Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. P. Tran-Gia

Quality of Experience Assessment of Cloud Applications and Performance Evaluation of VNF-Based QoE Monitoring

Lam Dinh-Xuan

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Bericht 1/18

Würzburger Beiträge zur
Leistungsbewertung Verteilter Systeme

Herausgeber

Prof. Dr.-Ing. P. Tran-Gia
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg

Tel.: +49-931-31-86630
Fax.: +49-931-31-86632
email: trangia@informatik.uni-wuerzburg.de

Satz

Reproduktionsfähige Vorlage des Autors.
Gesetzt in LaTeX Linux Libertine 10pt.

ISSN 1432-8801

Quality of Experience Assessment of Cloud Applications and Performance Evaluation of VNF-Based QoE Monitoring

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Lam Dinh-Xuan

aus

Thai Nguyen, Vietnam

Würzburg 2018

Eingereicht am: 09.07.2018
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr.-Ing. Phuoc Tran-Gia
2. Gutachter: Prof. Dr. Tobias Hoßfeld
Tag der mündlichen Prüfung: 10.10.2018

Acknowledgments

This study is funded within the project 911 of the Vietnamese government in cooperation with German Academic Exchange Service (DAAD), the scholarship is administrated by the Ministry of Education and Training, Vietnam International Education Cooperation Department. I would like to gratefully acknowledge all of those who give me enormous support to pursue this study.

This dissertation has been accomplished with not only the great help of people but also the professional working environment at the Chair of Communication Networks and the University of Würzburg.

First of all, I would like to express the deepest sense of gratitude to my supervisor Prof. Phuoc Tran-Gia, who offered me an amazing chance to study at the Chair of Communication Networks, the University of Würzburg. Thanks to the approval of Prof. Phuoc Tran-Gia, I have opportunities to learn new knowledge and technologies, to work with colleagues in a perfect environment, to join in the interesting INPUT project, and to share with people the unforgettable moments in Germany. Prof. Phuoc Tran-Gia not only gives me personal enthusiastic encouragement, valuable guidance in research but also supports me to participate in numerous conferences, workshops, and project meetings.

I wish to acknowledge the help provided by Prof. Tobias Hoßfeld, who is the second reviewer of my dissertation. Advice and critical comments given by Prof. Tobias Hoßfeld have been a great contribution to the enrichment of this work. Furthermore, my special thanks are extended to the member of the board of examiners, Prof. Samuel Kounev.

I am particularly grateful for the assistance given by Dr. Florian Wamser, who is the leader of my research group at the Chair. Thanks to his leading and guid-

ance on QoE research, cloud computing, and future Internet technologies, I have written together with him numerous research papers and project reports. He also gives me an extraordinary support in this thesis process, corrects a large part of my dissertation, and keeps my progress on schedule.

I would like to offer my special thanks to Dr. Florian Metzger, Dr. Michael Seufert, Dr. Valentin Burger, and Frank Loh. Their valuable and constructive suggestions have been a great contribution to this work. I would also like to express my great appreciation to Dr. Matthias Hirth and Dr. Christian Schwartz for their very significant supports, together with them I have written the first research paper that is also a part of this thesis. Furthermore, I would like to give special thanks to Prof. Thomas Zinner and Prof. Harald Wehnes for their important guidance from the beginning of my research progress.

I wish to extend my thanks to all former and current colleagues Christopher Metter, Anika Schwind, Kathrin Borchert, Stefan Geißler, Nicholas Gray, Alexej Grigorjew, Stanislav Lange, Christian Moldovan, Dr. Steffen Gebert, Susanna Schwarzmann, and especially Anh Nguyen-Ngoc who always encourages and shares with me the unforgettable times at the Chair and in Germany. I would also like to thank all my students and co-authors of joint papers, Christian Popp, Prof. Huong Truong-Thu, Constantinos Vassilakis, and Anastasios Zafeiropoulos. I would also like to thank Mrs. Alison Wichmann and Mrs. Susann Schmitt for their organizational support and administrative assistants.

Finally, I wish to say many thanks to my parents Dinh and Vu for their enthusiastic encouragement and supports. Especially, I would like to express the warmest thanks to my wife Van Nguyen-Thi and my little daughter Chi Dinh-Lan for their heartfelt love and endless inspiration.

Contents

1 Introduction **1**

1.1 Scientific Contributions 5

1.2 Outline of the Thesis 7

2 QoE Assessment and Placement for Cloud Applications **11**

2.1 Background and Related Work 15

2.1.1 Software as a Service Architecture 15

2.1.2 Cloud-based Photo Service in the Context of Egde Networks 17

2.1.3 Relationship Between Network QoS and Quality of Experience 18

2.1.4 Cloud-based Collaborative Word Processor 19

2.2 Impact of Delay and Packet Loss on Google Docs 20

2.2.1 Methodology and Testbed Setup 21

2.2.2 Impact of Different Network Conditions on Subprocesses in Single User Measurements 26

2.2.3 Impact of Different Network Conditions on Subprocesses in Collaborative Task 30

2.2.4 Impact of Delay and Packet Loss on Total Process in Collaborative Task 32

2.3 QoE Aware Placement of Cloud-based Photo Service in Edge Networks 34

2.3.1 QoS Model and File Downloading Measurements 36

2.3.2 QoE Model and the Placement of Content 42

2.4 Lesson Learned 47

3 VNF-based QoE Monitoring in the Cloud 51
3.1 Background and Related Work 54
3.1.1 HTTP Adaptive Video Streaming 54
3.1.2 QoE Assessment Methodologies 55
3.1.3 QoE Monitoring Methodologies 57
3.1.4 QoE Monitoring for HTTP Adaptive Video Streaming . 60
3.1.5 NFV Cloud Infrastructure for VNF-based QoE Monitoring 61
3.2 Impact of Network QoS on the Accuracy of QoE Estimation for HAS 62
3.2.1 Methodology and Measurement Setup 63
3.2.2 Impact of Bandwidth on the Accuracy of Video Buffer and QoE Estimation 70
3.2.3 Impact of Packet Re-Ordering on the Accuracy of QoE Monitoring for HAS 75
3.3 Study on the Accuracy of VNF-based QoE Monitoring in the Cloud 77
3.3.1 Architecture for VNF QoE Monitoring in the Cloud . . 78
3.3.2 Methodology 81
3.3.3 Measurement Setup 81
3.3.4 Video Quality Monitoring in the Testbed Scenario . . . 86
3.3.5 Influence of VNF Placement on QoE Estimation 91
3.3.6 Behavior of the Video Buffer Monitoring VNF in the Real Scenario 96
3.4 Lesson Learned 98

4 Performance Evaluation of SFC Placement Algorithms in the Edge Cloud 101
4.1 Background and Related Work 104
4.1.1 The Emergence of Network Function Virtualization . . 104
4.1.2 Definition of Simulative Service Function Chain 110
4.1.3 Cloud Computing Simulator 112

4.1.4 State of the Art in Service Function Chaining 112
4.2 SFC Placement Algorithms 114
4.2.1 Centralization Algorithm 115
4.2.2 Orchestration Algorithm 116
4.2.3 Service Response Time and Resource Optimization . . . 118
4.3 Simulative Performance Evaluation of SFC Placement Algorithms 122
4.3.1 EdgeNetworkCloudSim Extension 122
4.3.2 Edge Cloud Topology 124
4.3.3 Service Chain Characteristics 126
4.3.4 Performance Metrics 127
4.3.5 Performance Evaluation of SFC Placement Algorithms . 128
4.4 Lesson Learned 138

5 Conclusion **141**

Acronyms **149**

Bibliography and References **155**

1 Introduction

Over the past decade, Internet services have evolved tremendously. The user is now in the focus, driven by new diverse possibilities of fast-growing and evolving Internet technology today. For instance, before the Dynamic Adaptive Streaming over HTTP (DASH) standard was published in 2012 [7], the user could only watch videos with a single level of quality via a progressive download. Today, users can have their own full HD adaptive video channel that can be displayed on any personal computer or mobile smart device. In addition to this, the emergence of cloud computing has revolutionized the Internet ecosystem by providing the users with everything as services [8]. This means, the user only needs a thin client to run an arbitrary type of cloud application that is centralized at the data center or distributed in the edge cloud. By moving desktop-based software into the cloud, the users can flexibly access their applications from anywhere, enjoy the best user experience, and take advantages of the scalability of the cloud paradigm with nearly unlimited resources. Moreover, the shared model in Software as a Service (SaaS) provides the users with lower cost of usage while accessing a shared cloud application and maintaining their own data in the personal cloud storage (e.g., Google Docs). All these advantages have led to an explosion of the cloud service subscriptions in recent years.

Despite the potential increasing in revenue, challenges the network operators are to deal with the problem of a high service demand nowadays while the capacity is limited. Moreover, to successfully compete for a share of a prominent market and retain the prospective users, the providers have to take the user experience into account. For example, a degradation of the service quality like a video interruption may induce user churn [9–11]. As a consequence, the

user may stop using that service and seeks for another provider. Therefore, the network and cloud service providers, more than ever, need to be aware of the user experience with their products. This not only helps to satisfy the users and increase the revenue, but also gives the ability to react with traffic management when a network impairment occurs. To this end, a monitoring mechanism is required to understand the degree of the user experience with the cloud services, which is one objective of this thesis.

In the Internet, a prerequisite to fulfill user requirements is that the network operators need to ensure a high Quality of Service (QoS) connection to the users. However, the network QoS parameters such as bandwidth, delay, or packet loss do not reflect the user perception or feelings rather than the physical network conditions. Therefore, a new concept that can translate the user experience into a measurable metric is required and defined in [12], called Quality of Experience (QoE). QoE is the degree of delight or annoyance of the user of an application or service. It is conventionally measured by subjective tests or objective studies. This thesis covers different aspects of objective QoE research that may help the network providers to understand the impact of the network QoS on the user satisfaction. Based on this, traffic management decisions can be performed to improve the network accordingly.

