a number of data centers $K = 4$ for the US topology, shown in Fig. 4.12b, the maximum gap for a pareto solution with pre-selection is 2% in terms of the network load cost and 6% in terms of data center resources cost. This means that the pre-selection, based on the knowledge from the selected locations of the single objectives, can be used to reduce the problem's complexity while achieving a minimal optimality gap. We could also observe that the optimality gap with pre-selection decreases while adding more data cenetrs, i.e., moving from a centralized to a distributed data center infrastructure.

On the other hand, there is a significant optimality gap with the random pre-selection, i.e., given by the operator, compared to the optimal solutions. This observation holds for both topologies as well as for all demonstrated number of data

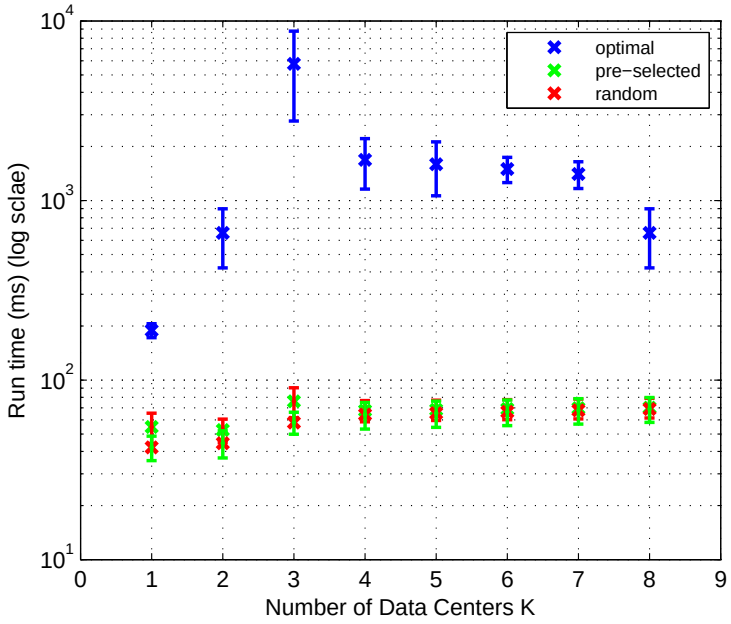


Figure 4.14: Run time comparison for the multi-objective model with and without data center locations pre-selection as well as with random pre-selection for the **US** topology.

centers $K = (1, 4, 8)$. This shows the impact of the data center locations on the resulting optimal cost. Additionally, it supports the importance of the joint placement of the data center infrastructure while solving the placement of the network paths for the dimensioning of a next generation mobile core network with the optimal cost.

4.6.2.2 Run Time Improvement with Pre-selection

Fig. 4.14 and Fig. 4.15 illustrate the average run time for the optimal multi-objective model compared to the multi-objective model with pre-selection, for the US and German topology, respectively. The run time is also evaluated for the multi-objective model with random pre-selection. For the US topology, we could observe that the pre-selection could significantly improve the average run time of the multi-objective model, e.g., at $K = 3$, from the order of several seconds to the order of tens of milliseconds. For the German topology, it could improve the run time from the order of hundreds of milliseconds to tens of milliseconds as well. The proposed pre-selection for the data center locations enables operators to use the multi-objective model for online cost optimization, while keeping a minimum gap to the optimal cost. The pre-selection also allows the multi-objective model to scale further for bigger core topology instances or more traffic demand sets.

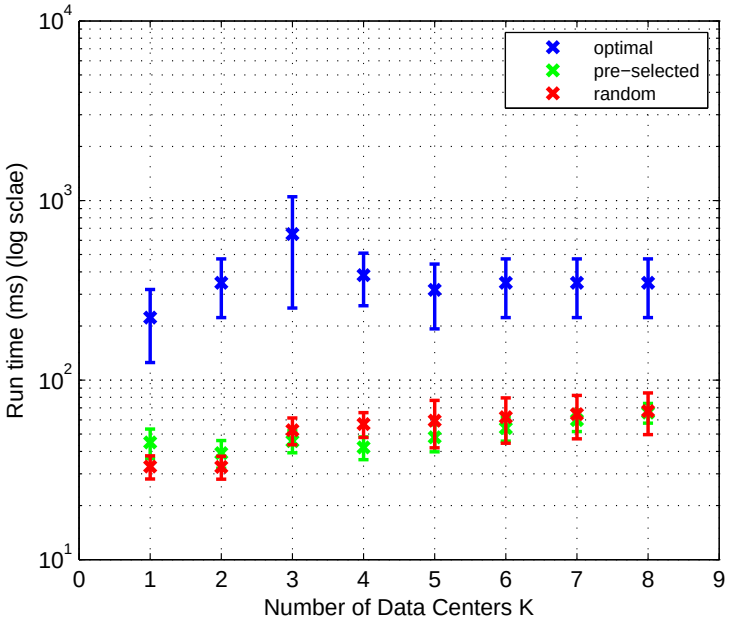


Figure 4.15: Run time comparison for the multi-objective model with and without data center locations pre-selection as well as with random pre-selection for the **German** topology.

4.7 Discussion and Outlook

In this chapter, we propose three dimensioning and planning models that aim at finding the optimal cost for a mobile core network based on both SDN and NFV concepts. The models provide the optimal cost solution regarding: a) the optimal placement of data centers, which host the VNFs and SDN controllers, b) the optimal mapping of VNFs and controllers to each data center, and c) the number and placement of the special purpose SDN+ switches. We introduce two single objective models that aim at minimizing the network load cost and the data center resources cost, respectively. Additionally, a pareto optimal multi-objective optimization model is proposed that results in pareto optimal solutions for the network and data center resources cost factors. Finally, we propose a data center locations pre-selection for the multi-objective model based on the solutions of the single objectives. All proposed models consider the data and control plane latency as key performance requirements for the mobile core network.

An extensive evaluation is presented comparing the proposed models in terms of the network load cost and the data center resources cost. Trade-offs between the two single objective models are observed, in terms of the optimal network load cost and data center resource cost. The multi-objective model results in pareto optimal

solutions where a balance between the two cost factors can be achieved. Additionally, solving the multi-objective model with the proposed data center locations preselection shows a significant improvement to the run time while keeping a minimal gap compared to the optimal pareto solutions of the multi-objective model. For future work, additional cost factors can be considered for the optimization models such as the cost of the SDN+ switches or the inter-data center links. The set of data centers locations could be extended to arbitrary locations on the core network topology, i.e., not the same locations as the SGW and PGW. Furthermore, the challenges of the joint co-existence of SDN and NFV mobile core functions can be investigated, e.g., orchestration and state distribution.

Chapter 5

Flexible, Reliable and Dynamic SDN Virtualization Layer

In this chapter, we explore Network Virtualization (NV) as a fundamental ingredient of the next generation mobile core network architecture. As discussed in the previous chapters, we consider a core network architecture that combines both concepts of SDN and NFV. While NFV provides virtualization for network functions, NV can provide virtualization for the the physical SDN network in order to provide virtual SDN network slices (vSDN). As proposed for the future 5G mobile network, slicing of the physical network resources enables the management of virtual network resources individually according to the demands of the network traffic. Furthermore, virtual SDN networks provide the capability to cope with the dynamics in Network Function Virtualization (NFV) use cases, where virtual SDN slices can dynamically interconnect virtual network functions according to the functions' demands. In this chapter, we discuss the virtualization of Software Defined Networks as a solution to provide logical programmable slices to different tenants that can share the same physical infrastructure. This chapter is based on our work presented in [79], [80] and [81].

5.1 Introduction and Motivation

In the previous chapters, we have discussed the application of virtualization on network functions, i.e., NFV concept, where network functions can benefit from IT virtualization and run as virtual software on commodity servers located in data centers. In this chapter, we look into the application of virtualization on the SDN network,

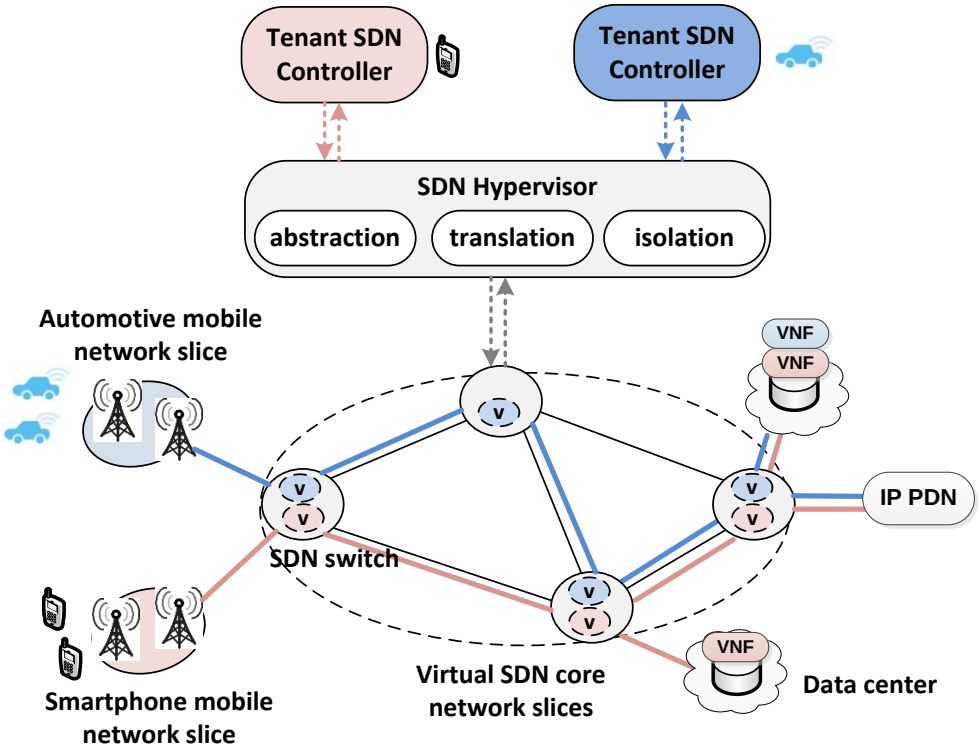


Figure 5.1: Mobile virtual SDN core network slices for different tenants sharing the same physical infrastructure, e.g., smartphone and automotive.

i.e., SDN network resources of nodes and links, by applying the concept of Network Virtualization (NV) to the SDN network as shown in Fig. 5.1. The figure shows an example of two mobile network tenants, e.g., a smartphone operator and an automotive manufacturer, which operate a mobile core network slice and share the same physical infrastructure. This way we could provide end-to-end logical mobile core networks that comprise of virtual network functions as well as virtual SDN networks in order to offer more flexibility for the next generation core network.

Virtual SDN Networks would allow multiple tenants to share the same physical infrastructure by acquiring virtual resources according to their services' demands. One main driver for network slicing is cost reduction, as a tenant can avoid huge investments in network infrastructure and acquire virtual resources instead. Virtual SDN networks enable faster network and service deployment which reduces the time to market. From the physical infrastructure operator point of view, allowing tenants to share its infrastructure would result in higher efficiency in utilizing its network resources and maximize its profits. Combining SDN and NV introduces

network programmability and automation since each tenant can use its own controller to operate its virtual SDN network.

In order to virtualize SDN networks, a hypervisor layer is needed to virtualize the network resources comparable to IT virtualization of servers [82]. The SDN hypervisor layer lies as an intermediate layer between the tenants' SDN controllers and the SDN physical infrastructure as shown in Fig. 5.1. The SDN hypervisor provides three main functions in order to virtualize an SDN network. First, the hypervisor is responsible for abstracting the physical network resources, including network topology as well as link and node resources, into virtual resources according to each tenant's demand. Second, the SDN hypervisor intercepts and translates the control messages that are exchanged between the tenants' controllers and their virtual SDN networks. Third, in order to provide guarantees on the performance of each virtual SDN, the hypervisor is responsible for providing isolation and protection for the acquired resources by each tenant.

There are several existing SDN hypervisors that are presented in Sec. 5.2, however, existing proposals have several limitations. Existing SDN hypervisors run either as software or they require special networking hardware, which in turn limits their deployment flexibility. Additionally, since SDN decouples the control from the data plane, an integral part of the performance of SDN is the realization of the control plane. Most existing hypervisors focus on data plane isolation only, while less attention has been devoted to the isolation of the control plane, which reduces the reliability of the proposed solutions. In particular, for virtual SDN networks, the performance of the control path, i.e., the physical links and nodes, may have an impact on the performance of the virtual SDN network. As the hypervisor is placed between virtual SDN controllers and virtual SDN networks, the hypervisor itself may influence the performance of the virtual SDN networks. Finally, current hypervisor solutions have no mechanisms to change their deployment on run time. For virtual SDN networks, the SDN virtualization layer is required to adapt to the new tenant demands, e.g., change the virtual topology or change the location of the tenant's controller. This means that the virtualization layer might have to scale, by increasing the number of the hypervisor instances or changing their location on run time, in order to adapt to the changes of the virtual SDN network demands.

In this chapter, we focus on addressing the aforementioned limitation of existing SDN virtualization solutions. We introduce an SDN virtualization layer, what we call

HyperFLEX, that offers deployment flexibility, reliability in terms of control plane isolation and dynamic deployment on run time. There are several other aspects that have been explored out of the scope of this thesis as part of the HyperFLEX project, e.g., SDN hypervisor modeling and placement [83] [84].

5.2 SDN Virtualization State-of-the-Art

There are several existing SDN hypervisor solutions to provide network virtualization for SDN networks. We are concerned with their proposed architecture, hypervisor platform and isolation mechanisms. We review several state-of-the-art hypervisors as illustrative examples to these different features. More examples are presented in detail in our survey on network hypervisors for SDN virtualization in [6].

FlowVisor [82] has been introduced as one of the earliest architectures to provide virtualzation for SDN, based on OpenFlow. It plays the role of a logically centralized hypervisor layer between the controllers of virtual tenants and the physical SDN infrastructure, although it can also be deployed in a hierarchical fashion. It realizes the network abstraction by providing a mapping between the controller and its allocated virtual network resources. FlowVisor is implemented in software, hence it is a purely software solution. Regarding isolation, FlowVisor provides flow table isolation by maintaining a maximum number of flow entries for each vSDN at their assigned SDN switches. It is also mentioned that FlowVisor has the capabilities to associate certain network QoS guarantees with a virtual slice, e.g., data plane bandwidth guarantees using the VLAN priority field.