Although QoE is considered as a reliable indicator in assessing the level of user satisfaction, subjective QoE measurements are costly and time consuming since it requires recruited participants. Additionally, different cloud services have different objective and subjective characteristics for perception of quality [13]. For instance, QoE assessment for a cloud-based photo service can be performed based on photo loading time [3, 14, 15]. Whereas, QoE assessment for video streaming conventionally relies on stalling frequency and length [4, 16, 17]. This means, the assessment is highly dependent on the type of application or service. Thus, performing QoE assessment for every Internet service is even more expensive. To tackle this problem, objective QoE [11, 18] becomes an alternative solution to estimate the QoE.

Objective QoE refers to the attempt to quantify the user experience based on

analytical and statistical models. The input for these models can be the network layer parameters such as delay or packet loss, or application layer parameters like login time or photo loading time. There, a high end-to-end latency may cause a longer login time of a cloud service like Google Docs that also may dissatisfy the users. Similarly, the loading time of a photo also depends on the network condition on the path to the users. The longer path the photo traverses, the higher delay with possible packet loss occurs that negatively impacts the photo quality and loading time, so the QoE. In this situation, QoE assessment for these cloud services is necessary. To this end, first, the influence of the network QoS on the performance of the services needs to be analyzed and evaluated. The outcome of this step is a correlation between the service qualities (i.e., login or loading time) and the levels of network QoS (i.e., delay or packet loss). Then, the results are mapped with a pre-defined QoE model to specify the degree of user satisfaction depending on the network conditions. Based on this, a monitoring mechanism can be defined and network management can be performed to improve the QoE perceived by the user. For instance, the cloud photo service can be migrated to the edge cloud to decrease the latency.

One of the most popular and rich-data cloud services is HTTP Adaptive Video Streaming (HAS). In today's Internet, Cisco predicts that nearly a million minutes of video content will cross the network in every second [19]. This introduces a potential increase in revenue for the video providers but also challenges for the network operators ensuring the user expectation. Therefore, QoE monitoring for HAS has become a necessary tool for the network administrators to perform QoE management in the network. However, since HAS is a real time service, QoE monitoring and traffic management should be performed in real time as well. Additionally, the monitoring function should be executed in the network and the dynamic geographical deployment of the function may also be required for the mobile users. To fulfill these requirements, Network Function Virtualization (NFV) has emerged as a promising solution for a flexible, scalable, and cost saving deployment of such a QoE monitoring function [20]. NFV aims to decouple software-based network function from the underlying physi-

cal hardware. This piece of software is called Virtual Network Function (VNF) that can be installed in any standard commodity server. Based on this, the VNF QoE monitoring for HAS can be deployed at any Point of Presence (PoP) in the network or at the cloud data center. Then, QoE for HAS can be objectively monitored with a reasonable level of accuracy.

Despite the promising advantages of the NFV paradigm, the performance of VNFs in general and the VNF QoE monitoring for HAS has not clearly investigated. First, since the QoE metric for HAS is estimated based on monitoring the application layer parameters in the network, it is not really understood how the network QoS influences the accuracy of the estimation. Second, in the NFV paradigm, the VNF QoE monitoring can be deployed in any PoP across the network. It is important to know the side-effects of different VNF placements on its performance. Next, while the data center network typically has high capacity, the network impairments conventionally occur right at the user mobile access network. As a consequence, a video interruption might happen when the user is losing the signal from a cellular base station. This situation may become a bottleneck in estimating QoE if the monitoring VNF is operating outside this network segment and is unaware of the occurring network conditions. To cope with these problems, a new study on evaluating the performance of the VNF QoE monitoring for HAS is required.

In fact, a QoE management system typically consists of different functions such as QoE controller, QoE monitoring, and QoE manager [21]. Wherein, the QoE controller acquires the application data traffic. Then, the monitoring function estimates the QoE based on the parameters provided by the QoE controller. An estimation of the QoE for the monitored application is forwarded to the QoE manager where a traffic management decision is made accordingly. These functions are executed in a specific order and called Service Function Chain (SFC) [22]. In the NFV architecture, the SFC promises to reduce the complexity when deploying heterogeneous network services. However, the placement of each function in the chain must be well defined with respect to latency or server utilization, since QoE management must be quickly performed in the network.

Thus, in the context of cloud computing and NFV, several challenges in QoE assessment and monitoring exist and need to be investigated. This monograph presents solutions to cope with these problems and challenges. We present a QoE assessment method for two popular cloud applications, the performance evaluation of VNF QoE monitoring for HAS in the cloud, and the strengths and weaknesses of different placement algorithms for SFC in the edge cloud. The next sections highlights the main contributions and the outline of this work.

1.1 Scientific Contributions

Figure 1.1 shows the main content structure and contributions of this thesis. Each circle with different colors indicates an individual research topic presented in corresponding content chapter. However, these topics are also relevant to each other as depicted by the arrow of the circle.

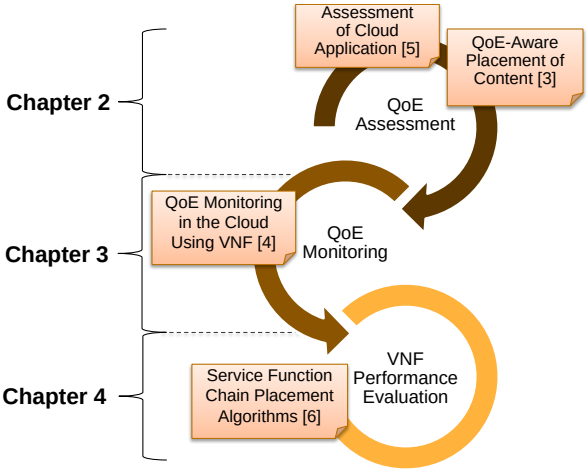


Figure 1.1: Overview of the Contributions of the Thesis

From the top of the figure, this monograph focuses on three main research areas, which are the *QoE assessment* of the cloud applications, the investigation of applying the NFV paradigm for *QoE monitoring* in the cloud, and *performance evaluation* of different placement algorithms for SFC. A majority part of these works have been conducted within the *EU Horizon 2020 (H2020-2014-ICT-644672) project In-Network Programmability for Next-Generation Personal Cloud Service Support (INPUT)* [23].

The first contribution of this dissertation is the evaluation of the impact of network QoS parameters on the performance of Google Docs and cloud-based photo services. Herein, the impact of delay and packet loss on different subprocesses of Google Docs service is evaluated through testbed experiments. The derived linear regression models can be further used in reference models for QoE assessment of such an exemplary SaaS solution. Regarding the cloud-based photo service, we determine the trade-off between the size of photo and its geographical placement to acquire a high QoE for photo loading time. To this end, we propose a mapping equation which is based on previous TCP and QoE models. This formulation is able to assess the QoE for photo loading time depending on different network QoS parameters and the distance between the user and the location of the service. This insight may help the providers to have another strategy to deploy such a photo service with respect to QoE.

Next, we conduct real-world experiments in the field of QoE monitoring for HAS as the second contribution. We propose to use VNF for monitoring video traffic in the network to leverage the advantages of the NFV paradigm. To this end, we design a VNF using deep packet inspection technique and an algorithm to estimate the video quality and QoE based on application layer parameters. Then, we evaluate the performance of the VNF in different deployment scenarios under the side-effects of the network QoS and the virtual environment of the cloud architecture. Our insights show that the different geographical deployments of the VNF influence the accuracy in estimating the video quality and QoE. Specially, packet re-ordering and mobile network can induce an overestimation of QoE for HAS. In this situation, moving the VNF next to the user

is highly recommended to achieve an accurate QoE estimation. This contribution may help the network providers to further understand the advantages and drawbacks of deploying such a VNF QoE monitoring in the cloud.

As the third contribution, we focus on, is a scenario where different VNFs interwork in a specific order to form a SFC. Herein, we propose four algorithms to place the VNFs across data centers in the edge cloud. We use the EdgeNetworkCloudSim simulator [24] to simulate users accessing the SFCs and the performance of the placement algorithms is evaluated with respect to service response time or server utilization. Our results show that the optimized solutions produced by using Integer Linear Programing (ILP) model obtain lowest service response time and least server utilization rate compared to the heuristic approaches. However, when the solution space is large due to a large network topology, the placement problems is *NP-hard* that requires a longer solving time. As a consequence, the service response time is negatively influenced. In this case, heuristic approaches can be the alternatives. This may help the network administrators to have different solutions for SFC placement in the edge cloud.

1.2 Outline of the Thesis

The outline of this thesis is depicted in Figure 1.2. From the top of the figure, after the introductory section, the scientific contributions and the organization of the thesis are presented in Chapter 1. Then, the main outcome of this work is described in detail in separate chapters.

In Chapter 2, we present QoE assessment methods for the two popular cloud applications, which are Google Docs and photo service. First, the SaaS service model in the cloud paradigm and basic concepts are introduced in Section 2.1. Several related studies are also addressed in this section. Subsequently, Section 2.2 assesses the influence of different network QoS parameters on the performance of Google Docs service. The investigation of the trade-off between the content size, service geographical placement, and QoE is detailed in Section 2.3. Finally, Section 2.4 summaries the main findings of this chapter.