Advanced FlowVisor or ADVisor [85] is developed as an extension of FlowVisor, in which a software proxy or an enhanced abstraction unit is added to improve the virtual resources abstraction. Hence, similar to FlowVisor, it is also a software solution. No control plane isolation mechanisms were proposed by this work.

The developers of VeRTIGO [86] take the virtual network abstraction a step further. It is also an extension of FlowVisor, hence a software solution as well. VeRTIGO allows the vSDN controllers to select the level of virtual network abstraction that they require. The abstraction feature can range from providing the whole set of assigned virtual resources, with full virtual network control, to abstracting the whole vSDN to a single abstract resource, where the network operation is carried out by the hypervisor while the tenant focuses on service deployment on top. While

this feature adds more flexibility in provisioning vSDNs, it also increases the hypervisor complexity and does not address the control plane isolation limitations of the previous solutions.

AutoSlice [87] thrives to improve the scalability of a logically centralized hypervisor and, thus, it distributes the hypervisor's workload. For that purpose, the physical infrastructure is segmented into non-overlapping SDN domains, while the hypervisor is split into a management module and multiple controller proxies, one for each SDN physical domain. The virtual resources assignment is done by the management module and stored at each proxy, where the proxies in turn carry out the translation of the messages exchanged between the vSDN controllers and the physical infrastructure in their respective domain. It is still a solution targeting software deployment. Regarding isolation, a partial control plane offloading could be offered by distributing the hypervisor over multiple proxies. However, within each SDN domain, the control plane isolation problem still persists.

A distributed virtualization architecture for vSDN has been introduced in [88], by placing translation units in every data plane SDN switch at the physical infrastructure. Hence, it is a solution targeting network elements. A virtualization controller supplies the translation units with the set of policies and rules, which include the assigned label, flow table and port for each vSDN slice. Such distributed architecture aims at minimizing the additional overhead of a logically centralized hypervisor by providing a direct access from the vSDN controllers to the physical infrastructure. However, processing complexity is added to the data plane physical infrastructure, which jeopardizes the data plane elements in case of overloading its control channel that could result in data plane performance degradation. Additionally, the data plane physical switches have to be modified to include a translation unit, which adds a higher complexity and eventually cost. Finally, there are still no isolation or protection functions proposed for the SDN control plane.

In summary, existing SDN hypervisor solutions offer a limited deployment flexibility. The isolation of the SDN control plane has not been addressed, while there are no solutions for dynamic deployment and scaling on run time. Addressing these limitations and providing solutions are the main focus of this chapter.

5.3 HyperFLEX SDN Virtualization Layer

In order to address the shortcomings of existing architectures, we introduce HyperFLEX, an SDN hypervisor layer based on flexible hypervisor function allocation. The design goals of HyperFLEX are flexibility, reliability in terms of control plane isolation and dynamics. In order to provide flexibility, HyperFLEX relies on the decomposition of the hypervisor into functions that can be hosted on different platforms. The hypervisor functions' realization and placement adapts flexibly to the performance of the target host platform and to the virtual SDN network demands. Thus, HyperFLEX can work in different operation modes, which may provide performance gains for different networks. Additionally, HyperFLEX is the first virtualization solution for SDN networks that provides control plane isolation. We first outline the design challenges of a hypervisor considering control plane isolation. Next, we introduce HyperFLEX's architecture and its modes of operation. Finally, we address the control plane isolation function and provide examples of its realization in our proposed architecture.

5.3.1 Design Features

In the following, we point out important aspects that a hypervisor for virtual SDN networks should fulfill for the virtual resource and hypervisor functions. These aspects are based on the design goals for NV given in [89] and [18].

Flexibility One of the main properties of NV, in general, is the ability to cope with the network dynamics, where virtual networks provide the flexibility to adapt to the changes in the network, e.g., time-varying traffic [90] or network failure [91], through the migration of virtual nodes or the assignment of virtual link capacities. In vSDNs, network dynamics are not only to be observed at the physical infrastructure, i.e., SDN data plane, but also at the tenants' vSDN controllers, i.e., SDN control plane, as their control plane can dynamically adapt to the changes in the data plane, for example by relocating the controllers, to provide the desired overall performance, e.g., control plane latency. Hence, a vSDN hypervisor layer is also required to provide the flexibility to adapt to the network dynamics at the SDN physical infrastructure as well as the vSDN control layer.

Isolation Classical NV requires resource isolation and protection between co-existing virtual networks on the physical data plane infrastructure. Resource guarantees are important in order to provide the tenants with a reliable operation. Protection means that operations of multiple tenants should not affect each others' performance, i.e., there is no performance degradation due to sharing the network among multiple tenants. Again, in vSDN networks, isolation and protection are required on the physical infrastructure for the data plane traffic as well as for the SDN control plane. Since the SDN control plane path, between the vSDN controllers and the physical SDN infrastructure, includes the hypervisor layer, it is important to provide isolation and protection mechanisms for the hypervisor's processing resources among the different vSDN tenants.

Dynamics Each tenant of a vSDN could use its own controller to manage its vSDN topology. A hypervisor architecture for a virtualized infrastructure should provide a high degree of scalability in terms of control plane access to vSDN topologies, i.e., the physical connection for the tenants to the hypervisor. In large-scale networks, a centralized hypervisor may lead to bottlenecks or to significant waste of network resources. The tenants' controller, for instance, may send messages over long network paths before being discarded due to network over-utilization or hypervisor exhaustion. Additionally, vSDN controllers might be actually managing disjoint sets of physical resources. In such case, having a dynamic SDN virtualization layer would lead to an improved control performance. The SDN virtualization layer should have the capability to adapt on run time to the changing virtual SDN network demands. Such adaption can be in the form of instantiating new hypervisor instances or changing the location of deployed hypervisors.

5.3.2 HyperFLEX Architecture

The first main concept of HyperFLEX relies on the decomposition of the hypervisor virtualization functions required to realize vSDN, where having a functional granularity achieves a more tailored operation. This approach of function decomposition is seen to be more optimal in terms of resource efficiency. If a distributed hypervisor layer is needed, individual virtualization functions could be distributed such that a tailored virtualization layer is achieved, instead of duplicating the whole hypervisor instance along the network.

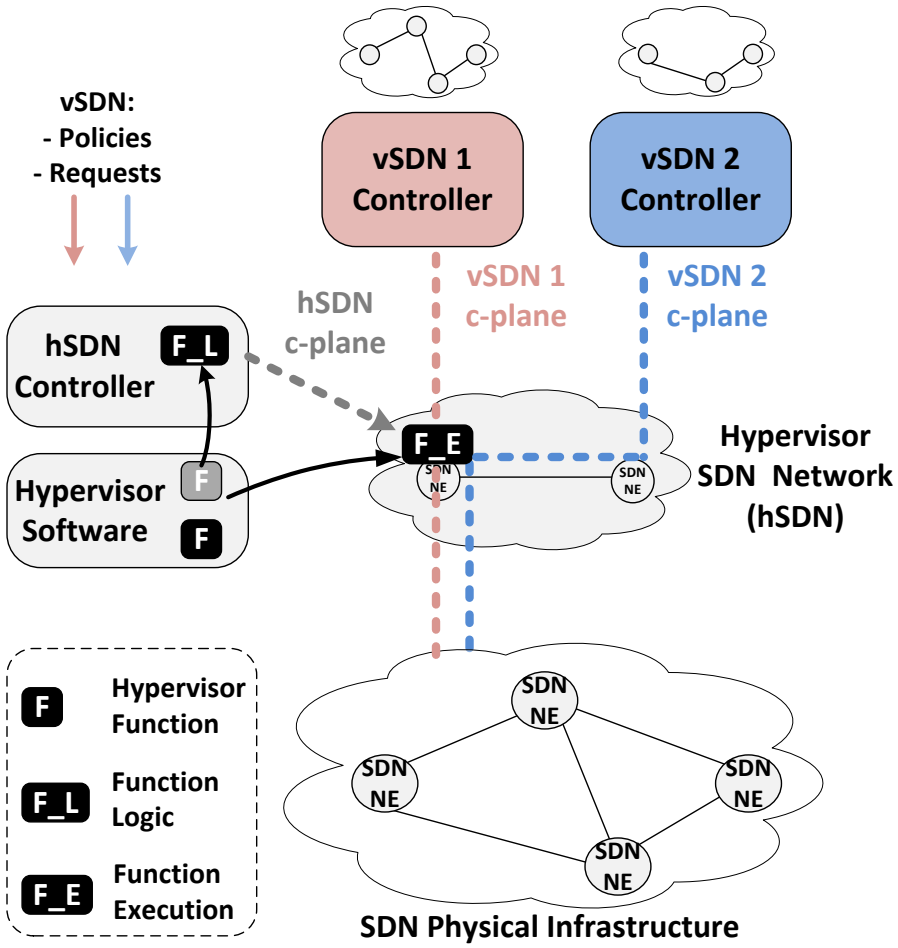


Figure 5.2: HyperFLEX architecture that consists of hypervisor software as well as hypervisor SDN (hSDN) network and controller.

The second main concept is the hosting platform of the SDN hypervisor layer. The hypervisor functions can be implemented in software. However, since in large-scale networks it is reasonable to assume that multiple network elements are interconnecting the hypervisor with the tenants' SDN controllers and with the physical network infrastructure. We propose using the available processing functions, e.g., traffic shapers or packet inspection, on these interconnecting network elements to execute hypervisor functions. Thus, the virtualization layer can be spanned on software as well as SDN network elements, which we call the hypervisor SDN network (hSDN) as illustrated in Fig. 5.12. This hSDN network is operated and controlled by a hypervisor SDN controller. The proposed architecture has the flexibility to allocate the virtualization functions, on a per-function basis, among the software and the SDN network elements of the hypervisor network.

In case a hypervisor function is delegated to the network elements, it needs to be transformed to its corresponding SDN data plane execution function (F_E) at the hSDN network element and a function control logic (F_L) residing at the hSDN controller, as shown in Fig. 5.12. Additionally, the hSDN controller is responsible for managing the delegated functions as well as maintaining the connectivity between the vSDN tenants and the physical infrastructure.

A hypervisor consists of multiple virtualization functions that could be potentially delegated or even better suited to heavy-load processing functions at the network elements. As for example, the translation function that transparently maps the vSDN control messages to their physical resources. The translation could be done by the network elements according to their packet inspection capabilities and packet modification performance. Another example is the policy enforcement function, which ensures that a vSDN controller has access only to its acquired virtual resources, or the abstraction function, which determines the granularity of information to be exposed to the vSDN controller. These policies can be translated to rules enforced by the network elements. Additionally, the isolation functions, that slices the data and control plane resources among the different vSDNs, could benefit from the traffic shaping mechanisms of the network elements.

A main advantage of this proposed architecture is distributing the load of performing the virtualization functions between the software and the hypervisor network elements. This implies that the resources of the servers, hosting the hypervisor software, are extended by the resources of the SDN hypervisor network, which has a direct impact on improving the virtualization performance. It also adds another flexibility dimension by providing the possibility to dynamically change the functions placement during operation, since both options are present, namely the software and the SDN hypervisor network. Such flexibility is crucial, for example, in case the software or the network element is running out of resources and is not capable of offering the required performance for a certain virtualization function. Such function could be offloaded and executed at another software instance or network element.

It is intended to separate the SDN hypervisor network from the SDN physical infrastructure to isolate the virtualization functions from the data plane. Hence, through this separation, we could eliminate any impact that could be imposed by virtualization on the data plane performance.

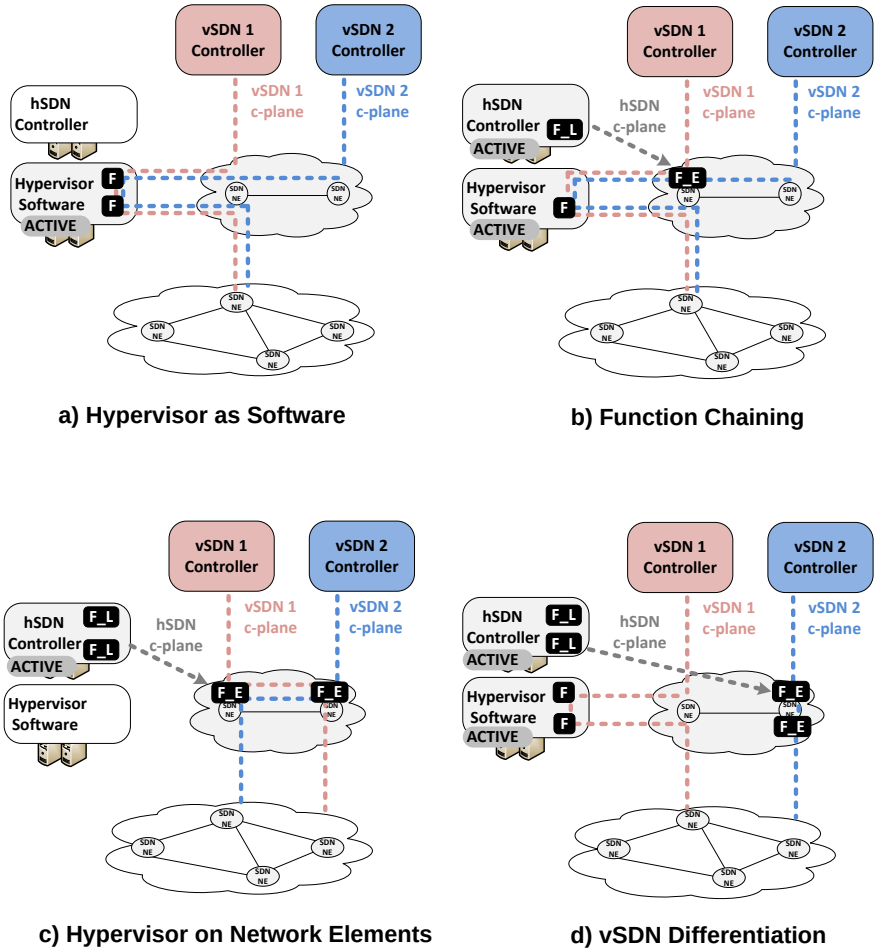


Figure 5.3: Alternative modes of operation of HyperFLEX.

5.3.3 HyperFLEX Modes of Operation

Hypervisor in Software The flexibility in HyperFLEX's architecture provides the option to operate a classical virtualization layer in software as a centralized or distributed hypervisor. This mode of operation, shown in Fig. 5.3a, is adequate in case a powerful and a wide-spanned servers infrastructure is available to host the hypervisor software. Alternatively, it suffices in case the vSDN tenants are geographically close to the hypervisor or in case of a small-scale SDN physical infrastructure, which allows the performance requirements by the vSDN tenants to be met, i.e., SLAs which include latency or bandwidth guarantees on the data and control plane. This mode is selected given that the interconnecting network elements can not execute the virtualization functions due to the functions' complexity or the network elements' limited resources.

Function Chaining This mode of operation, shown in Fig. 5.3b, is the intermediary solution between running hypervisor functions solely in software or exclusively on network elements. As it exploits the functional-granularity and the processing resources of the hSDN network, it assigns the virtualization functions to the software or network elements, forming function chains that realize the hypervisor layer. The hSDN controller has to ensure that the correct chains are set along the hSDN network including the hypervisor software. This mode is seen beneficial in case some functions are too complex or not possible to realize by the network elements, e.g., DPI or stateful forwarding. Such virtualization functions are then implemented in software. The function placement decision is based on the available network resources, network elements' capabilities and vSDN SLA requirements. These parameters can be quite variant, thus making this flexible operation very advantageous.

Hypervisor on Network Elements Realizing the hypervisor functions on network elements enables the full deployment of the hypervisor layer on SDN network elements, as illustrated in Fig. 5.3c. This could help bringing the virtualization functions nearer to the vSDN tenants or the physical infrastructure, hence improving the scalability and performance. It can also be seen as an enabler to achieve infrastructure savings, as servers are only needed to host the hSDN controller. This mode fully exploits the available processing functions in the SDN network elements to host the hypervisor functions. However, such network elements might not provide all hypervisor functions out of the box, according to their SDN implementation. Therefore, this mode of operation adds the complexity of perhaps modifying the network elements and transforming the hypervisor function to an SDN realization. In addition, this mode adds a management and control overhead to the hypervisor layer to handle the transformed functions.

vSDN Differentiation Another advantage inherited from the flexible operation on a functional-granularity is the possibility to allocate hypervisor functions to vSDN tenants, thus realizing vSDN differentiation. Each hypervisor function can be instantiated on different locations within the network to provide different performance quality for different vSDN tenants. As shown as an example in Fig. 5.3d, vSDN1 is served by the functions running in software while vSDN2 is using the functions hosted by the hSDN network. This elasticity enables more differentiated services and agreements, which impacts the business position of the hypervisor provider.

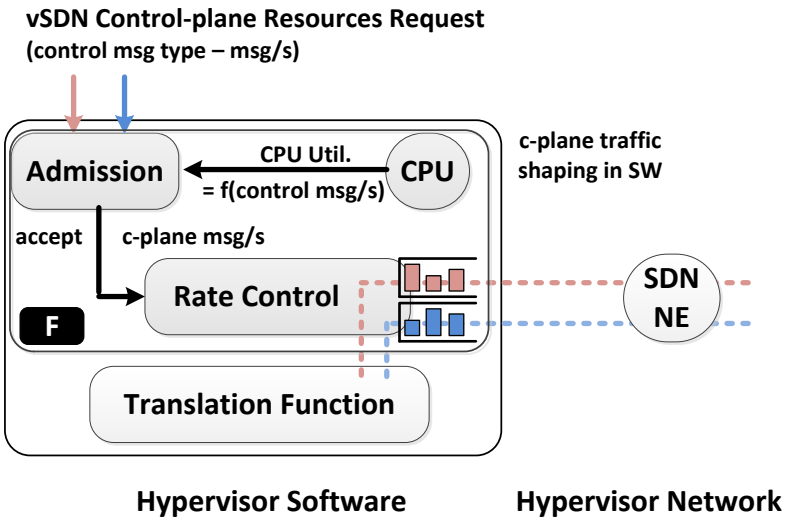


Figure 5.4: Control plane isolation function in software.

5.3.4 Control Plane Isolation

As mentioned in Sec. 5.3.2, the hypervisor layer is composed of multiple virtualization functions, forming a function chain that is traversed by the SDN control plane between the vSDN controllers and the SDN physical infrastructure. The control plane isolation function is intended to ensure guarantees on each of the virtualization functions shared among multiple tenants and to protect the hypervisor's processing resources, e.g., CPU or memory, from being exhausted by one of the vSDN tenants. If we take the translation function as an example, its resources can be exhausted if a vSDN controller is sending excessive *FLOW_MOD* messages, downstream to its virtual SDN switches, or if a virtual SDN switch is overloading the translation function with *PACKET_IN* messages, upstream to the controller.

Considering a scenario where the translation function is in software and we target, for instance, to isolate and protect the CPU processing resources for this software. Fig. 5.4 shows the control plane isolation function realization in software. The first isolation logic module is the admission module which receives control plane resources requests by each vSDN controller, in the form of requested number of control messages per second from each message type. As the hypervisor's CPU utilization is proportional to the amount of control plane messages to be processed from each vSDN controller, the admission module contains a CPU utilization as a function of the control messages rate, on which it bases its decision to accept the requests. This information is passed to the message counter module which intercepts the control

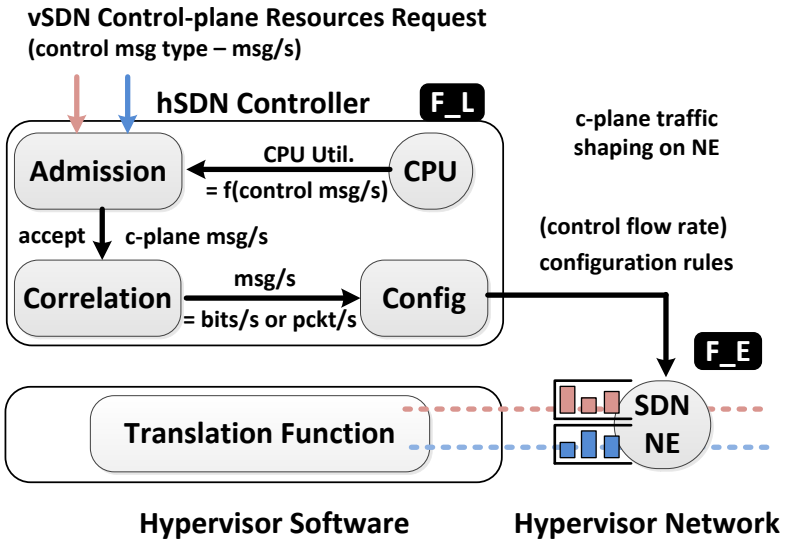


Figure 5.5: Control plane isolation function on network elements.

plane messages and makes sure that the limit is not yet reached before forwarding the control messages to the translation function.

Fig. 5.5 illustrates the function chaining mode, where the translation function is in software while the control plane isolation function is realized on a network element, that is controlled by the hSDN controller. The admission module plays the same role, however currently at the hSDN controller. The correlation module translates the messages rate into a rate that a network element could enforce, e.g., bits per second or number of packets per second. The rate information is passed to the configuration module. This module configures the isolation function at the network element, which can be, e.g., a port traffic shaper or an OpenFlow meter.

This way, the control plane messages monitoring and isolation would be done by the hypervisor network element before forwarding the messages to the software to be translated, which exploits the processing power of the networking elements. It also protects and improves the utilization of the server, hosting the hypervisor software, as well as the hypervisor network since the control plane traffic is shaped at the network point of entry according to the assigned control plane shares of each vSDN controller.

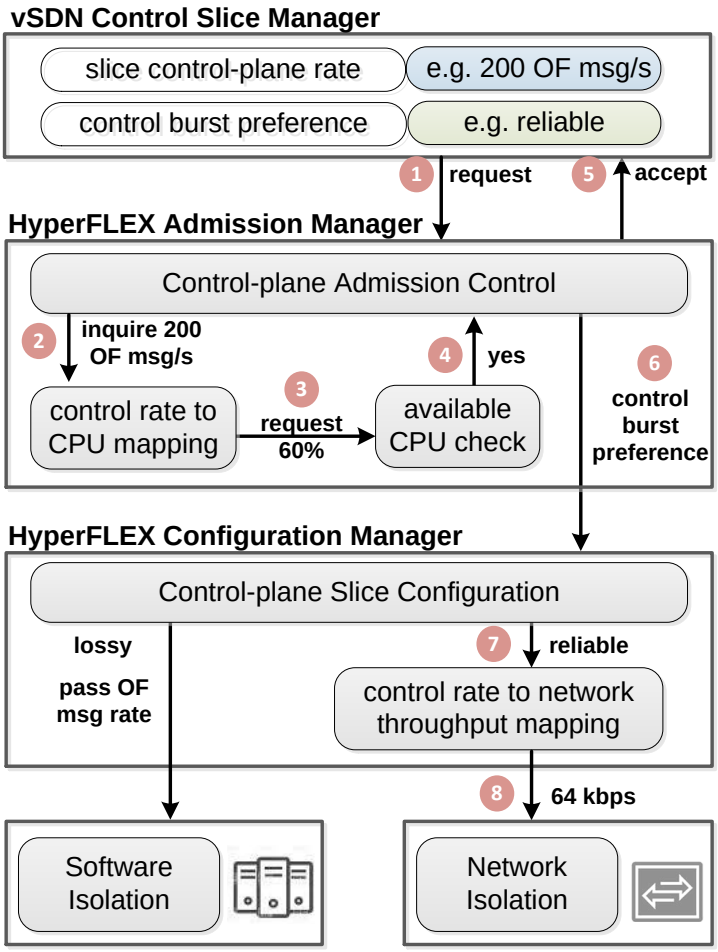


Figure 5.6: Virtual SDN control slice request, admission and configuration.

5.3.5 Control Plane Admission Control

After discussing the control plane isolation functions, we introduce the admission control framework for the control plane slices used by HyperFLEX, as shown in Fig. 5.6. A virtual SDN network tenant can request a control slice via the vSDN control slice manager. The requested slice is expressed in terms of OpenFlow (OF) message rate and its preference for control slice isolation mechanism. As explained in the previous section, HyperFLEX offers two mechanisms for control plane isolation, either in the hypervisor's software or on the hypervisor network, i.e., switches between tenants and hypervisor.

HyperFLEX Admission Manager receives the vSDN control slice requests. An admission control triggers the validation of the vSDN requests and acts as an interface to accept or reject requests. The validation starts with a function that maps the

requested OF rate to hypervisor resources, e.g., CPU resources. The mapping of OF rate to CPU utilization is based on hypervisor benchmark measurements. Next step is to check whether the requested CPU resources are sufficient, which determines whether to accept or reject the vSDN control slice request.

In case hypervisor CPU resources are available, i.e., the request is accepted, the admission control triggers the slice configuration, which configures the isolation mechanism based on the vSDN request preference. The software isolation directly takes the OF rate as an input parameter. The network isolation requires a further step of mapping the OF rate to a network throughput parameter to be configured on the network. vSDNs can change their isolation preference on run time, thus automation can be provided for SDN control plane virtualization with parameter self-configuration and an adaptive operation based on vSDN requests.

We could observe that each isolation mechanism results in different behavior in case of exceeding the requested control rate. With software isolation, in case of control plane traffic exceeding a virtual SDN network demand, e.g., due to control bursts, over load, or misconfiguration, the software module drops OF messages exceeding the requested rate prior to processing. In contrast, network isolation throttles the control connection throughput according to the requested control rate. The software isolation results in control message loss, i.e., "fast and lossy". Alternatively, the network isolation deploys a network policy that drops network packets. As OpenFlow uses TCP on the control channel, this leads to retransmissions and avoids any control loss, i.e., "reliable". However, retransmissions can significantly impact the tenant's control performance and increase the control latency. Both mechanisms lead to an efficient control isolation. It is the tenant's choice to decide on the control isolation mechanism. This control mechanism determines the control behavior, either "lossy" or "reliable", for control traffic exceeding the requested control rates.

5.3.6 Results and Evaluation

In this section, we provide the evaluation for the proposed control plane isolation mechanisms on the data plane as well as on the control plane. The evaluation demonstrates the trade-off between the software control isolation and the network control isolation. Additionally, the response time needed to change between the two options upon a tenant's request is shown.