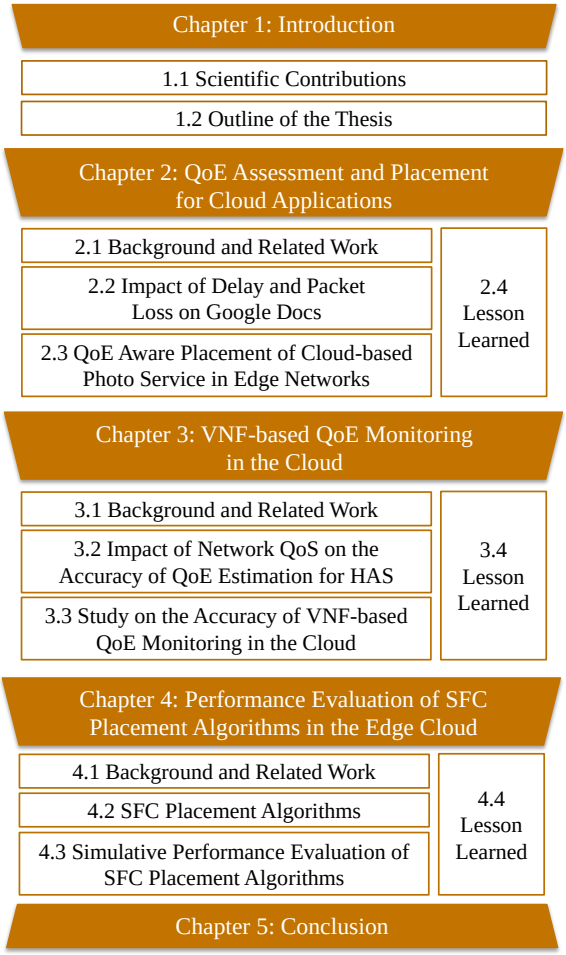


Figure 1.2: Dissertation Outline

Chapter 3 focuses on the QoE monitoring for HAS in the network using VNF. Different QoE assessment methods, QoE monitoring at different layers, and previous studies are highlighted in Section 3.1. In Section 3.2, we propose the use of VNF for QoE monitoring using deep packet inspection technique and design an algorithm to estimated the video quality based on application layers parameters. The performance of the VNF QoE monitoring for HAS is evaluated in a practical emulation testbed. Section 3.3 investigates the side-effects of deploying the VNF QoE monitoring in the Amazon Web Service cloud environment. The impact of mobile access network on the accuracy of the VNF is also examined in this section. Our insights about the performance of the VNF QoE monitoring for HAS is summarized in Section 3.4.

Chapter 4 investigates the performance of different placement algorithms for SFC in the context of the edge cloud. There, we first introduce the NFV paradigm and the state of the art in SFC research area in Section 4.1. Subsequently, four placement algorithms are presented in Section 4.2 consisting of two heuristic approaches and optimized solutions obtained by using the ILP model. In Section 4.3, we first introduce the extension of EdgeNetworkCloudSim simulator and the network topology for the simulation. Thereafter, several SFC characteristics and performance metrics are given. The performance evaluation of different placement algorithms with respect to the service response time and server utilization is detailed in Section 4.3.5. Section 4.4 concludes this chapter with lessons learned.

Finally, Chapter 5 highlights the major contributions of this monograph and gives an outlook to future research works.

2 QoE Assessment and Placement for Cloud Applications

Cloud computing and Software as a Service (SaaS) have received considerable interest by both the research and the industrial community. Cloud computing refers to a pool of configurable, virtualized resources such as computation, memory, storage, and network that can be dynamically allocated for applications, services, and user virtual machines. In cloud computing, SaaS provides the user with software that can be accessed anywhere by using a thin client or a web browser over the Internet.

The benefit of SaaS is scalability and the provisioning of almost unlimited resources compared to legacy software solutions. Unlike locally launched applications, since cloud applications are hosted in data centers, the quality of provisioned applications is no longer only dependent on the cloud provider infrastructure and allocated resources, but also on the condition of the network connection to the user. For instance, the users who are far away from the data center might experience a lower quality connection with much delay compared to the one who is nearby. Moreover, users typically are not interested in technical problems of the network such as delay, packet loss rate, or throughput. They only care about the performance on the application layer.

As a consequence, a network problem may cause a bad experience perceived by the user when using even a well-designed cloud application. Thus, network operators now, more than ever, need to understand the expectation of the user and provide services accordingly if they want to support cloud applications. In other words, technology-centric network management is no longer the only ap-

proach in today's Internet, but rather user-centric approaches play an increasingly important role.

For this reason, Quality of Experience (QoE) has emerged as a popular topic in recent years. QoE is a measure of satisfaction or annoyance of a user with a particular application or service [12]. QoE for a service reflects the degree of human expectations, feelings, or perceptions for that service. Thus, it is important to be aware of the QoE for an application or a service. Being able to estimate the QoE perceived by the users for a particular application will help the network operators to understand what is maybe wrong with the application and they can react to improve it.

To measure the QoE for a specific application, the QoE influence factors must be taken into account. The authors in [25] classify QoE into three categories of *human*, *context*, and *system* influence factors, in which *human* factors refer to age, gender, or visual and auditory acuity. *Context* factors are spatial or temporal, economic or social factors that influences the QoE. Among the others, *system* influence factors receive the most considerable interest by the research community. It is primarily due to a strong relationship between the QoE and the Quality of Service (QoS) in network engineering domains [11] [26]. *System* influence factors consist of application layer factors (e.g., media content, encoding, or resolution), network layer factors (e.g., latency, packet loss rate, or throughput), and physical layer factors (e.g., transmission media or user devices). Since the QoE assessment based on *human* and *context* influence factors is costly, network operators can efficiently estimate the QoE based on *system* influence factors that can be measured using a dedicated testbed.

Since cloud applications are hosted in data center and delivered to the users over the Internet, the performance of the cloud applications is highly dependent on the network condition. To ensure the user satisfaction with the applications, new studies on the QoE for the cloud-based applications are required. Especially, the influence of the network connection on the QoE perceived by the end user need to be taken into account. In this chapter, we tackle the problem by studying the QoE assessment based on *system* influence network factors which

are latency, packet loss rate, and link capacity of the network connection to the user. Specifically, we first investigate the impact of delay and packet loss on the performance of a cloud-based word processor namely Google Docs. This is the first step to further evaluate the influence of network factors on the QoE for the cloud-based word processor service. Secondly, we focus on the influence of content placement on the QoE. This corresponds in particular to the proximity of the service to the user. To achieve the goal, we study a cloud-based photo service and investigate the relationship between the photo loading time and the QoE perceived by the user. We find out that the geographical placement of the content is one of the key factors that affects the user satisfaction. The closer the content to the user geographically is, the faster it will be delivered to the user that will also increase the user perceived QoE. Thereby, migrating the content near to the users is a prominent approach in this situation. For instance, Figure 2.1 shows an overview of possible content placements in the cloud data center or edge server to adjust the QoE.

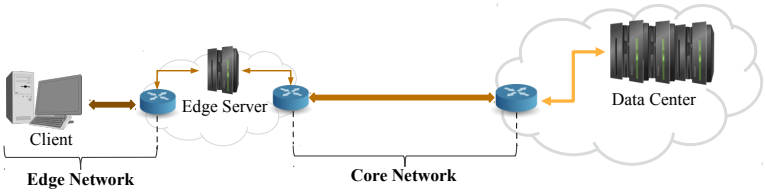


Figure 2.1: Overview of Possible Content Placements

In this figure, the content is assumed primarily to be stored in a data center and can be accessed anywhere. The users access the content over the Internet, in which the locations of the users are different from each other. If the users are far away from the data center, their access might be affected by a high latency and possibly network congestions. In this case, the loading time of the content might be significantly higher than the one who is closer. The users therefore experi-

ence a lower quality (e.g., lower photo resolution due to progressive download) and they may stop using the service. A possible solution is to migrate the content closer to the users. To this end, the network operators need to be aware of the QoE perceived by the users for the service at first. If a low QoE is encountered, the content is then managed to migrate to edge server as shown in the Figure 2.1. By doing this, the problem of low QoE due to long distance access is solved. Although Content Delivery Network (CDN) [27] [28] can also act similarly, this network technology only caches a part of content on the edge server. Additionally, replicating the same content over the Internet is not always an efficient solution, since it increases overhead and the cached content may not be the one that the users are interested in. Thus, a better solution for the network providers is to investigate the QoE for the service, then manage the network and the content placement accordingly to improve the user satisfaction.

The contribution of this chapter is twofold. First, we analyze the performance of a cloud-based word processor, namely Google Docs regarding the identified performance metrics with respect to QoE. Herein, we provide a model used to derive Google Docs performance metrics given a set of network parameters and quantify the goodness of fit. This contribution is the first step to further evaluate the QoE for such a cloud-based application. Especially, the derived model can be further used in analytical models for optimization or estimate the QoE for Google Docs based on monitored network QoS parameters. Secondly, we propose a method to investigate the relationship between the content placement and network parameters to derive a QoE mapping model for the content loading time. This contribution may help the network providers to have an alternative approach to monitor the QoE for content loading time based on various parameters that is also important in QoE and network management.

The content of this chapter is mainly taken from [5], [3]. The remainder of this chapter is structured as follows. Section 2.1 highlights the background and related work that is divided into four different subsections. Firstly, SaaS deployment model of cloud computing is introduced in Section 2.1.1. Subsequently, Section 2.1.2 presents a cloud-based photo service in the context of edge net-

works, followed by the discussion about the relationship between QoE and network QoS in Section 2.1.3. Thereafter, Google Docs and its technology behind is described in Section 2.1.4 adding related work on the impact of network conditions on word processors in general. In Section 2.2, we evaluate the impact of different network parameters on the performance of Google Docs service in both single and collaborative scenarios. Next, Section 2.3 presents the trade-off between the QoE for photo loading time and its placement in the cloud by using a mapping model. Finally, Section 2.4 concludes this chapter with lesson learned.

2.1 Background and Related Work

In this section, we first present the SaaS architecture in Section 2.1.1. Then, we introduce the two popular cloud-based services, namely photo album and Google Docs. Firstly, we present an overview of photo album service in the context of edge network and discuss the relationship between QoS and QoE in Section 2.1.2 and Section 2.1.3, respectively. Thereafter, the technology behind Google Docs and researches on the impact of network conditions on word processors in general is presented in Section 2.1.4.