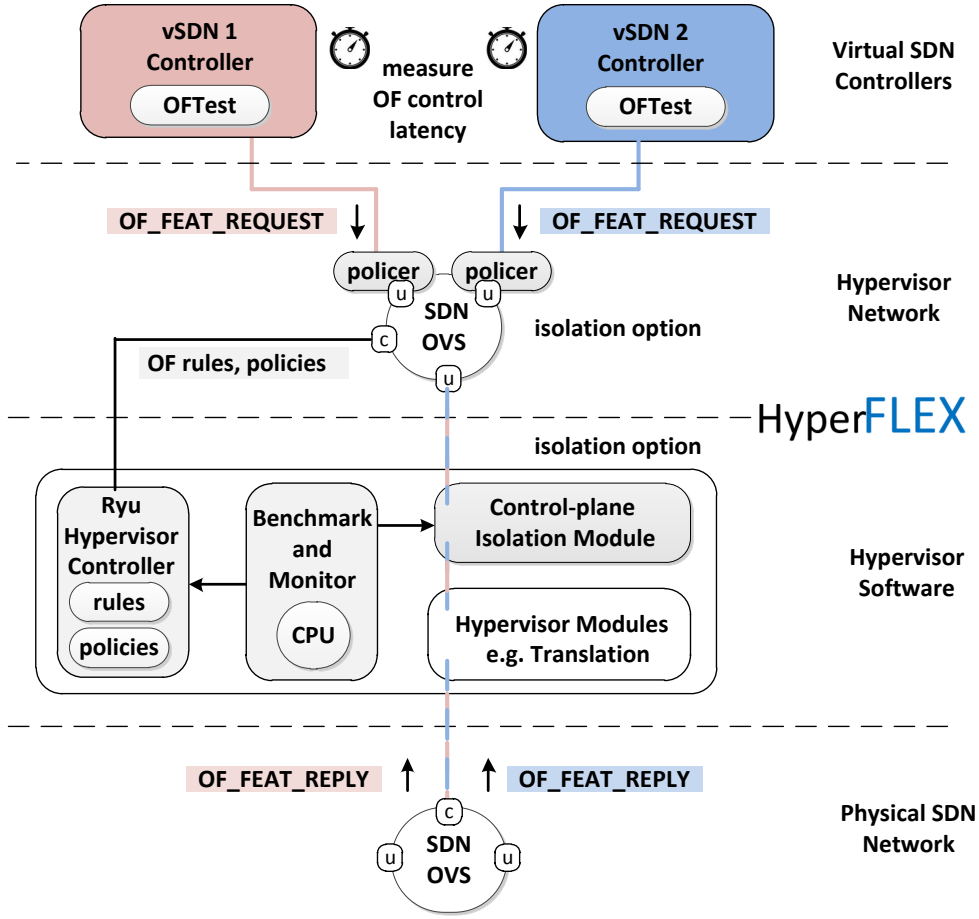
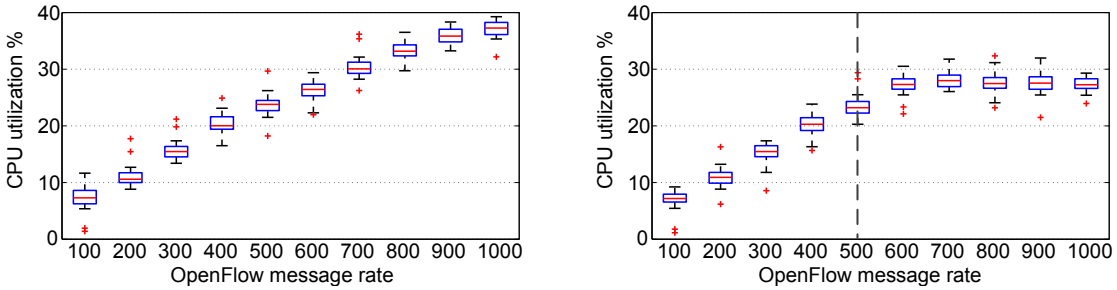


Figure 5.7: HyperFLEX testbed and evaluation setup for vSDN control plane performance.

5.3.6.1 Control plane Isolation Performance

First, measurements in an OpenFlow-based SDN testbed are conducted in order to evaluate a prototype implementation of HyperFLEX. In detail, we demonstrate a proof-of-concept of the flexible function allocation and control plane virtualization. We investigate the performance of different setups for the hypervisor control plane isolation function providing performance guarantees, which means that there should not be any cross effect between virtual SDN networks. A cross effect may result in performance degradation of one or more virtual SDN networks. Such performance degradation could be, for example, a longer flow setup time due to additional latency on the control paths. We consider the delay of OF control messages as a performance metric for all following evaluations. The overall goal is to guarantee the control message delay for each virtual SDN network.



(a) Hypervisor CPU utilization **without** isolation and protection on network.

(b) Hypervisor CPU utilization **with** isolation and protection on network.

Figure 5.8: Hypervisor CPU utilization versus different OF messages rates with and without network control plane isolation.

The measurement setup is shown in Fig. 5.10. Two tenant controllers, namely vSDN-C1 and vSDN-C2, are running on a Virtual Machine (VM). For testing purpose, both controllers are based on *OFTest* [92], a framework and test suite for OF switches. *OFTest* is extended to generate an average constant rate of OF messages to test the hypervisor performance and control plane virtualization under varying control plane traffic. Both vSDN controllers are connecting to a data plane switch of the SDN hypervisor network. The data plane switch is an Open vSwitch (OVS) also running on a VM. An instance of *FlowVisor* [82] is used to provide the core hypervisor functions, e.g., the translation function. To implement the hypervisor controller, a *Ryu* controller [93] is used to control the hypervisor network. Both *FlowVisor* and *Ryu* share one VM. An additional instance of OVS represents the shared physical SDN data plane network, that is running on another VM.

The measurement procedure is as follows. Both vSDN controllers are sending OF_FEAT_REQUEST messages to the shared data plane switch and receive OF_FEAT_REPLY messages back. We evaluate the control plane performance in terms of latency to receive the reply message back from the switch for each request message. All OF messages from both controllers have to be processed by the hypervisor. This produces load on the hypervisor's CPU resources. This CPU load can have a direct impact on the control plane latency. Thus, the CPU utilization of the hypervisor needs to be monitored and isolated for each vSDN.

Hence, before investigating the performance latency for vSDNs, a first measurement is conducted to measure and analyze the impact of the rate of OF messages on the hypervisor CPU. Thus, the CPU utilization of the VM that hosts the hypervisor software is monitored for two setups. In the first setup, no isolation function is

Table 5.1: Scenario configurations with different vSDN OpenFlow rates.

	vSDN-C1 rate	vSDN-C2 rate	soft isolation	net isolation
(uu)	500	500	-	-
(ou)	500	1000	-	-
soft	500	1000	500 per vSDN-C	-
net	500	1000	-	260 *kbps* per vSDN-C

active. In the second setup, the isolation function on the OVS switch of the hypervisor network is activated. The allowed OF message rate is set to 500 OF msg/sec, which corresponds to a traffic rate of 260 *kbps*. Note that this rate holds as long as only one OF message is sent per network packet. In both setups, 100 to 1000 OF_FEAT_REQUESTs msg/sec are sent to FlowVisor. Each run lasts 25 seconds and is repeated 10 times.

Fig. 5.8 shows boxplots for hypervisor's CPU utilization in % according to the rate of OF messages received for both setups. In addition, all figures that are showing boxplots have the corresponding mean values shown as red squares and additionally as values on top. In case no isolation is active, as shown by Fig. 5.8a, the CPU utilization increases linearly with the amount of OF messages sent. The maximum mean value for this setup is $39.3\ \%$. In case of limited CPU resources or shared hypervisor resources, too many network packets containing OF messages can lead to a performance degradation of the hypervisor. In order to avoid CPU over-utilization due to excessive OF messages or packet rate, the network isolation can force the sender of OF messages to aggregate messages. This is additionally marked by the dashed line, as shown in Fig. 5.8b. The dashed line illustrates the limit of 500 OF messages.

While the CPU utilization linearly increases with the message rate from 100 to 500, the rate limiter installed on the hypervisor network throttles the network rate between 500 and 1000 messages. After reaching a maximum mean value of $27.4\ \%$, the rate limiting even decreases the CPU utilization. Thus, the hypervisor could be protected from over-utilization. In order to additionally illustrate the impact of OF messages on the CPU utilization of the hypervisor, the available processing capacity is limited to $40\ \%$ in all following experiments. According to Fig. 5.8, this allows the hypervisor to handle 1000 OF msg/sec.

For the investigation of all operation modes of HyperFLEX guaranteeing control plane isolation, vSDN-C1 and vSDN-C2 are each assigned an equal share of

Table 5.2: Loss rate of OF_FEAT_REQUEST messages for vSDN-C1 and vSDN-C2 in each evaluation scenario.

	under-util	over-util	isolation in software	isolation on network
vSDN-C1	0%	0%	5.8%	0%
vSDN-C2	0%	0%	52.9%	0%

500 OF_FEAT_REQUEST msg/sec. The performance of the control plane isolation is evaluated for the following four scenarios. The configurations of all setups are additionally provided in Table 5.1.

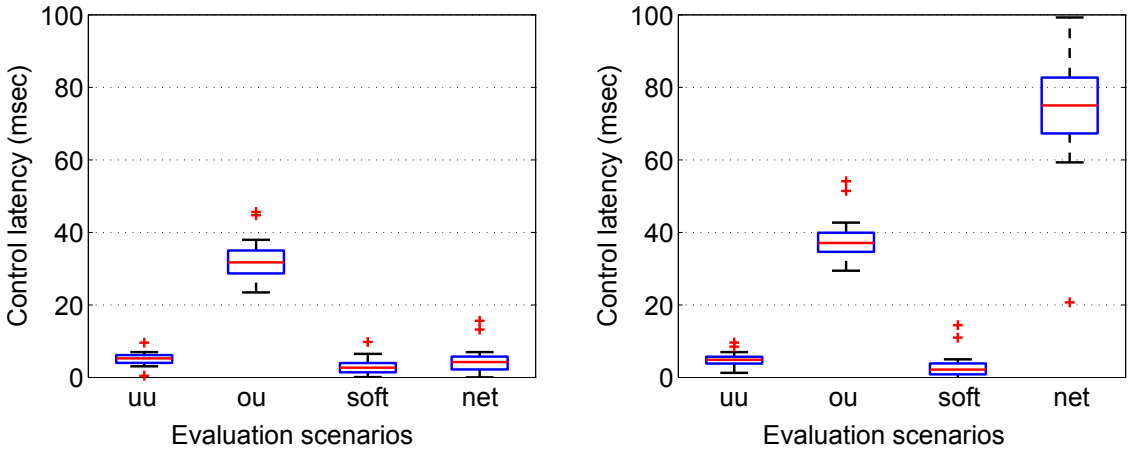
Hypervisor CPU Under-utilization (uu) In this scenario, each of vSDN-C1 and vSDN-C2 is generating its assigned rate of 500 OF_FEAT_REQUEST msg/sec.

Hypervisor CPU Over-utilization (ou): vSDN-C2 exceeds its assigned rate by sending 1000 OF_FEAT_REQUEST msg/sec, while vSDN-C1 keeps sending 500 OF_FEAT_REQUEST msg/sec. The are no control plane isolation mechanisms applied in this scenario.

Control Plane Isolation in Hypervisor Software (soft): vSDN-C2 exceeds its assigned rate in this scenario as well by sending 1000 OF_FEAT_REQUEST msg/sec. However, the control plane software isolation function is activated to monitor the OF messages rate sent by each vSDN controller with a limit of 500 OF_FEAT_REQUEST msg/sec. OF messages exceeding the rate limit are dropped.

Control Plane Isolation on Hypervisor Network (net): In this scenario, control plane isolation is provided by the hypervisor network. Rate policing is activated at the ingress ports for each vSDN controller. The OF messages rate is transformed to bitrate, where the assigned share of 500 OF_FEAT_REQUEST msg/sec corresponds to 260 $kbps$, in case a single OF message is sent per packet.

All scenarios were run for 25 seconds and repeated 30 times in order to get statistical evidence. Fig. 5.9a and Fig. 5.9b show the control plane latency of vSDN-C1 and vSDN-C2, respectively, for the four scenarios via boxplots. Both controllers observe a similar average control plane latency of 4 ms given that the hypervisor's CPU is under-utilized. In case of over-utilization of the hypervisor CPU, which is caused by vSDN-C2 exceeding its assigned control plane message rate, an up to 6 times higher average control plane latency is observed for both controllers. This shows the



(a) Control plane latency of vSDN-C1.

(b) Control plane latency of vSDN-C2.

Figure 5.9: Control plane latency of vSDN-C1 and vSDN-C2 for four evaluation scenarios: hypervisor CPU under-utilization (uu), CPU over-utilization (ou), isolation in software (soft), and isolation on network (net).

cross effect and impact of vSDN-C2 on the control plane performance of vSDN-C1, provided that no control plane isolation is implemented.

With control plane isolation activated in the hypervisor software, control plane latency for both controllers is once more similar to the under-utilization case, thus control plane isolation is achieved. Note that the latency observed by vSDN-C2 is only for the conforming 500 OF messages out of 1000 total messages that are generated per second. The software isolation function, which implements rate limiting on the application layer, i.e., OF layer, drops all OF messages exceeding the limit per second. This results in a loss rate of 52.9% for OF_FEAT_REQUEST messages generated by vSDN-C2, as shown in Table 5.2. Note that there is a minor loss rate of 5.8% observed by vSDN-C1 since the OF message generation is not ideal and might also slightly exceed the assigned 500 OF messages.

In case of control plane isolation activated on the network element, the control plane latency of vSDN-C1 is similar to the under-utilization scenario. However, a much higher control plane latency is experienced by vSDN-C2. Rate limiting on the network element drops TCP packets exceeding the rate limit on the ingress port. In contrast to OF, TCP initiates retransmissions for the dropped packets. Hence, all 1000 OF_FEAT_REQUEST messages are transmitted to the hypervisor and no OF message loss is experienced, as shown in Table 5.2. However, retransmissions result in a higher control plane latency.

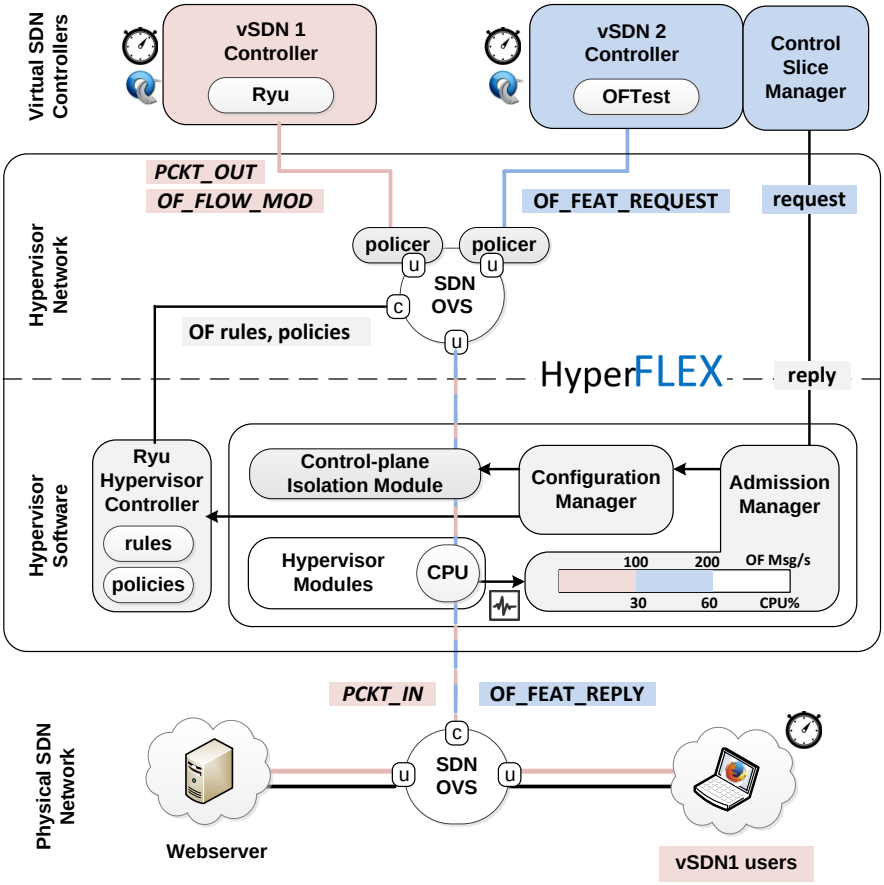


Figure 5.10: HyperFLEX testbed and evaluation setup for vSDN data plane performance and HyperFLEX admission control.

5.3.6.2 Admission Control and Data Plane Performance

Since the control plane performance in SDN networks has a direct impact on the data plane, in particular for new flow setup, we have evaluated the impact of the control plane isolation mechanisms implemented in HyperFLEX on the data plane of the virtual SDN networks. Additionally, we evaluated the response time of the proposed admission control in HyperFLEX that provides the tenant with the possibility to change the control plane isolation mechanism on run time For this purpose, we have extended the testbed setup as shown in Fig. 5.10 in order to measure the impact of the control plane isolation on the data plane as well as evaluate the response time of the admission control.

The testbed is extended with the implementation of the control plane admission and configuration managers as discussed in Sec. 5.3.5. In order to measure the data

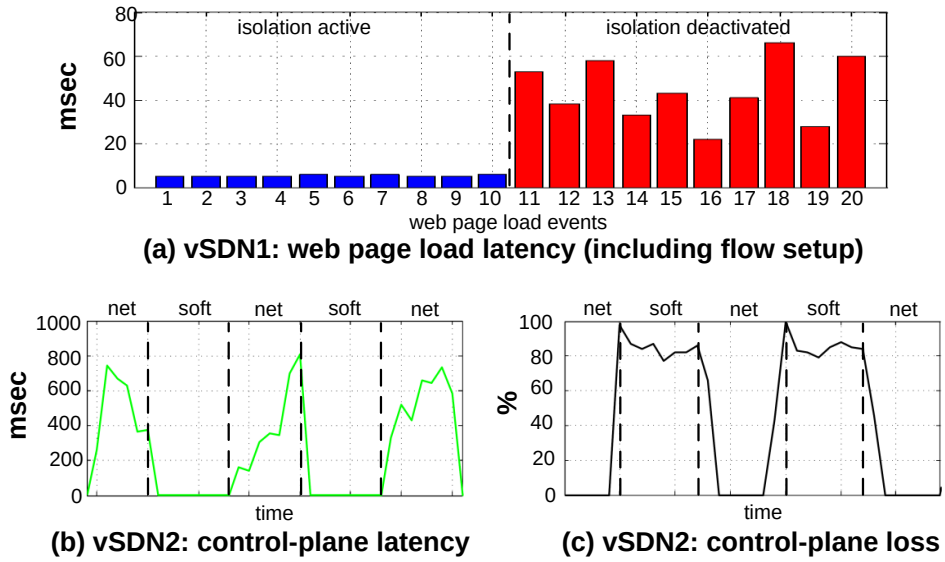


Figure 5.11: Evaluation of vSDN data plane performance and HyperFLEX admission control.

plane performance, in vSDN1, users start web browsing, i.e., initiating new connections. Each web page load event requires the setup of new flows from the controller of vSDN1, where Ryu controller is used as vSDN-C1. We measure the data plane latency, i.e., webpage load latency, which measures the impact of the control plane latency, i.e., flow setup latency, in vSDN1. At vSDN2, a control plane load is generated by OFTest at vSDN-C2 using OF_Feat_Request and its corresponding OF_Feat_Reply, in order to evaluate different control plane loads.

Fig. 5.11a shows the significance of control plane isolation. In case no control isolation mechanism is deployed and vSDN2 is experiencing a control burst, the web page load latency, including the flow setup time, of vSDN1 users goes from an average of *5 msec* up to *70 msec*. This means that the performance of vSDN1 is coupled with the control load of vSDN2 in case control plane isolation is deactivated. The control slice automation and adaption allows vSDN2 to alternate between software and network isolation options. The trade-off between both options is observed, where the control plane latency, shown in Fig. 5.11b, increases from an average of *2 msec* up to *700 msec* in case isolation moves to network. The average loss rate, shown in Fig. 5.11c, increases from *0%* to around *80%* in case isolation moves to software. Both Fig. 5.11b and Fig. 5.11c show the quick response of the control plane admission control to process the requests of vSDN2 and alternate between the control plane isolation options, including the setup of the software counters or the network policies.

5.4 Dynamic SDN Virtualization Layer

As introduced in the previous section, the virtualization of SDN networks is realized through a hypervisor layer that operates as an intermediate layer between SDN controllers and their particular virtual SDN network. An important feature of the SDN virtualization layer is its scalability. Hence, in favor of achieving a scalable hypervisor, there have been several implementation concepts that rely on a distributed architecture, e.g., FlowN [94]. Such a distributed concept is needed, e.g., if the SDN controllers of the slices are also distributed among a network, as it might be the case for wide area networks or for distributed cloud networks. Furthermore, a distributed hypervisor layer can adapt more efficiently to changing requirements, e.g., in case slices are added and removed or in case demands of slices are changing over time. Changing requirements may also lead to varying load on the distributed components. In order to adapt to changing requirements, a distributed hypervisor requires mechanisms that allow a dynamic and efficient adaptation of the hypervisor virtualization layer.

Distributed hypervisors provide the capability to spread hypervisor instances, which are hosting the functions realizing the virtualization, among the network. The instances connect the SDN controllers with their slices, i.e., they establish control connections between SDN controllers and their virtual SDN networks. As control connections are associated with load, efficient mechanisms to change the load distribution among the instances are necessary in order to provide an efficient virtualization. One possibility to adapt the load could be the migration of control connections between the instances. Besides load balancing, energy efficiency could be another operation target. As less hypervisor instances may provide a better energy efficiency, instances need to be turned on and off dynamically. Again, active control connections would need to be migrated. Although these examples are illustrating an important goal of hypervisor implementations, existing distributed SDN hypervisors do not provide any mechanisms that support the dynamic operation of the distributed instances and their control connections.

We propose a control path migration protocol for distributed SDN hypervisors, which are composed of multiple hypervisor instances. Such protocol is needed to allow distributed hypervisors to dynamically change virtual SDN networks. With our protocol, control connections between tenants' controllers and virtual SDN networks can be changed dynamically. We define a control connection as the connection from

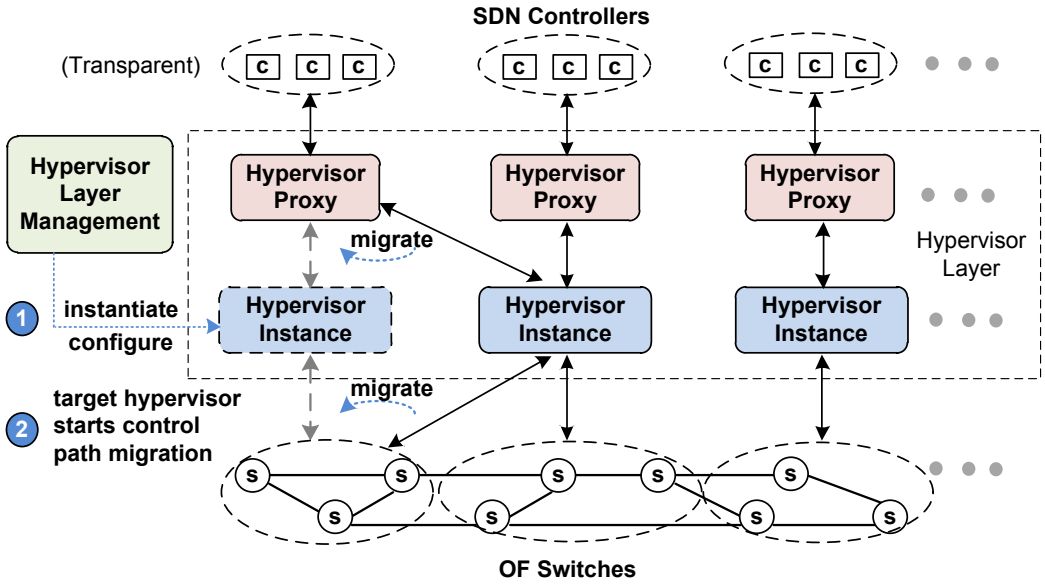


Figure 5.12: Dynamic distributed SDN hypervisor layer, which includes hypervisor instances, management and proxies.

the virtual SDN network, i.e., virtual SDN switch, to the SDN controller through a hypervisor instance. The protocol supports the migration of OpenFlow (OF) control connections. In detail, the control connections of one hypervisor instance can be migrated to another hypervisor instance with negligible control latency overhead at run time. In addition to the migration protocol, we also propose an architecture for a dynamic SDN virtualization layer. We implement our protocol for an SDN network, that uses OpenFlow to demonstrate a proof of concept. The measurement results are quantified in terms of control plane performance overhead.

5.4.1 Architecture for Dynamic SDN Hypervisors

We outline a virtualization architecture that provides a dynamic distributed SDN hypervisor layer. We consider a hypervisor layer that dynamically adapts to changes in the network state. The changes in the network state can be in terms of, e.g., over utilization for the resources of a hypervisor instance [79]. It can be also in terms of new requirements from the virtual SDN networks, e.g., requested control latency. The adaption is realized by instantiating or terminating hypervisor instances and using our proposed protocol to achieve the migration of the active control connections. The hypervisor layer is comprised of three main entities: 1) hypervisor instances 2) hypervisor proxies and 3) hypervisor layer management, as illustrated in Fig. 5.12.

The hypervisor instances are the controllers of the physical SDN network, which can be considered as the main workforce in the hypervisor layer. They implement the functions needed for SDN virtualization, e.g., topology abstraction, policy translation, control and data paths isolation [89]. The hypervisor functions are extended in our work to include a migration function that executes the proposed protocol.

We aim at providing a dynamic virtualization layer that adapts its structure without interrupting or even notifying the tenants' SDN controllers, i.e., changes in the hypervisor layer are transparent to the virtual SDN slices. For this purpose, we add hypervisor proxies on top of the hypervisor instances. The hypervisor proxies would act as an interface between the hypervisor layer and the SDN controllers. In case direct control connections are established between the hypervisor instances and the SDN controllers, a change in the control path would require the SDN controller to establish an OF control connection with the target hypervisor and to close the existing OF control connection towards the initial instance. This would mean that the dynamic scaling of the hypervisor layer would not be transparent anymore to the tenants' controllers. Moreover, it might lead in many cases to control path performance issues depending on the controller's supported OF version and implementation. For instance, if the controller does not support OF auxiliary connections, introduced in OF v1.3.0 [49], a control connection interruption would occur during the hypervisor migration. Thus, we place a proxy that maintains the control connection with the SDN controllers and has a connection to the running hypervisor instances. The proxy contains forwarding policies which specify how to forward the control messages from the SDN controllers to the designated hypervisor, and vice versa.

Finally, a management entity is needed to make the decision of adding and removing hypervisor instances based on the network state information that it collects from the network, i.e., performance of the switches and running hypervisor instances, or based on the requirements that it receives from the SDN controllers. The decision includes the location of a new hypervisor instance that has to be added or the identifier of the instance that should be terminated. Once a decision is made, the management starts the instantiation or termination of the hypervisor instances. The management is also required to install the configuration at the target hypervisor, which contains the addresses of its assigned proxies, controllers and switches. The target hypervisor uses this configuration to trigger and start the migration protocol, i.e., establishing the connections to the configured proxy and switches.

Fig. 5.12 illustrates an example where a decision has been made by the management entity to add a hypervisor instance to the SDN virtualization layer. The additional hypervisor instance is instantiated through the management entity. The instantiated hypervisor instance uses the proposed protocol to take over a set of OF switches, i.e., physical network domain, and a set of SDN controllers. In other words, the instantiated hypervisor triggers the migration of the control path, i.e., connections to designated proxy and switches.

5.4.2 Control Path Migration Protocol

In this section, we present our proposed protocol to enable a dynamic migration of the control path for virtual SDN networks, i.e., between SDN controllers, hypervisors and OF switches. The protocol is composed of three phases: 1) initialization, 2) controller path migration and 3) switch path migration. The procedure is illustrated in Fig. 5.13.

5.4.2.1 Initialization Phase

In non-virtual SDN networks, the control connection is defined by the OF specification to be initialized by the OF switch. The controller IP address is configured on each OF switch, where each switch attempts to initiate the connection to its configured controller. In the context of SDN virtualization, hypervisors act as controllers for the OF switches. In case of a dynamic on-demand instantiation of hypervisors, switches require to know the IP addresses of the target hypervisors. The first option is to install a pre-determined set of addresses for hypervisors at each switch. This acts as a limitation as it sets an upper bound to the maximum number of distributed hypervisors that can be instantiated. It can also induce a waste of switch resources in case of a high number of active hypervisor instances since the switches have to store this set of possible addresses of hypervisors. It also requires a trigger during run time to inform the switch about the active hypervisor that the switch should use.