2.1.1 Software as a Service Architecture

The paradigm of SaaS has gained great attention in the last decade. SaaS allows end users to use complex software directly from their browsers, transferring heavy computation to servers in the cloud. By doing this, users benefit from using demanding services on lightweight devices without installation and operation of applications on their own computer. Moreover, service providers can also benefit from simplifying service maintenance, management, and support.

In SaaS, most of solutions are based on multi-tenancy model. With this model, one application instance can be operated for multiple enterprises and provided for multi-tenants that can reduce tremendously capital and operational costs. Figure 2.2 shows an overview of multi-tenancy model where users can either

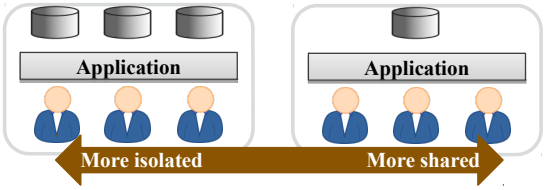


Figure 2.2: Overview of SaaS Multi-Tenancy Model

choose a more isolated model with their own database or share the database to reduce cost. On the one side, SaaS applications can be delivered to the tenants with different degree of isolation depending on business requirements. In this model, each user can own an individual database that is completely separated from the others to ensure privacy and security (e.g., a personal photo album). The cost of ownership is therefore higher than in the shared model. On the other side, with the success of social network, multi-tenant architecture with shared model also allows users to collaborate and share information in a joint space (e.g., collaborative editing in Google Docs). This model is the most preferred approach where one database can be shared for multiple users. Hereby, the cost and maintenance of the share model is significantly lower.

Although SaaS paradigm brings advantages and reduces costs, one of the main drawbacks of SaaS is latency. With the centralized architecture of the cloud computing, users who are far away from data centers will face the problem of network latency. This may influence the delay-sensitive applications (e.g., live streaming, online gaming, or collaborative wording). As a consequence, users may dissatisfy and stop using the applications, so decrease the QoE. In the followings, we introduce the two well-known SaaS applications and analyze the influence of different network conditions and content placement on the applications and the level of QoE perceived by the users in the next sections.

2.1.2 Cloud-based Photo Service in the Context of Egde Networks

Photo has been one of the most popular content delivered over the Internet in the last decades. In 2016, it was reported that over 95 million photos are shared on Instagram every day (Instagram Inc., 2016). To store and share this type of content, the common way is using a web-based photo album in the cloud. Indeed, along with the increasing diversity of SaaS, a trend is the replacement of entertainment applications running on PC by a SaaS (e.g., online cloud gaming, YouTube video streaming). An Edge network Photo album Cloud service (EPC) is another example. While a desktop-based album application manages and stores photos predominantly on a PC, an cloud-based album provides almost unlimited space to store the user photos, accessible everywhere. Furthermore, an edge network cloud service refers to a location-aware, flexible placement of the service, and the content among multiple resources in the cloud and in the edge network. This means, the service providers can decide to place the EPC in a resource-efficient manner, such that the user perceived QoE for the photo album service is high.

An EPC is a SaaS, which allows users to upload and manage photos created by any digital device (e.g., digital camera, smart phone, etc). As a web-based service, an EPC typically uses HTTP or HTTPs over TCP to deliver stored photos over the Internet. Users can access and manage an EPC using any modern web browser. Thus, the photo loading time is influenced primarily by the file size, the distance between server and client, and the network QoS. If the user has to wait too long to view or upload a photo, the user may stop using the service. In order to achieve a high satisfaction with the photo album service, the challenge is to efficiently place the photo content in appropriate geographical location to achieve a high QoE perceived by the user. In other words, the trade-off between user perceived QoE, network QoS, and placement of content is an important factor for developing such an EPC service.

2.1.3 Relationship Between Network QoS and Quality of Experience

As defined in [12], "Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service". The definition is possibly suitable in the context of cloud-based multimedia services. In this context, QoE is the level of user satisfaction and/or enjoyment of an application or a service. The QoE is typically evaluated using Mean Opinion Score (MOS) [29].

In fact, the QoE evaluation of a cloud service, as well as the relationship between QoE and QoS, is widely studied. However, we observe rare research about the QoE-aware placement of content for cloud services. The concept of QoE refers to the overall level of customer enjoyability with a service is introduced in [30]. Regarding the web-based services, QoE has a strong relationship with the network QoS. The IQX hypothesis is proposed in [11], which described a natural and generic exponential relationship between QoE and network QoS. Meanwhile, the authors in [26] reported a logarithmic relationship between QoE and network QoS. However, the relationship between QoE of a specific application and network QoS highly depends on the application.

In [31], Mok et al. investigated the relationship between QoE for HTTP Adaptive Video Streaming (HAS) and network QoS using analytical models and empirical evaluation. In [32], Casas et al. provide a result of concrete cloud QoE studies, in which Cloud Storage and File Synchronization, Remote Virtual Desktop and telepresence system such as Microsoft Lync Online were conducted by subjective lab experiments. While, HAS (e.g., YouTube) is evaluated by field trials approach. The relationship between waiting times of interactive data services and QoE is discussed in [14] and [15]. The authors focus on the time perception and its relation to the user satisfaction rather than the trade-off between the placement of content and QoE as our main consideration. Nevertheless, the authors explained a logarithmic relationship between user perceived QoE and photo loading time which benefits us as a QoE reference model as it will be described in more detail in Section 2.3.

2.1.4 Cloud-based Collaborative Word Processor

Another popular cloud-based service, used for multiple purposes is Google Docs. This cloud SaaS is a web-based word processing application whose client side front end is based on HTML and JavaScript and can be accessed using any modern web browser [33]. In contrast to standalone office software products, e.g., Microsoft Office or LibreOffice, Google Docs requires a permanent Internet connection as documents are not stored locally on the client but on the Google server infrastructure. Google Docs does not provide rich feature sets like stand alone office products, however, it offers an easy way to share documents and enable collaboratively editing with up to 10 users simultaneously by sharing a link to the document or granting explicit rights to other registered users. Nevertheless, since Google Docs is an Internet-based word processor, its performance might be influenced by different network conditions and to the best of our knowledge, the evaluation of the quality of service for Google Docs has not been taken into account so far.

Considering the impact of network conditions on other network-based word processors, Schlosser et al. analyze the behavior of Microsoft Word and Excel running in a remote desktop environment under different network conditions [34]. They consider the Microsoft Remote Desktop Protocol (RDP) and Citrix Presentation Server (CPS) as possible thin-client solutions. Their results show that delay less than $500\,\mathrm{ms}$ or packet loss less than $2\,\%$ does not have any influence. However, the combination of delay and packet loss results in measurable impairments.

In [35], the authors focus on how Input Buffer and Speedscreen options can improve the performance of CPS in a WAN scenario. They perform measurements with a user typing a text, scrolling a text, and selecting specific sub-menus on Microsoft Word and Textpad, respectively. The test duration under different network conditions is the main criteria to evaluate the performance of CPS. From the results, the author conclude that with an increasing of network delay up to $500\,\mathrm{ms}$ in combination with packet loss lower than $2\,\%$, CPS with the combination of Speedscreen and Input Buffer takes less time to finish the test

than without these options.

In both studies, the applied methodology is similar to the one used in this work. However, we are not focusing on traditional thin-clients but rather study a web based solution. Further, we also consider a collaborative use case, which is not studied in previous publications. Other works study cloud services and theirs network requirements. In [36], the authors focus on five fundamental challenges for wide adoption of cloud computing using the *OPTIMIS* toolkit. Amrehn et al. conclude that for general file storage services, the upload and download speed, financial aspects, privacy, and security are important QoE influence factors [37]. The authors in [38] use a prediction system to forecast the CPU demands for web based cloud services. Other studies evaluate the subjective user satisfaction, i.e. QoE, with cloud services [39, 40]. These studies focus on different aspects of cloud computing. However, the authors do not evaluate a specific cloud application or investigate the impact of network conditions on the performance of cloud applications.

2.2 Impact of Delay and Packet Loss on Google Docs

While a traditional desktop word processing application such as Microsoft Word provides a more complete feature set, Google Docs is a lightweight utility with sufficient office features and high flexibility. As an additional feature, Google Docs enables users to share created documents with other users or even collaboratively edit them. However, as Internet-based cloud application, the performance of Google Docs depends on the network quality between server and client. In this section, we evaluate the performance of Google Docs with regard to different network conditions in two scenarios. First, a *single user scenario* is studied. In this scenario, a user has to take several subprocesses such as *Login* or *Typing*. We consider the time required to complete the subprocesses as a metric for the performance of the service. In the *collaborative scenario*, with two users login to Google Docs. The first user edits a document while the other user observes the editing. Here, we consider the time both users require to complete the

total process as well as all composite subprocesses as a measure of the application performance. To evaluate the influence of different network conditions on the processing time in both scenarios, we emulate various network delay conditions and packet loss settings in a local testbed that is described in Section 2.2.1. In the sequence, the impact of different network conditions on scenarios is analyzed in Section 2.2.2 to 2.2.4.

2.2.1 Methodology and Testbed Setup

We use a dedicated testbed including a network emulator to analyze the influence of different network parameters on the behavior of Google Docs, allowing for an easy adaption of network parameters such as delay and packet loss. In the following we first detail on the methodology and then test setups for the single user case and the collaborative scenario.

Method

In the reminder of this section, we analyze how network parameters influence the behavior of Google Docs. To assess this in an objective manner, we are going to emulate user interactions and measure the time it takes to complete them. We consider two scenarios, which are derived from common Google Docs use cases. First, we discuss the single user scenario with one user editing a document. Here, a session is divided in five steps, which we will refer to as *subprocesses*. In a first step, the user logs into the system to gain access to a previously created document or to create a new document (*Login*). In the next step, the user creates a new document (*Creating*). Then, the user starts typing while the client continuously sends updates to the server to stores entered text at the server (*Typing*). After entering the text, it takes a short amount of time to save the last changes to the text (*Saving*). The session is then ended with the logout of the user (*Logout*).

The durations Δt_{login}, $\Delta t_{\text{creating}}$, Δt_{typing}, Δt_{saving}, and Δt_{logout} of the five subprocesses *Login*, *Creating*, *Typing*, *Saving*, and *Logout*, as well as the total time of the session Δt_{total} are considered as an objective metric to assess the impact

of network conditions on the performance of Google Docs. While it is intuitive that most of the aforementioned metrics depend on the network parameters, Emmert et al. [41] showed that the effective typing speed of an user also depends on network parameters in thin client environments. We will refer to this scenario as *single user scenario* in the remainder of this section.

As mentioned before, one of the major benefits of Google Docs is collaborative editing. In this case, the user session is more complex than in the single user case. For this scenario, we assume that *user 1* is creating the document and shares it with a collaboration partner *user 2*. Therefore, the work flows of *user 1* and *user 2* are almost similar to the work flow in the single user scenario. However, to share the document, *user 1* sends a link to *user 2* which grants him access rights to the newly created document. In this scenario, we additionally define two waiting times: (1) the duration *user 1* has to wait until *user 2* is ready to receive text, (2) the duration that *user 2* has to wait until *Receiving* starts. After *user 2* accessed the document, *user 1* starts writing and the content is automatically synchronized with *user 2* via Google Docs. As *user 2* is not actively editing the document, he does not observe a *Saving* phase.

Testbed Setup for Single User Scenario

The testbed setup for the single user scenario is schematically depicted in Figure 2.3. It consists of one measurement server, one network emulator, and a control PC. The measurement server hosts the virtual machine VM1 used as Google Docs client for *user 1*. The virtual machine is connected to the Internet via another server running NetEm[1], which enables us to adjust packet loss and delay on the connection. To control the measurements, we use a control PC that is connected to the network emulator and the Google Docs client via a dedicated control network to avoid interference with the tests.

The measurement server and the network emulator are SUN FIRE X4150 servers with 8 CPUs 2.5 GHz, 16Gb RAM, and 4 Ethernet 1Gbps NICs. VMware

[1] http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

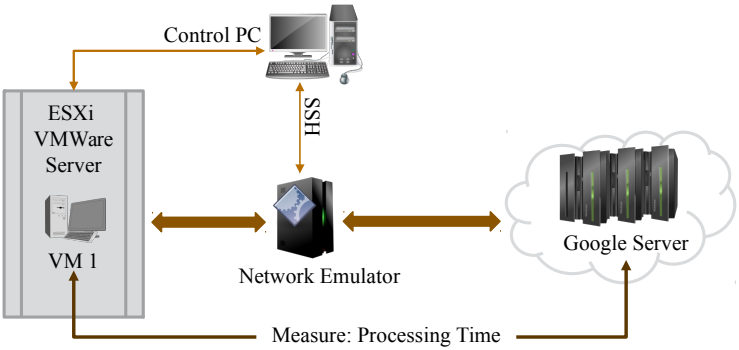


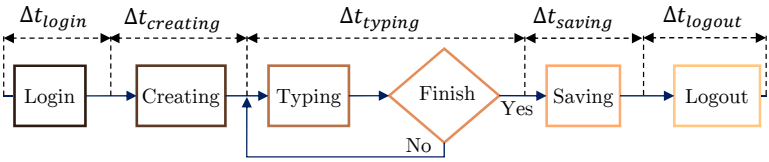
Figure 2.3: Overview of Testbed Setup for the Single User Scenario

ESXi 5.5[2] is used as virtualization solution and both the Google Docs clients and the network emulator use Ubuntu 12.04 LTS as operation system. The testbed is connected to the Internet with a research network. We measure the baseline network parameters with a round trip time of 3.91 ms and no packet loss over 1000 packets. For later evaluation, we consider network delays from that baseline up to 1000 ms. Such high delay values can, e.g., occur due to long distance Internet access [42] or bottlenecks [43]. We consider packet loss from the baseline up to 4% which may occur in a wireless link in urban area [44].

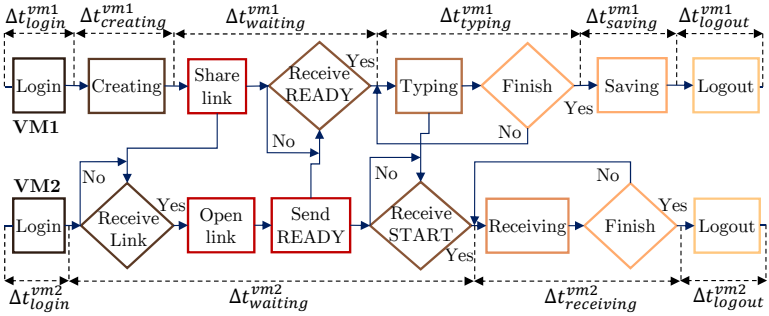
As discussed in Section 2.1.4, we assess the influence of the network parameters by measuring the duration of the subprocesses. To this end we use the Selenium Webdriver[3] to automatically generate user interactions. Figure 2.4a depicts the program flow of the measurement script and the recording of the time stamps used for measuring the duration of the subprocesses.

First the control PC sets up the network emulator with the desired configuration. Thereafter the Selenium script is started, which signs in to Google Docs

[2]https://www.vmware.com
[3]http://www.seleniumhq.org



(a) Single User Scenario



(b) Collaborative Scenario

Figure 2.4: Measurement Workflows

and creates a new document. The content entered by the script is an English text taken from the introduction part of Selenium webpage. To evaluate the influence of the length of the text of the duration of the typing process, we use a short text of 1548 characters, which corresponds approximately one paragraph in a document. Besides this we also use a long text with about 6189 characters, which corresponds to about two pages of A4 document. After the automatic typing is complete, the Selenium script waits until the document is saved and logs out of the Google Docs. For each network parameter setting we produce 50 replications within several days to avoid measuring diurnal effects.

Testbed Setup for Collaborative Task Scenario

In the collaborative scenario we consider two users working simultaneously on the same document, with one user editing the content of the document and the other user reading the document. To analyze this scenario we extend the testbed configuration described in Figure 2.3 by adding another virtual machine (VM2) as *user 2* on the measurement server. In this scenario we require synchronized clocks for both client PCs. While this is challenging when using two different physical machines, it can be accomplished using two virtual machines sharing the host clock. Similar to VM1, VM2 is connected to the Internet via the network emulator, so that both VMs share the same network parameters. VM2 is also connected to the control PC using a dedicated control network. Additionally a second control network is established between the two virtual machines to synchronize the workflows of the machines as describe below. In the measurement, we use short sample text from the single user scenario and the same network settings.

Figure 2.4b shows the workflow in the collaborative scenario. The upper and the lower part of figure represents the processes on VM1 and VM2, respectively. The workflow for VM1 is similar to the one in the single user scenario. However, after creating the new document, VM1 shared the document with VM2 by sending a link. The workflow of VM2 differs in such a way that VM2 does not create a new document itself, but just waits for the link to the shared document. In or-

der to synchronize the workflows of the two virtual machines, VM1 waits after sending the link to VM2, until VM2 places a marker in the shared document. Thereafter, VM1 starts typing in the document and VM2 observes the changes.

In addition to the times measured in the single user scenario, we also consider the waiting times of the two virtual machines in this case. This is for VM1 the time between creating the document and the notification from VM2 that it successfully accessed the shared document, and for VM2 the time between logging in and observing the first changed by VM1 in the shared document. Moreover, we also measure the time it takes until all changes on the document made by VM1 are visible on the document seen by VM2.

2.2.2 Impact of Different Network Conditions on Subprocesses in Single User Measurements

Based on the methodology and measurement setup discussed in Section 2.2.1, we study the impact of network parameters, i.e. packet loss and delay, and text length on the single user and collaboration scenarios, with regard to the subprocess and total durations introduced earlier. All measurements were performed between February 12, 2015 and March 24, 2015. For each parameter setting 50 repetitions of the measurement were performed, in order to increase statistical significance. The measurement settings are chosen according to the values discussed in Section 2.2.1. In order to avoid measuring diurnal effects, we did not perform measurements with the similar settings consecutively, but distributed them over different times of day.

In this section, we first investigate the single user scenario and show the results in Figure 2.5. For all figures, the y-axis gives the subprocess duration with 95% confidence intervals in seconds. For sake of readabilities the y-axis is cropped to $30\,\mathrm{s}$, but the measurement values are given in the figure description.

In Figures 2.5a and 2.5b, we study the impact of different network parameters for different text lengths. Here, the x-axis in the left sub figures shows the different delay settings in milliseconds, from the baseline unmodified delay, to

an additional delay of 1000 ms in increments of 250 ms. The right sub figures show the impact of packet loss on the single user scenario. Here, the x-axis gives the additional induced packet-loss from the baseline setting without additional packet loss, up to 4% in increments of 1%. All figures give subprocess durations as bars, colored depending on the subprocess type. Additionally, for each subprocess a linear regression is performed, which is shown as a colored line, depending on the subprocess type. Table 2.1 shows the detailed results of the linear regression depending on the delay, including the Coefficient of Determination (CoD) r^2 as a measure for the goodness of fit.

Figure 2.5a and Figure 2.5b show the packet loss in the considered range up to 4 % does not affect the processing time for any subprocess. However, the increase of network delay results in increased processing times for all subprocesses except the *Typing* time, with *Login* and *Creating* document being the most sensitive to network delays. When delay increases from baseline to 1000 ms, the *Typing* time remains almost constant at 60 s and 247 s for the short and the long text, respectively. This is due to the fact that updates to the server are sent asynchronously and the typing process does not depend on the reply of the server. Particularly, the duration of the *Login* process is about 7 times longer than for a delay of 500 ms then for the baseline measurement and 12 times longer at a delay of 1000 ms.