Therefore, we propose a novel method to use the listening ports of OF switches. Almost all available protoype and commercial switches, e.g., Open vSwitch, CPqD, HP, Brocade and Pica8, implement a listening port that can be used to initialize the control connection from the hypervisor's side. That means that the initialization of the switch-controller TCP connection works in the counter opposite of the OF specification. The target hypervisor instance is configured by the centralized hypervisor

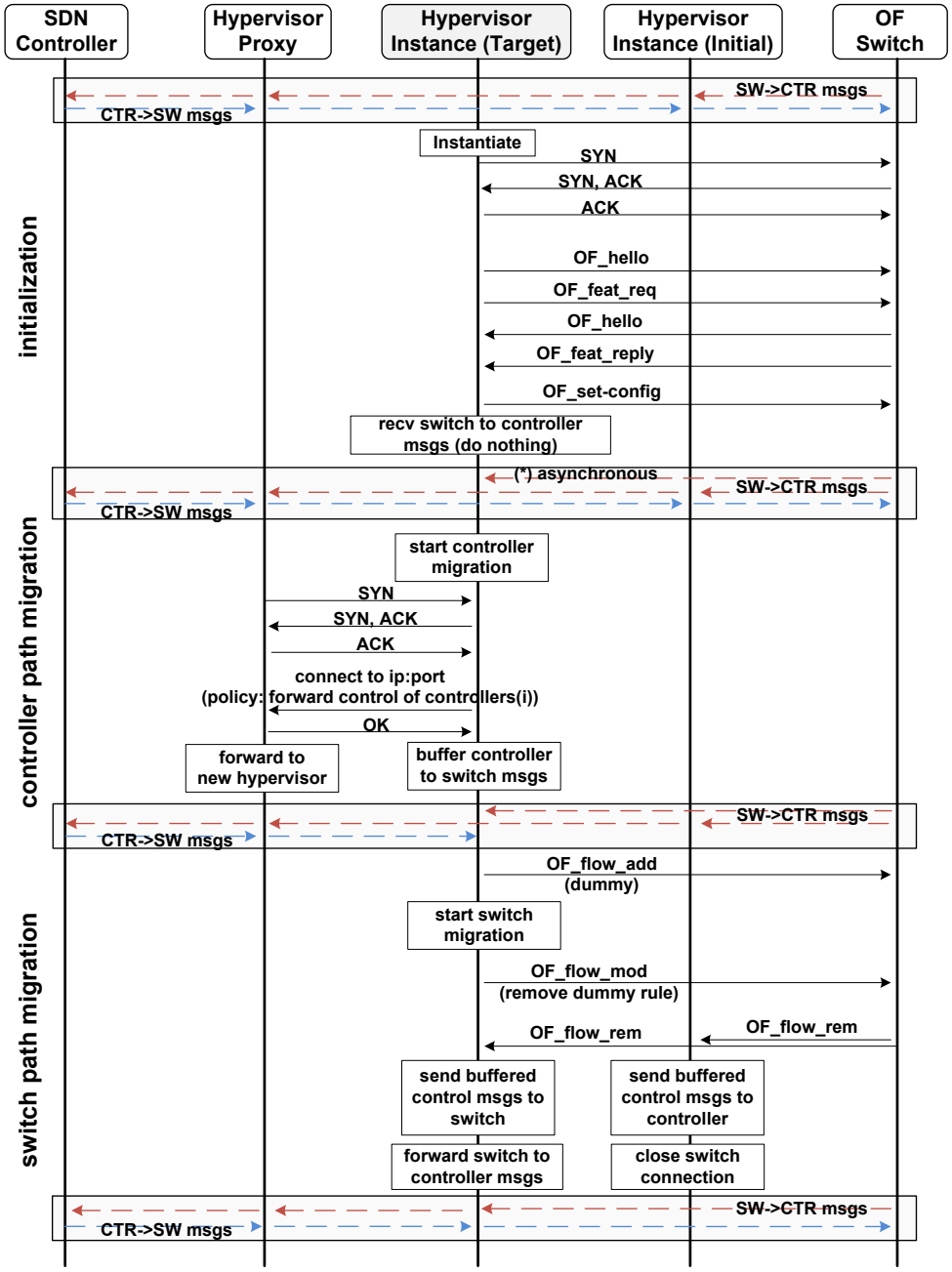


Figure 5.13: The message exchange and states of the control path during migration from an initial to a target hypervisor instance.

management. This configuration includes the switch IP addresses and their respective listening ports to which it needs to initiate the connection. After the connection is established, the OF_Hello and the switch configuration procedures follow the OF specification.

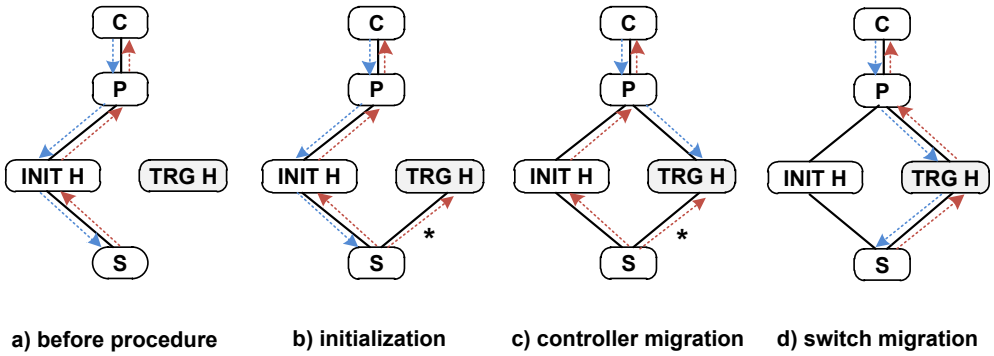


Figure 5.14: Phases of control path migration from an initial to a target hypervisor instance. Connections: solid black. Switch to controller messages: dashed red. Controller to switch messages: dashed blue. (*): asynchronous OF messages.

Once the target hypervisor is connected to the OF switch, it acts as a second controller to the switch as illustrated in Fig. 5.14b compared to Fig. 5.14a. This means that asynchronous OF messages, i.e., not in a request-reply form, e.g., OF_Pckt_in, would be sent to the target hypervisor as well. However, at that phase, there is no connection yet between the target hypervisor instance and the proxy, i.e., towards the controller. Hence, the target hypervisor can discard the received messages from the switch momentarily till the whole procedure is finished.

5.4.2.2 Controller Path Migration Phase

After the initialization phase, the target hypervisor instance initiates a TCP connection to the designated proxy from its configuration. The proxy addresses are preconfigured in the hypervisor instance by the management. The hypervisor informs the proxy of its assigned controllers. Consequently, the proxy installs policies to forward the control messages of those SDN controllers to the target hypervisor instance as shown in Fig. 5.14c. An example adaption of the forwarding policy at the proxy can be as follows: (remove: connection controller C1 $\Leftrightarrow$ connection hypervisor H1) and (forward: connection controller C1 $\Leftrightarrow$ connection hypervisor H2). Since the controller path migration is executed before the switch migration, the target hypervisor buffers the controller to switch messages till the switch migration is completed.

5.4.2.3 Switch Path Migration Phase

The final phase to complete the control path migration is the switch path migration. We have developed a protocol that uses OF messages to do the switch migration, that can be used starting from OF v1.0 [95]. The switch migration is initialized with the target hypervisor sending a dummy OF_flow_add message to the switch. This Flow_add message can be defined from OF flowspace fields which are not utilized by the controllers and are known by all hypervisor instances, e.g., OF cookie field, which is a controller issued ID and has 64 bits.

Afterwards, the target hypervisor starts the switch path migration by sending an OF_flow_mod to the switch that deletes the dummy flow from the switch. The reason for this is that deletion of a flow dictates an OF switch to send an OF_flow_rem to all its assigned controllers, and hence the initial as well as the target hypervisors would receive it. In this way, the initial hypervisor can be notified of the switch migration and withhold the responsibility for that switch. Meanwhile, the target hypervisor instance would know that it can take over the switch, thus meaning the control path migration completion. The target hypervisor then starts to send the buffered control message to the switch and continues normal operation as illustrated in Fig. 5.14d.

5.4.3 Results and Evaluation

We have conducted an initial evaluation of a prototype system that has been developed in a real SDN testbed. The evaluation setup can be seen in Fig. 5.15. The setup includes an established virtual SDN network, with an SDN controller and an OF switch. The control connection between the switch and the controller is assigned to the initial hypervisor instance. We evaluate the scenario where a new hypervisor instance, i.e., target hypervisor, is instantiated. The target hypervisor triggers our proposed protocol to take over the assignment of the SDN controller and switch by migrating the control connection, i.e., hypervisor to switch as well as hypervisor to proxy connections.

For the hypervisor layer, we have implemented a hypervisor instance in Python. The hypervisor implementation uses OpenflowJ Loxi library [96], which is an open source library. This library provides an OF protocol API that can be used for OF message parsing and generation. The library can support OF version 1.0 up to version

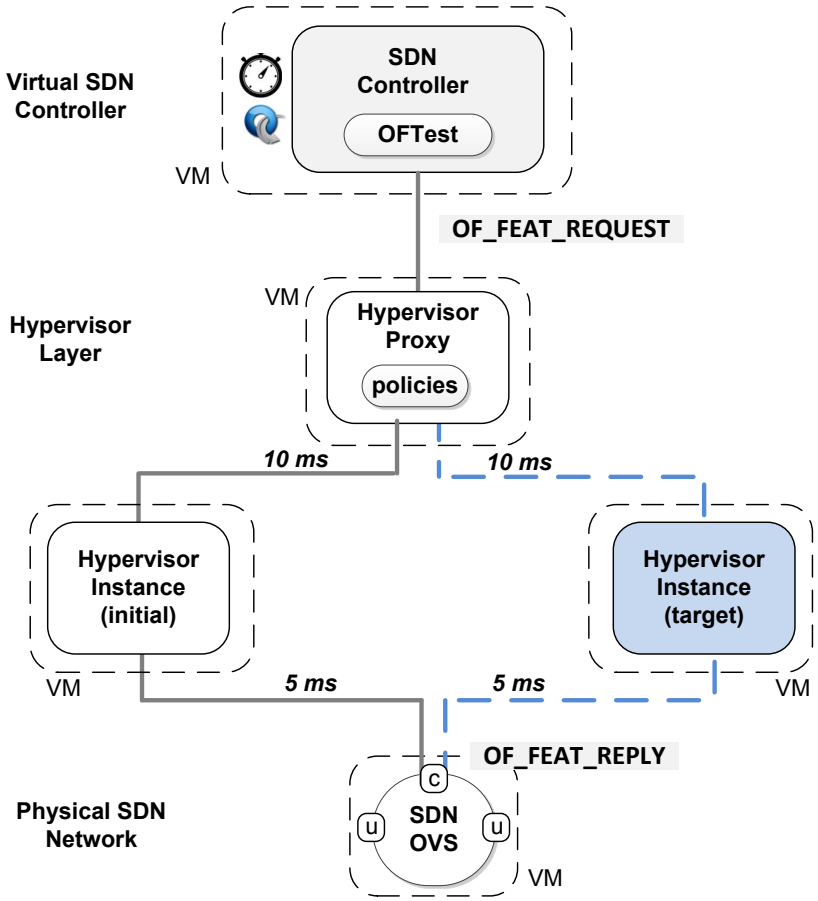


Figure 5.15: Evaluation setup for the control path migration protocol.

1.3. The hypervisor implementation also includes the proposed control path migration protocol. Additionally, a hypervisor proxy has been implemented in Python, that resides between the hypervisor and SDN controller. It contains the forwarding policies, i.e., mapping, for the connections to the controller and hypervisor instances.

The physical SDN network is realized as an Open vSwitch (OVS) [97], running OF version 1.2 and controlled by the implemented hypervisor. We use the listening port 6634 on the OVS in the initialization phase of the migration protocol. The SDN controller is realized by OFTest [92], a test suite framework for OF switches. The default OFTest can only be configured to generate a total number of OF message, without controlling the message sending rate. We have extended OFTest extensively to generate an average constant sending rate of OF messages to evaluate the impact of the assignment protocol on the control plane performance. Taking control latency as a performance metric, we use OFTest to generate OF_feat_request messages at an

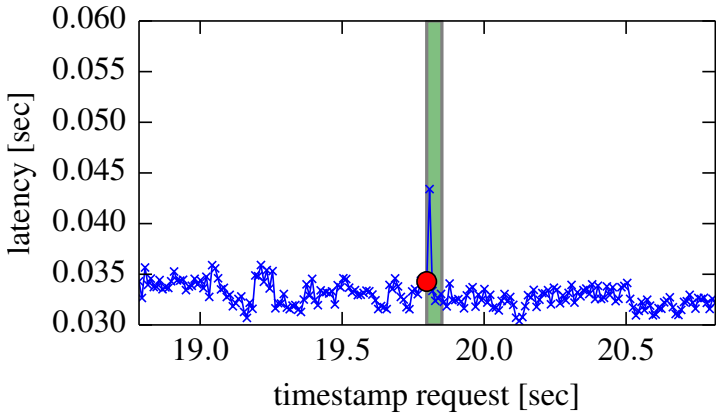


Figure 5.16: Time-series for OF message rate 100 msg/sec. Red dot shows the start of the migration procedure. The highlighted area shows the duration of the migration protocol and the affected OF messages.

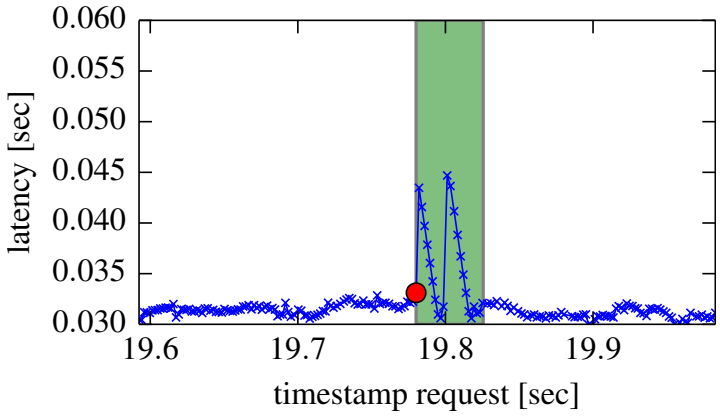


Figure 5.17: Time-series for OF message rate 500 msg/sec. Red dot shows the start of the migration procedure. The highlighted area shows the duration of the migration protocol and the affected OF messages.

average rate and we measure the latency to receive the corresponding OF_feat_reply back at OFTest, by using message xids which are unique transaction identifiers used to match requests to replies.