The duration of the *Creating* document process doubles and almost triples for the corresponding delay values in comparison to baseline measurement. This is due to the fact that the *Login* and *Creating* subprocesses rely on multiple communications between client and server which are executed in serial order. In contrast to this, the *Saving* time only slightly increases and the *Logout* time takes approximately 3.30 s at 1000 ms delay compared to 0.60 s at baseline delay. Due to the synchronization of the typed text in a background process, the saving of a document relies only on few communications with the server and thus is not influenced by a large measure. In the measurement of the long text as shown in Figure 2.5b, the behavior of the *Login*, *Creating*, *Saving*, and *Logout* subprocesses is similar to the behavior observed for the short text.

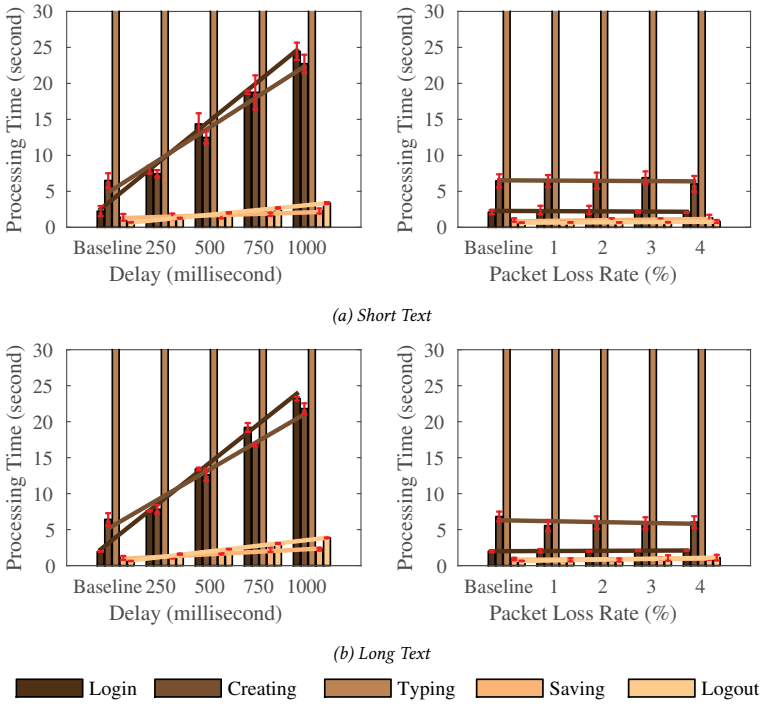


Figure 2.5: Impact of Network Conditions on Subprocess Durations in Single User Scenario

Table 2.1 summarizes the results of the linear regression for the subprocesses, both for short text and long text measurements for a given delay x. Packet loss is not considered, as the impact on subprocess duration is negligible in this scenario. We observe that increasing delay results in a large increase of the *Login* time and *Creating* time, while the effect on the other subprocesses is less significant. The linear function of *Typing* time has small slope coefficient of 3.70 compared to its intercept of 56.71. Therefore, the length of text is primary factor changing the *Typing* time, not the network delay or the packet loss. The durations of the *Typing* subprocess vary non-linearly with increasing delay, resulting in an inadequate fit and a low CoD value.

Table 2.1: Linear Regression of Subprocesses for Delays in Single User Scenario

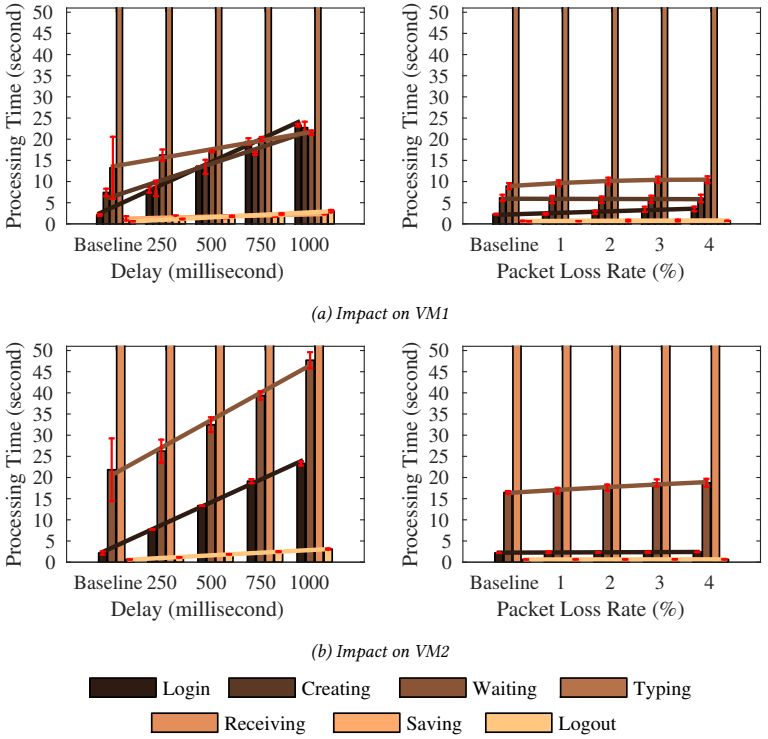
Subprocesses	Short-text Measurement	r^2	Long-text Measurement	r^2
Login	$22.07 \times 10^{-3} \cdot x + 2.54$	0.99	$21.64 \times 10^{-3} \cdot x + 2.26$	0.99
Creating	$17.50 \times 10^{-3} \cdot x + 4.83$	0.96	$15.89 \times 10^{-3} \cdot x + 5.14$	0.94
Typing	$3.70 \times 10^{-3} \cdot x + 56.71$	0.49	$-4.95 \times 10^{-3} \cdot x + 247.35$	0.32
Saving	$0.84 \times 10^{-3} \cdot x + 1.29$	0.87	$1.41 \times 10^{-3} \cdot x + 0.95$	0.92
Logout	$2.70 \times 10^{-3} \cdot x + 0.65$	0.99	$3.16 \times 10^{-3} \cdot x + 0.71$	0.99

Our measurements show that in the single user scenario, Google Docs is robust against packet loss. However, delay affects the system negatively, especially during processes depending on multiple serial communication between client and server, e.g. login or while creating new documents. The actual typing process, which represents interaction between the user and the client, is insensitive to the network conditions, as it is basically a background process, which does not affect the user directly. The measurements show that the duration of the typing process is mainly depending on the length of the text.

2.2.3 Impact of Different Network Conditions on Subprocesses in Collaborative Task

We now analyze the impact of different network conditions on the subprocesses in the collaborative scenario. As described in Section 2.2.1, we use two virtual machines. The document is edited by VM1 while VM2 observes the creation process. In Figure 2.6, the y-axis shows the duration of each subprocess in seconds, while the x-axis in the left figures shows the network delay and the packet loss rates in the right figures. Bars are colored by subprocesses and give the mean and 95% confidence interval of duration, the lines show the linear regression.

Similar to the results from Section 2.2.2, Figure 2.6a and Figure 2.6b indicate that a packet loss of less then 4% has no influence on the observed subprocess durations. Increasing delay results in an increasing duration of almost all subprocesses in both VM1 and VM2, with *Login*, *Creating* document and *Waiting* being the most sensitive processes. In contrast to this, the *Typing* time on VM1 and *Receiving* time in VM2 are only slightly fluctuating around $60\,\mathrm{s}$ even for higher delays. Again, this is due to the fact that synchronization between both VMs occurs asynchronously and does not depend on the responses of the server. As expected, the *Login* times are similar for both machines, since both experience the same network parameters. Furthermore, we observe that the *Waiting* time for VM2 is approximately the sum of the *Creating* and *Waiting* time of VM1. This can be explained, by the fact that both machines start with the login process at about the same time but VM1 has to create the document first. The following starts the synchronization process for both workflows, c.f. Figure 2.4b, which ends with the start of the *Typing* process on VM1 and the start of the *Receiving* process on VM2. These process again mark the end of the waiting periods of both machines. Interestingly, the *Typing* and *Receiving* process take about the same amount of time on both machines, independent of the network conditions. Again the parameters for the linear regression models are summarized in Table 2.2. The missing values in the table indicate the subprocesses does not occur on the corresponding virtual machine.



(a) Impact on VM1

(b) Impact on VM2

Figure 2.6: Impact of Network Conditions on Subprocess Durations in Collaborative Scenario

Table 2.2: Linear Regression of Subprocesses for Delay in Collaborative Scenario

Subprocesses	VM1	r^2	VM2	r^2
Login	$21.53 \times 10^{-3} \cdot x + 2.51$	0.99	$21.47 \times 10^{-3} \cdot x + 2.41$	0.99
Logout	$2.39 \times 10^{-3} \cdot x + 0.59$	0.99	$2.50 \times 10^{-3} \cdot x + 0.58$	0.99
Creating	$15.56 \times 10^{-3} \cdot x + 5.95$	0.96	-	-
Typing	$-1.73 \times 10^{-3} \cdot x + 59.68$	0.29	-	-
Saving	$1.07 \times 10^{-3} \cdot x + 1.24$	0.83	-	-
Waiting	$8.16 \times 10^{-3} \cdot x + 13.57$	0.97	$25.94 \times 10^{-3} \cdot x + 20.56$	0.98
Receiving	-	-	$-2.81 \times 10^{-3} \cdot x + 60.08$	0.49

Considering the CoD, we again observe that the *Login*, *Logout*, *Creating*, and *Saving* subprocess can be fit using a linear model, in contrast to the *Typing* and *Receiving* which do not show linear behavior regarding the considered parameters. For the *Typing* and *Receiving* process the intercept is again much larger then the slope coefficient, which indicates that the network parameters again have only little influence on the subprocess durations. Again, we do not provide a linear regression concerning packet loss due to the negligible impact of the variable.

Our measurements show that in the collaborative scenario, Google Docs behaves similar to the single user case. It is rather robust against packet loss and more sensitive to delay. Processes depending on repeated communication between client and server are more affected by additional delay, then e.g., the typing process which uses an asynchronous communication pattern.

2.2.4 Impact of Delay and Packet Loss on Total Process in Collaborative Task

After analyzing delay and packet loss separately, we now consider the total process duration given packet loss and delay occurring at the same time. We consider total processing time Δt_{total} required for inputting the short text in the collaborative scenario. The results for the measurement show that the differ-

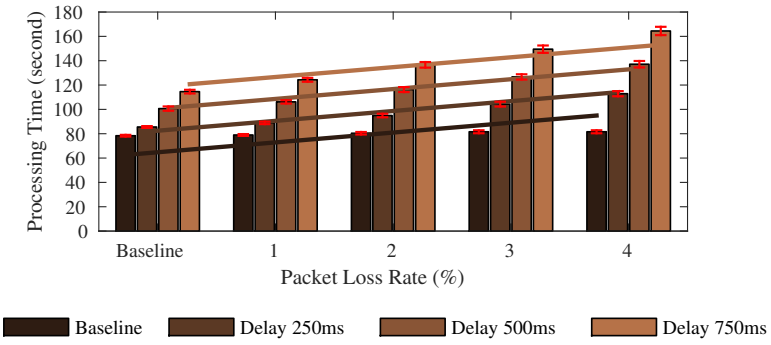


Figure 2.7: Impact of Combined Delay and Packet Loss on the Total Duration of the Collaborative Task

ences of $\Delta t_{\text{total}}^{\text{vm1}}$ and $\Delta t_{\text{total}}^{\text{vm2}}$ are negligible. Therefore we focus our discussion on obtained values for $\Delta t_{\text{total}}^{\text{vm1}}$ depicted in Figure 2.7.

In Figure 2.7, the y-axis shows $\Delta t_{\text{total}} = \Delta t_{\text{total}}^{\text{vm1}}$. The x-axis shows the different packet loss values from baseline to 4%, different delay values are shows as grouped bars, including the 95% confidence intervals for each measurement setting. We show an examples of a more general linear regression parameterized for the measurement parameters as lines colored according to the specific delay.

For the baseline packet loss and the considered delay values we observer that Δt_{total} increases, as discussed in the previous sections. We also see, that Δt_{total} is almost independent of the packet loss as long as the delay is small, i.e., at the baseline. This is intuitive, because in this case retransmission of lost packets can be considered as almost instantaneous and does not affect the transmission at all. However, in case of larger delays, the impact of packet loss starts to increase, as retransmissions take longer and consequently the time until information is successfully transmitted between server and client increases, as well.

This can be modeled using a linear regression with a CoD value of 0.943 as

$$\Delta t_{\text{total}} = 62.247 + 0.077962 \cdot x + 809.54 \cdot y,$$

for a delay x and a packet loss of y. Comparing the predicted results of the model with our measurements, we observe that while fluctuations up to 10% occur at the bounds of our parameter set, the results are of sufficient quality to be used in general cases.

These results show that while packet loss alone has no significant impact on application performance, a combination of both packet loss and delay can negatively impact application behavior. In real world scenarios, especially if WiFi or cellular access is concerned, both network parameters can be degraded noticeably. However, results from Sections 2.2.2 and 2.2.3 show, that the impact can be mitigated by using asynchronous communication patterns between client and server.

2.3 QoE Aware Placement of Cloud-based Photo Service in Edge Networks

As introduced in Section 2.1.2, the edge network photo album cloud service (EPC) uses HTTP or HTTPs over TCP to deliver stored photos. Thus, the photo loading time is influenced primarily by file size, distance between server and client, and network QoS. If the user has to wait too long to view or upload a photo, the user may stop using the service. In order to achieve a high satisfaction with the photo album service, the challenge is to efficiently place the its content to an appropriate geographical location to gain a high QoE perceived by the user.

In this section, we propose a mapping function from content size, distance and different network QoS parameters (i.e., link capacity, delay, packet loss) to the QoE of photo loading time. This can be used to decide the placement of content.

To derive this mapping function, we conduct a study with several steps which are depicted in Figure 2.8.

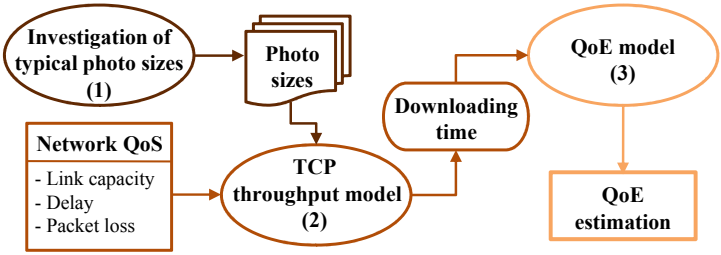


Figure 2.8: Measurement Workflows

First, we investigate the properties of cloud-based photo content through a well-known photo album service (i.e., Google Photos). In this step (1), we measure the sizes of downloaded photos at different screen resolutions. The range of average photo sizes is then selected for QoS measurements of file downloading to validate a TCP model in the next step. In the second step (2), we use a TCP throughput model proposed in [45] to calculate the downloading time of different photo sizes at various QoS parameters. We validate this TCP model in a local testbed, where different network parameters can be configured. In the last step (3), we formulate a mapping function to calculate the MOS value from a QoE model adding the output of the TCP model. Our mapping function allows determining QoE for photo loading time, depending on photo size, location, and network parameters. This helps to investigate the trade-off between the size of the photo and its placement in the cloud or edge network to achieve a high QoE for photo loading time.

The remainder of this section is structured as follows. Section 2.3.1 presents the QoS model and the testbed setup for the validation. Then, the QoE model and the discussion of the placement of content are described in Section 2.3.2.

2.3.1 QoS Model and File Downloading Measurements

In this section, we first describe the TCP throughput model used for our evaluation with the input parameters in order to figure out the relationship between network QoS and photo QoE. Thereafter, we describe the testbed setup for the measurements which are used to validate the accuracy of the TCP model.

TCP Throughput Estimation Model

Despite the fact that the TCP CUBIC is currently implemented in Linux operating systems, most of TCP CUBIC throughput models are complex analytical models for special purposes, e.g., in the context of wireless environments [46], or for multiple TCP connections [47]. As the focus of this paper is not to provide accurate results but rather to present the methodology and to conduct a qualitative study, we employ a simpler TCP Reno throughput model proposed by Padhye et al. [45]. This model has an intuitive throughput calculation and fits well to the available parameters in our measurement scenario. Note that the methodology presented here can nevertheless be applied to the recent, more accurate TCP CUBIC models.

In [45], TCP throughput is computed as follows

$$T_p \approx \min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p\left(1+32p^2\right)}\right). \tag{2.1}$$

T_p is the estimated TCP throughput, W_{max} is maximum TCP window size, $W_{max} = 64\,\mathrm{KBytes}$, p is packet loss rate, b is the number of packets that are acknowledged by an received ACK, typically b is 2. RTT is the round trip time, T_0 is the retransmission time out. To achieve the objective of the study, we calculate the RTT parameter in more detail. In fact, RTT is affected by link capacity and additional delay in network. It is the sum of transmission, propagation, and

additional delay. Thus, RTT is calculated as

$$RTT = \frac{bL}{C} + d + D_{pg}, \tag{2.2}$$

where L is average packet length, C is available bandwidth of the link, d is additional delay and D_{pg} is propagation delay. In [48], Balej et al. proposed a geographic distance estimation based on round trip time, where the propagation delay is calculated as

$$D_{pg} = \frac{2s}{c \cdot r}. \tag{2.3}$$

where s is geographic distance between server and client, r is parameter of the velocity of signal propagation, $r = 0.335$. c denotes speed of light in vacuum. The propagation delay calculated by Equation (2.3) can give us the hint about the placement of content in the cloud.

Testbed Setup and Methodology

To validate the TCP throughput model, we measure the TCP throughput of file downloads in a testbed. The results show the behavior of TCP throughput under the impact of different network parameters. First, we specify the range of file sizes for the download measurements by investigating a real web-based photo album. We choose Google Photos as an example of a well-known cloud photo album. By manually uploading and browsing a photo at the different screen resolutions, we summarize the properties of rescaled photos in our tests in Table 2.3. The photo uploaded to Google Photos is taken from a typical digital camera. It has 5184×3456 pixels in resolution and 5711 KBytes in size. Table 2.3 shows that the resolution as well as the size of original photo is rescaled at the different screen resolutions. This adaptation is also explained in [49] and [50]. From this result, we select the range of file sizes corresponded with the rescaled photo sizes, which are 128 , 256 , 512 , and 1024 KBytes.

To measure the TCP throughput of file downloads, we setup a testbed which is schematically depicted in Figure 2.9. It consists of three PCs and one server

Table 2.3: Rescaled Photos at Different Screen Resolutions

Screen Resolution	Photo Resolution	Size (KByte)
1920 x 1200	1658 x 1105	472
1680 x 1050	1433 x 956	372
1440 x 900	1208 x 805	276
1280 x 800	1058 x 705	218

running Ubuntu 12.04 LTS. The given files are transferred from Server to Client via the server running NetEm [51]. This network emulator server can adjust available bandwidth, delay, and packet loss of the connection. We use a separated Control PC to manage the testbed via SSH protocol. To transfer files from Server to Client, we use the Linux `netcat` command. We use `tcpdump` to capture the packets. The TCP throughput is then calculated by the total length and duration of packets. For the later evaluation, we emulate the different network QoS on NetEm. These parameters are the typical network characteristics of the Internet that are documented in [52] and [53] as well. The link capacity is also limited to evaluate the impact of available bandwidth on the TCP throughput.

Table 2.4 specifies the different network parameters we emulate on NetEm. The *Baseline* round trip time is measured in the testbed without any configuration on the NetEm server and we observe an average round trip time of $0.4\,\mathrm{ms}$ over 1000 packets.