Throughout each run, we instantiate a new hypervisor instance and trigger the migration protocol. Hence, we can evaluate the impact of the migration on the control plane performance in terms of control latency. Each of the system components runs on a separate Virtual Machine (VM). In order to emulate a more realistic system, we have added an artificial latency of 10 millisecond on the link between the proxy and the hypervisor instances. In addition, a latency of 5 millisecond has been added on the link between the hypervisor instances and the switch, as shown in Fig. 5.15. For evaluation, we iterate over an OF sending message rate from 100 requests/sec

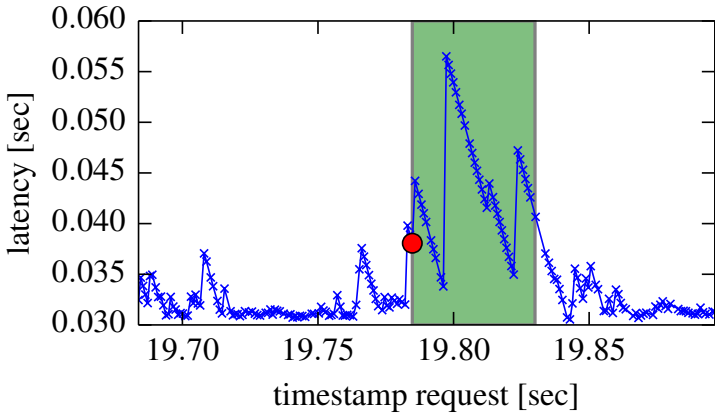


Figure 5.18: Time-series for OF message rate 1000 msg/sec. Red dot shows the start of the migration procedure. The highlighted area shows the duration of the migration protocol and the affected OF messages.

to 1000 requests/sec. For each rate, we repeat the evaluation run 40 times. Each run has a duration of 40 seconds.

Fig. 5.16, Fig. 5.17 and Fig. 5.18 show the time series for a sample run with an OF message sending rate of 100, 500 and 1000 requests per second, respectively. We identify the start of the migration protocol with the xid of the last OF_feat_request sent from the SDN controller to the initial hypervisor instance. The end of the protocol is marked by the xid of the first OF_feat_reply sent from the target hypervisor instance to the SDN controller. This indicates the duration needed to complete the migration protocol and the number of affected OF packets as well.

For one evaluation run, we observe an increase in the control latency during the migration execution compared to normal operation. During the migration execution, the maximum, i.e., peak, control latency goes to 43 milliseconds at 100 requests per second, refer to Fig. 5.16). At 1000 requests per second, as shown in Fig. 5.18, the maximum control latency goes to 57 milliseconds. Hence, the maximum increase in control latency can be noted to be proportional to the OF message rate on the control plane up to 1000 requests per second. The same observation can be made for the number of affected packets by the control path migration, where it increases in a proportion to the OF control rate. We consider the number of affected packets as an indicator to the operational duration in which the control channel is affected by the migration to another hypervisor. It is important to note that there is no packet loss (0%) due to migration for all evaluated OF rates. This implies that we could achieve a control path migration that is transparent to the SDN controllers in terms of control packet loss.

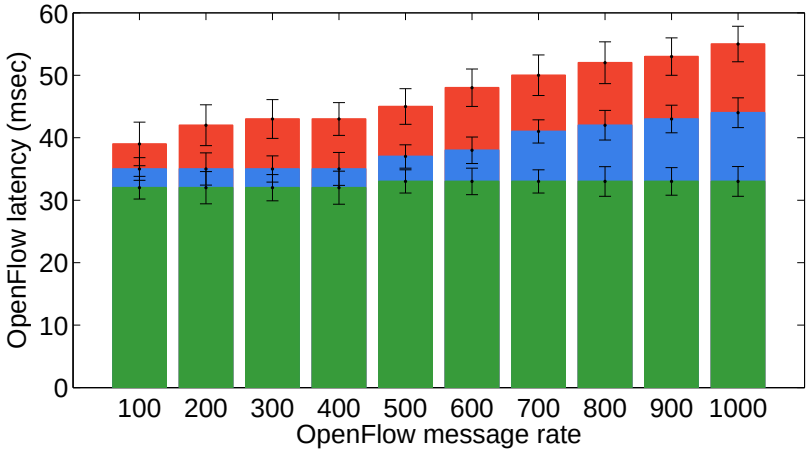


Figure 5.19: Latency of OF packets during migration process. Green (bottom) bars show the average latency, blue (intermediate) bars show the average latency of affected packets, red (upper) bars show the average of the maximum latency with standard deviation of all runs. For all bars, the standard deviation is added via black error bars.

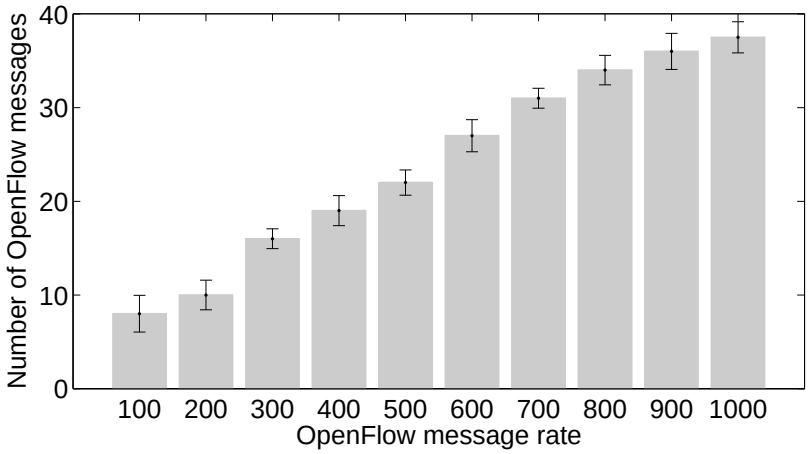


Figure 5.20: Number of affected OF packets during migration process. The error bars indicate the standard deviation of the average values for all runs.

In Fig. 5.19, we demonstrate statistical evaluation for the control latency over 40 runs for all marked packets affected by the migration. The green (bottom) bars show the average and standard deviation of the control latency in normal operation. This is the average of the 50 packets before the migration starts and of the 50 packets after the migration has finished. The average and standard deviation of the control latency during migration is represented via the blue (intermediate) bars. Finally, we calculate the average and standard deviation of the maximum latency reached during migration, which is illustrated via red (upper) bars.

The first observation is that the average control latency in normal operation, i.e., green bars, is not affected by the control rate. The average control latency during migration, i.e., blue bars, confirms the correlation between the increase in the control latency during migration and the OF control rate. The same observation can be shown for the average of maximum control latency, i.e., red bars, during control path migration. This can be explained as follows. During the migration, the protocol starts with the controller path migration, afterwards it initializes the switch path migration. All SDN controller to switch messages are buffered at the target hypervisor till the switch migration is completed, then it sends all buffered control messages to the switch. Thus, the number of affected, i.e., buffered, packets during migration increases with increasing the control rate as shown in Fig. 5.20. However, note that the number of affected packets during migration is negligible compared to the total number of packets during operation. For example, at 1000 requests per second and for a run duration of 40 seconds, the average number of affected packets is 45 packets out of a total of 40000 packets, which is only 0.1%.

5.5 Discussion and Conclusion

In this chapter, we introduce an SDN hypervisor layer, what we call HyperFLEX, that provides flexible, reliable and dynamic SDN virtualization. HyperFLEX decomposes the hypervisor for SDN networks into functions, which are flexibly placed among servers or SDN network elements. As the hypervisor functions can be flexibly placed among the network, the architecture provides different operation modes that can be used according to the performance guarantees needed for the virtualized SDN network. HyperFLEX also provides control plane isolation, which has been neglected by existing SDN hypervisors, that ensures control plane isolation among the virtual SDN tenants while protecting the hypervisor from over-utilization. The realization of the control plane isolation function is thoroughly discussed as a software or a network function. Furthermore, HyperFLEX provides a scalable and dynamic distributed SDN hypervisor layer. In order to support dynamic virtual SDN network changes, we propose an OpenFlow-based control path migration protocol for distributed SDN hypervisors, that can adapt the virtualization layer without interrupting the virtual SDN networks, i.e., transparent.

A first prototype of HyperFLEX is implemented and evaluated in order to show the trade-offs between the different operation modes of the proposed hypervisor architecture. Measurement results taken from a real SDN testbed for different utilization scenarios reveal that the flexible hypervisor provides improved control plane isolation which reflects on the performance of virtual SDN networks of multiple tenants. Furthermore, the architecture proves to be more reliable as the performance protection of hypervisor functions is improved. The impact of our proposed control path migration protocol on the performance of the virtual SDN control plane is also evaluated through measurements of the control latency with varying OF message sending rates. Our evaluation shows that the control latency and the number of packets affected during migration are proportional to the OF rate. We observe no control packet loss for all OF rates. Additionally, we observe that the number of packets affected by the migration is negligible compared to the total umber of packets in normal operation. Hence, we conclude that the dynamic migration of the control path in the virtualization layer has no significant impact on the virtual SDN slices, i.e., transparent to their operation.

Chapter 6

Conclusion and Outlook

In this chapter, we draw the conclusions of this thesis and we discuss our observations and results. We also outline several directions for future work.

6.1 Conclusion and Discussion

In summary, this thesis studies the application of the SDN and NFV concepts on the design of the mobile core network in order to achieve cost reduction and to cope with future dynamics. An analysis is presented to the realization of the mobile core network functions based on SDN and NFV, with a focus on two main functions, namely the Serving Gateway (SGW) and Packet data network Gateway (PGW). The analysis shows that for NFV, the gateway functions can be directly adapted to NFV through running their code on commodity hardware. However, for SDN, the defacto OpenFlow protocol in its latest version requires various extensions to support the mobile core network operation, e.g., GTP tunneling and charging.

According to the analysis of the impact of SDN and NFV on the data as well as control planes of the mobile core network, we observe trade-offs between both realizations in terms of performance, i.e., latency, and cost, i.e., network and data center resources cost. For this purpose, several optimization models are developed in order to find the optimal placement of the SDN and NFV mobile core network functions as well as the optimal location of data centers, what we call the Function Placement Problem (FPP). The problem is evaluated with static as well as time-varying traffic patters. The evaluations show that a design that combines both SDN and NFV, i.e., a core network that is partly operated through SDN while the other part operates through NFV, results in an optimal cost while satisfying the latency requirements.

A design solely based on either SDN or NFV brings more cost than a combined design and does not satisfy the data and control planes latency requirements. The evaluation also demonstrates the trade-offs between the network cost and the data center energy consumption cost, where a balance between both cost factors can be achieved. Furthermore, a pareto optimal multi-objective problem for the network cost and data center cost is introduced, which provides operators with the possibility to find the right balance between both cost factors. An algorithm is proposed for the multi-objective problem that infers a set of input locations for the data centers which exhibits savings in the problem's solving run time.

Finally, multiple solutions are developed to improve the SDN virtualzation layer that provides virtual logical SDN mobile core network slices. The developed solutions focus on providing flexibility for the SDN virtualization as well as control plane isolation between the tenants. The evaluation through prototype measurements shows that the control plane isolation is an important feature for an SDN virtualization layer. Without an isolation function, the control planes of different virtual network tenants, that share the same physical infrastructure, can interfere resulting in a degradation in their data plane performance. Furthermore, a protocol is proposed for the migration of the SDN virtualization control plane in order to support the dynamics of virtual network slices. The evaluation of the implemented prototype shows that a seamless migration of the control path, i.e., transparent to the tenants, can be achieved. The performance overhead of the proposed protocol is shown to be negligible, in terms of the control plane latency and control message loss.

6.2 Directions for Future Work

There are different directions that can be followed for future work. Regarding the application of SDN and NFV to the mobile core network design, further core network functions can be integrated in the network design in order to evaluate their impact. For instance, the Mobility Management Entity (MME) can be modeled and integrated in the proposed optimization problems to study the impact of users' mobility on the core network design. Further cost factors and constraints can also be considered for the optimal placement and dimensioning of the SDN and NFV mobile core network. For instance, the cost of the SDN+ switches or the inter-data center links can be modeled and included in the proposed problems. The traffic models for the placement and dimensioning problems can be extended to include massive num-

ber of users, e.g., Machine-to-machine (M2M) use-case, or to include highly mobile users, e.g., automotive use-case. The set of data centers locations could be extended to arbitrary locations on the core network topology, i.e., not the same locations as the graph nodes. Additionally, constraints on the network as well as the data center resources can be further investigated as a model for restricted resources of some mobile operators.

The investigations of the SDN virtualization layer can also be extended in several directions. The control plane isolation mechanisms provided by the SDN virtualization layer can be extended to other resources, e.g., OpenFlow switch flow table or data plane bandwidth. In a flexible distributed SDN virtualization layer, more investigations can be conducted to support more dynamic changes for the virtual network tenants, e.g., dynamic virtual topology updates. Additionally, efficient models and algorithms are needed to find the optimal distribution for the flexible SDN hypervisor functions across the network according to the virtual network demands, e.g., tenants' SDN controller locations or virtual topology distribution. The resilience of the virtualization layer is also an important open aspect, where solutions and concepts are needed to cope with failures in the virtualization layer and to provide a more reliable operation for the virtual network tenants.

List of Figures

List of Tables

Bibliography

[1] Nokia Networks, "Nokia Networks Technology Vision 2020," 2015, https://networks.nokia.com/innovation/technology-vision.

[2] Nokia Networks, "5G use cases and requirements ," 2015, https://resources.alcatel-lucent.com/asset/200010.

[3] European Telecommunications Standards Institute (ETSI), "ETSI GS NFV-MAN 001 V1.1.1," 2014, https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf.

[4] Open Networking Foundation (ONF), "SDN Architecture," 2014, https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf.

[5] N. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications magazine*, vol. 47, no. 7, pp. 20–26, 2009.

[6] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, 2016.

[7] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, "A virtual SDN-enabled LTE EPC architecture: a case study for S-/P-gateways functions," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

[8] Y. Li and M. Chen, "Software-defined network function virtualization: a survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[9] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[10] S. Schmid and J. Suomela, "Exploiting locality in distributed SDN control," in *Proceedings of ACM Hot topics in software defined networking (HotSDN), ACM SIGCOMM*, 2013, pp. 121–126.

[11] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[12] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN programming with pyretic," *Technical Reprot of USENIX*, 2013.

[13] Open Networking Foundation (ONF), "OpenFlow switch specifications 1.5.1," Mar. 2015, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[14] European Telecommunications Standards Institute (ETSI), "Network Function Virtualization, update white paper," October 2013.

[15] "VMware virtualization overview," 2015, https://www.vmware.com/pdf/virtualization.pdf.

[16] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[17] "Intel Data Plane Development Kit (DPDK)," 2015, http://dpdk.org/.

[18] A. Khan, A. Zugenmaier, D. Jurca, and W. Kellerer, "Network virtualization: a hypervisor for the internet?" *IEEE Communications Magazine*, vol. 50, no. 1, pp. 136–143, 2012.

[19] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "LTE mobile network virtualization," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, 2011.

[20] A. Khan, D. Jurca, K. Kozu, W. Kellerer, and M. Yabusaki, "The reconfigurable mobile network," in *2011 IEEE International Conference on Communications Workshops (ICC)*. , 2011, pp. 1–5.

[21] 3GPP, "3GPP TS 23.401 LTE; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access) ," 2011, http://www.etsi.org/deliver/etsi_ts/124399_123401/ts_123401v130500p.pdf.

[22] Nokia Networks, "Enabling mobile broadband growth – Evolved Packet Core, white paper ," 2009, http://www.mforum.ru/arc/20110319_NSN_EPC_wp_MForum.pdf.

[23] 3GPP, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access, 3GPP TS 23.401, release 10," 2012, http://www.etsi.org/deliver/etsi_ts/123400_123499/123401/11.03.00_60/ts_123401v110300p.pdf.

[24] R. Kreher and K. Gaenger, *LTE signaling: troubleshooting and optimization*. John Wiley & Sons, 2010.

[25] 3GPP, "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS), 3GPP TS 24.301," 2009, http://www.etsi.org/deliver/etsi_ts/124300_124399/124301/10.03.00_60/ts_124301v100300p.pdf.

[26] P. Lescuyer and T. Lucidarme, "Evolved packet system (eps): The LTE and SAE information of 3G UMTS" 2008.

[27] Nokia Networks, "LTE-capable transport: a quality user experience demands an end-to-end approach," 2011, http://www.hit.bme.hu/~jakab/edu/litr/Mobile_backhaul/lte_transport_requirements_NOKIA.pdf.

[28] 3GPP, "Charging management; Charging architecture and principles, 3GPP TS 32.240, release 11," 2013, http://www.etsi.org/deliver/etsi_ts/132200_132299/132240/11.05.00_60/ts_132240v110500p.pdf.

[29] "Advanced Telecom Computing Architecture (ATCA)," 2015, http://www.picmg.org/openstandards/advancedtca/.

[30] J. C. Mogul, J. Mudigonda, J. R. Santos, and Y. Turner, "The NIC is the hypervisor: bare-metal guests in IaaS clouds," in *Hot Topics in Operating Systems*, 2013.

[31] 3GPP, "Policy and charging control architecture, 3GPP TS 23.203, release 9," 2010, http://www.etsi.org/deliver/etsi_ts/123200_123299/123203/12.06.00_60/ts_123203v120600p.pdf.

[32] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th workshop on All things cellular, ACM SIGCOMM*, 2014, pp. 33–38.

[33] A. Basta, A. Blenk, M. Hoffmann, H. J. Morper, K. Hoffmann, and W. Kellerer, "SDN and NFV dynamic operation of LTE EPC gateways for time-varying traffic patterns," in *International Conference on Mobile Networks and Management*, 2014, pp. 63–76.

[34] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: taking control of cellular core networks," *Proceedings of ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2013, pp. 163–174.

[35] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: toward software-defined mobile networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 44–53, 2013.

[36] M. R. Sama, L. M. Contreras, J. Kaippallimalil, I. Akiyoshi, H. Qian, and H. Ni, "Software-defined control of the virtualized mobile packet core," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 107–115, 2015.

[37] M. Moradi, W. Wu, L. E. Li, and Z. M. Mao, "Softmow: eecursive and reconfigurable cellular wan architecture," in *Proceedings of ACM Conference on emerging Networking Experiments and Technologies (CoNEXT)*, 2014, pp. 377–390.

[38] G. Hampel, M. Steiner, and T. Bu, "Applying software-defined networking to the telecom domain," *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2013, pp. 133–138.

[39] J. Kempf, B. Johansson, S. Pettersson, H. Lüning, and T. Nilsson, "Moving the mobile evolved packet core to the cloud," in *IEEE 8th Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2012, pp. 784–791.

[40] P. Gurusanthosh, A. Rostami, and R. Manivasakan, "SDMA: a semi-distributed mobility anchoring in LTE networks," in *Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, 2013, pp. 133–139.

[41] T. Mahmoodi and S. Seetharaman, "Traffic jam: handling the increasing volume of mobile data traffic," *IEEE Vehicular Technology Magazine*, vol. 9, no. 3, pp. 56–62, 2014.

[42] H. Lindholm, L. Osmani, H. Flinck, S. Tarkoma, and A. Rao, "State space analysis to refactor the mobile core," in *Proceedings of the 5th Workshop on All Things Cellular, ACM SIGCOMM*, 2015, pp. 31–36.

[43] V.-G. Nguyen, T.-X. Do, and Y. Kim, "SDN and virtualization-based LTE mobile network architectures: A comprehensive survey," *Wireless Personal Communications*, vol. 86, no. 3, pp. 1401–1438, 2016.

[44] H. Baba, M. Matsumoto, and K. Noritake, "Lightweight virtualized evolved packet core architecture for future mobile communication," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2015, pp. 1811–1816.

[45] W. Kiess, X. An, and S. Beker, "Software-as-a-service for the virtualization of mobile network gateways," in *IEEE Conference on Global Communications (GLOBECOM)*, 2015, pp. 1–6.

[46] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "Mobiscud: a fast moving personal cloud in the mobile network," in *Proceedings of the 5th Workshop on All Things Cellular, ACM SIGCOMM*, 2015, pp. 19–24.

[47] A. Banerjee, X. Chen, J. Erman, V. Gopalakrishnan, S. Lee, and J. Van Der Merwe, "MOCA: a lightweight mobile cloud offloading architecture," in *Proceedings of ACM workshop on Mobility in the evolving internet architecture*, 2013, pp. 11–16.

[48] J. Cho, B. Nguyen, A. Banerjee, R. Ricci, J. Van der Merwe, and K. Webb, "SMORE: Software-defined networking mobile offloading architecture," in *Proceedings of the 4th workshop on All things cellular, ACM SIGCOMM*, 2014, pp. 21–26.

[49] Open Networking Foundation (ONF), "OpenFlow Switch Specifications 1.3.0," Oct. 2012, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf.

[50] "Floodlight OpenFlow Conroller," http://www.projectfloodlight.org/floodlight.

[51] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," *Proceedings of ACM Hot topics in software defined networking (HotSDN*, 2012, pp. 7–12.

[52] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, 2013, pp. 1–9.

[53] "Gurobi Optimization Solver," http://www.gurobi.com/products/gurobi-optimizer/.

[54] "LTE Coverage Map," http://www.mosaik.com/marketing/cellmaps/;http://platform.cellmaps.com/.

[55] Nokia Networks, "Enabling Mobile Broadband Growth, white paper," 2013, https://nsn.com/system/files/document/epc_white_paper.pdf.

[56] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM Queue - Large-Scale Implementations Magazine*, vol. 11, no. 12, pp. 20–40, 2013.

[57] J. Erman and K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl," *Proceedings of the 2013 conference on Internet measurement conference (IMC)*, 2013, pp. 353–360.

[58] V. Gehlen, A. Finamore, M. Mellia, and M. M. Munaf, "Uncovering the big players of the web," in *Proceedings of the 4th international conference on Traffic Monitoring and Analysis (TMA)*, 2012, pp. 15–28.

[59] L. Qian, B. Wu, R. Zhang, W. Zhang, and M. Luo, "Characterization of 3G data-plane traffic and application towards centralized control and management for software defined networking," *IEEE International Congress on Big Data*, 2013, pp. 278–285.

[60] C. Rossi, N. Vallina-Rodriguez, V. Erramilli, Y. Grunenberger, L. Gyarmati, N. Laoutaris, R. Stanojevic, K. Papagiannaki, and P. Rodriguez, "3GOL: power-boosting ADSL using 3G onloading," in *Proceedings of ACM conference on Emerging networking experiments and technologies (CoNEXT)*, 2013, pp. 187–198.

[61] Y. Zhang and A. Arvidsson, "Understanding the characteristics of cellular data traffic," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 42, no. 4, p. 461, 2012.

[62] UMTS Report, "UMTS Forum Report 44 Mobile traffic forecasts 2010-2020," 2011, http://www.umts-forum.org/component/option,com_docman/task,doc_download/gid,2537/Itemid,213/.

[63] US Population Distribution, "Annual estimates of the resident population: 2010 to 2013, united states – metropolitan and micropolitan statistical area," 2014, available at http://factfinder2.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=bkmk.

[64] GeoLocations Library, "MaxMind GeoLocations," 2014, http://dev.maxmind.com/geoip/legacy/geolite/.

[65] A. Basta, A. Blenk, K. Hoffmann, H. J. Morper, M. Hoffmann, and W. Kellerer, "Towards a cost optimal design for a 5G mobile core network based on SDN and NFV," in *IEEE Transactions on Network and Service Management (TNSM)*, to be published, 2017.

[66] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 12, no. 1, pp. 4–17, 2015.

[67] A. Sallahi and M. St-Hilaire, "Optimal model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 19, no. 1, pp. 30–33, 2015.

[68] S. Gebert, D. Hock, T. Zinner, P. Tran-Gia, M. Hoffmann, M. Jarschel, E.-D. Schmidt, R.-P. Braun, C. Banse, and A. Köpsel, "Demonstrating the optimal placement of virtualized cellular network functions in case of large crowd events," in *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 44, no. 4, 2014, pp. 359–360.

[69] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the NFV provisioning puzzle: efficient placement and chaining of virtual net-

work functions," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 98–106.

[70] S. Sahhaf, W. Tavernier, M. Rost, S. Schmid, D. Colle, M. Pickavet, and P. Demeester, "Network service chaining with optimized network function embedding supporting service decompositions," *ELSEVIER Computer Networks*, vol. 93, pp. 492–505, 2015.

[71] R. Shi, J. Zhang, W. Chu, Q. Bao, X. Jin, C. Gong, Q. Zhu, C. Yu, and S. Rosenberg, "MDP and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization," in *IEEE International Conference on Services Computing (SCC)*, 2015, pp. 65–73.

[72] A. Baumgartner, V. S. Reddy, and T. Bauschert, "Mobile core network virtualization: a model for combined virtual core network function placement and topology optimization," in *IEEE Conference on Network Softwarization (NetSoft)*, 2015, pp. 1–9.

[73] O. Grodzevich and O. Romanko, "Normalization and other topics in multi-objective optimization," in *Proceedings of the Fields–MITACS Industrial Problems Workshop*, 2006, pp. 89–101.

[74] OpenSignal, "LTE coverage map," http://opensignal.com/.

[75] Internet eXchange Point (IxP) Locations, "Data center IxPs," http://www.datacentermap.com/ixps.html.

[76] 3GPP, "3GPP TS 25.913 Requirements for Evolved UTRA (E-UTRA) and Evolved UTRAN (E-UTRAN) ," 2010, http://www.etsi.org/deliver/etsi_tr/125900_125999/tr_125913v090000p.pdf.

[77] 3GPP, "3GPP TS 36.913 Requirements for further advancements for Evolved Universal ," 2010, http://www.etsi.org/deliver/etsi_tr/136900/tr_136913v090000p.pdf.

[78] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review (CCR)*, vol. 39, no. 1, pp. 68–73, 2008.

[79] A. Blenk, A. Basta, and W. Kellerer, "Hyperflex: an SDN virtualization architecture with flexible hypervisor function allocation," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 397–405.

[80] A. Basta, A. Blenk, Y.-T. Lai, and W. Kellerer, "Hyperflex: demonstrating control-plane isolation for virtual software-defined networks," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1163–1164.

[81] A. Basta, A. Blenk, H. B. Hassine, and W. Kellerer, "Towards a dynamic SDN virtualization layer: Control path migration protocol," in *International Conference on Network and Service Management (CNSM)*, 2015, pp. 354–359.

[82] R. Sherwood, J. Naous, S. Seetharaman, D. Underhill, T. Yabe, K.-K. Yap, Y. Yiakoumis, H. Zeng, G. Appenzeller, R. Johari, N. McKeown, M. Chan, G. Parulkar, A. Covington, G. Gibb, M. Flajslik, N. Handigol, T.-Y. Huang, P. Kazemian, and M. Kobayashi, "Carving research slices out of your production networks with OpenFlow," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 40, no. 1, p. 129, 2010.

[83] A. Blenk, A. Basta, J. Zerwas, and W. Kellerer, "Pairing SDN with network virtualization: the network hypervisor placement problem," in *IEEE Conference on Network Function Virtualization and Software Defined Network (IEE NFV-SDN)*, 2015, pp. 198–204.

[84] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control plane latency with SDN network hypervisors: the cost of virtualization," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 13, no. 3, pp. 366–380, 2016.

[85] E. Salvadori, R. D. Corin, a. Broglio, and M. Gerola, "Generalizing virtual network topologies in OpenFlow-based networks," *IEEE Conference on Global Communications (GLOBECOM*, 2011, pp. 1–6.

[86] R. D. Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, "VeRTIGO: network virtualization and beyond," *European Workshop on Software Defined Networking (EWSDN)*, 2012, pp. 24–29.

[87] Z. Bozakov and P. Papadimitriou, "AutoSlice: automated and scalable slicing for software-defined networks," in *Proceedings of the ACM conference on CoNEXT student workshop*, 2012, p. 3.

[88] P. Skoldstrom and W. John, "Implementation and evaluation of a carrier-grade OpenFlow virtualization scheme," in *European Workshop on Software Defined Networks (EWSDN)*, 2013, pp. 75–80.

[89] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *ELSEVIER Computer Networks*, vol. 54, pp. 862–876, 2008.

[90] A. Blenk and W. Kellerer, "Traffic pattern based virtual network embedding," in *Proceedings of the ACM conference on CoNEXT student workshop*, 2013, pp. 23–26.

[91] A. Basta, B. Burcu, M. Hoffmann, G. Carle, and D. Schupke, "Failure coverage in optimal virtual networks," in *Optical Fiber Communication Conference (OFC)*, 2013, pp. 3–6.

[92] "OFTest." https://github.com/floodlight/oftest

[93] "Ryu SDN Controller." http://osrg.github.io/ryu/

[94] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined Networks," *IEEE Internet Computing*, vol. 17, pp. 20–27, 2013.

[95] Open Networking Foundation (ONF), "OpenFlow Switch Specifications 1.0.0," Oct. 2009, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf.

[96] "OpenFlowJ Loxi library." https://github.com/floodlight/loxigen/wiki/OpenFlowJ-Loxi

[97] "Open vSwitch (OvS)." http://openvswitch.org/