Table 2.4: Emulated Network Parameters in the Measurement

Network QoS	Parameters
Available Bandwidth (KByte per second)	128; 256; 512; 1024
Round Trip Time (millisecond)	Baseline; 250; 500; 750; 1000
Packet Loss (%)	0; 2.5; 5; 7.5; 10

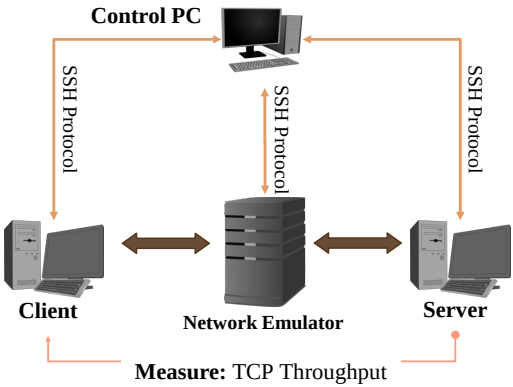


Figure 2.9: Overview of Testbed Setup for File Download Measurements

Validation

In this section we present the comparison of the values generated by the Equation (2.1) and the results obtained from the measurements. To calculate the TCP throughput from the equation, we execute all available network parameters presented in Table 2.4. Additionally, RTT is calculated in Equation (2.2) with $D_{pg} \approx 0$ due to the short distance between Server and Client in the testbed. T_0 is the TCP retransmission timeout defined in RFC document [54] and it is usually estimated by RTT and its variation. However, we observe a negligible round trip time variation in the testbed, therefore $T_0 \approx RTT$. The packet size is averaged through the `tcpdump` trace, given by $L = 2557\,\mathrm{KBytes}$. We observe that the behavior of TCP throughput is mostly similar for different file sizes. Hence, we only show the measured TCP throughput of the file $512\,\mathrm{KBytes}$ as an example in the following graphical results.

In all figures, the TCP throughput in $\mathrm{KByte/s}$ is depicted on the y-axis. The solid lines and the pluses represent the TCP throughput obtained from the Equa-

tion (2.1) and the measurements, respectively. The error bars on the pluses show 95 % confidence intervals over 30 runs. To validate the discrepancy between the model and the measurements, we calculate the relative error as

$$\frac{(|x_m - x|)}{x_m} * 100 \quad (\%), \tag{2.4}$$

where x_m and x are the throughput values calculated from the TCP model and obtained from the measurements, respectively.

Figure 2.10 shows the impact of delay and packet loss on the TCP throughput. The file is transferred at link capacity $C = 3750$ KByte/s to avoid bottleneck at both sender and receiver. In the figure, the x-axis indicates different packet loss rates ranged from 0 to 10 %. The different colors of the solid lines and the pluses represent the TCP throughput under the impact of specific network delay combined with packet loss. For sake of readabilities the y-axis is cropped to 130 KByte/s, but the maximum actual value is 233.90 KByte/s. As displayed on the figure, we observe that the results from the model and the measurements agree with each other.

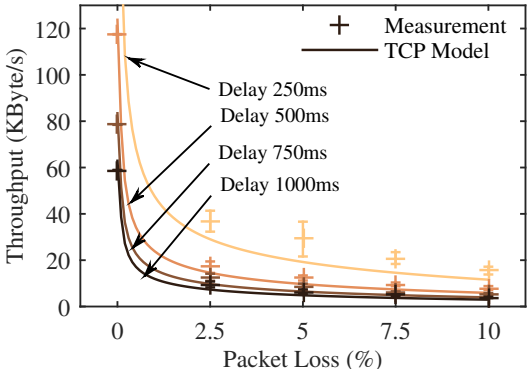


Figure 2.10: Impact of Delay and Packet Loss on TCP Throughput

Figure 2.11 shows the impact of available link capacity C and packet loss on TCP throughput. The x-axis indicates the different packet loss rates. The darker lines and pluses depict the TCP throughput calculated and measured at lower link capacities, respectively. The graph shows that there are small errors between the results calculated from the model and measured from the tests.

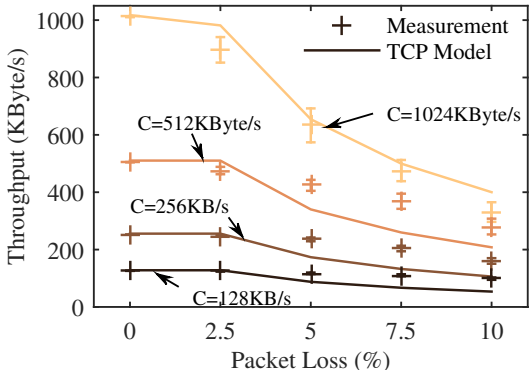


Figure 2.11: Impact of Available Bandwidth and Packet Loss on TCP Throughput

Next, we investigate the impact of the path Bandwidth Delay Product (BDP) described in [55] on the TCP throughput without the presence of packet loss. The RTT is therefore recalculated as

$$RTT = \frac{W_{max}}{C} + d + D_{pg}. \tag{2.5}$$

The results from the measurements and the Equation (2.1) are compared in Figure 2.12. The x-axis shows the different delay values ranged from *Baseline* to 1000 ms. The lines and the pluses with different colors represent TCP throughput at various link capacities C. From the figure, the TCP throughput calculated from the model and obtained from the measurements are proximately close to each other. We observe a majority of the results have errors less than 40 % cal-

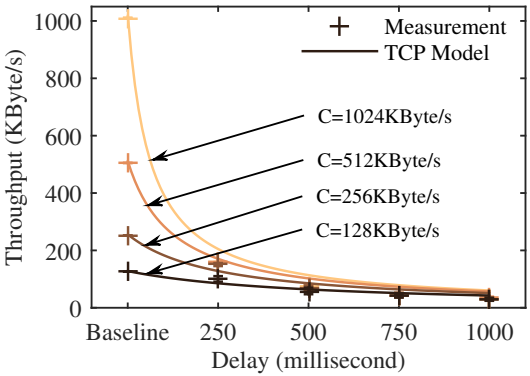


Figure 2.12: Impact of BDP Path on TCP Throughput

culated by Equation (2.4).

Figure 2.13 shows the Cumulative Distribution Function (CDF) of relative errors between the results from the TCP model and the measurements, where the x-axis indicates the error rates, the different lines shows the relative errors of different experiments. From this figure, we observe that there are approximately $60\,\%$ of the measurements values have errors less than $30\,\%$ compared to the TCP model. To conclude this section, we believe that the TCP throughput Equation (2.1) with the RTT calculated in Equation (2.2) and Equation (2.5) has sufficient reliability to be deployed in general measurements.

2.3.2 QoE Model and the Placement of Content

In this section, we describe a QoE study for photo loading time. The QoE is estimated as a function of MOS given by the duration of loading a photo. Meanwhile, the downloading time of a photo with a given photo size and network QoS can be calculated by the TCP model described in Section 2.3.1. Therefore, this time factor plays a role as a bridge in order to connect the TCP throughput

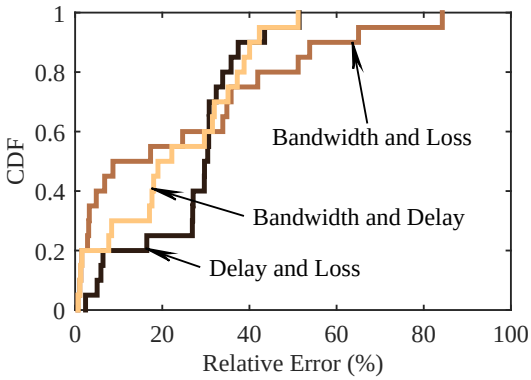


Figure 2.13: The CDF of Relative Error Between TCP Model and Measurements

model with the QoE model. The remaining of this section presents our discussion about the placement of content with regard to the user satisfaction.

QoE Estimation Model

In [14], Egger et al. contribute a study of waiting times in the context of interactive applications. They examined the QoE for several web applications including web browsing, email processing, VoIP, as well as video streaming. The authors conclude that the user perceived QoE for web-based services has a logarithmic decrease along with the increase in waiting time. In addition, the logarithmic behavior of QoE regarding to the time factor is also reported in [26] and [56]. Regarding the waiting times in the context of browsing photos, in another work [15], Egger et al. proposed a logarithmic fitting function to describe specifically the relationship between picture loading time and the user perceived QoE as follows

$$QoE\,(t) = -0.80 \ln (t) + 3.77, \tag{2.6}$$

where $QoE\,(t)$ is the function of MOS given by the picture loading time t. The authors measured the goodness of fit by calculating the coefficient of determination r^2 which has value of 1.00 in this case. The verification of the model can be found in [15]. Despite the fact that QoE model has been widely studied (e.g. in [11] [26] [56]), we choose model (2.6) as the mapping function due to its high reliability and it fits well to our measurements where photo loading time t can be calculated by the TCP model.

The Placement of Content

We present in this subsection a trade-off between the photo size, its geographical placement, and network QoS in which the user perceived QoE can be estimated from these parameters. Indeed, from model (2.1), we calculate the duration of loading a photo as

$$t = \frac{size}{T_p\,(C, rtt_s, p)} \quad , \tag{2.7}$$

where T_p is the TCP throughput estimated by monitoring the network QoS with C, rtt_s, p are link capacity, round trip time, and packet loss, respectively. Round trip time rtt_s is estimated according to Equation (2.2) and Equation (2.3) with the distance s between server and client. The $size$ is given size of a photo. From Equation (2.6) and Equation (2.7), the estimated QoE model based on network parameters, photo size, and distance is formulated as

$$QoE\,(size, s) = -0.80 \ln\left(\frac{size}{T_p\,(C, rtt_s, p)}\right) + 3.77. \tag{2.8}$$

Equation (2.8) can completely compute at which photo size and level of network QoS to gain an acceptable QoE. Figure 2.14 shows an example of the estimated QoE for loading a photo under the impact of network delay and packet loss. The x-axis indicates the packet loss rates. The y-axis shows the estimated MOS values which represent the user perceived QoE. The MOS can take the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent. The darker lines

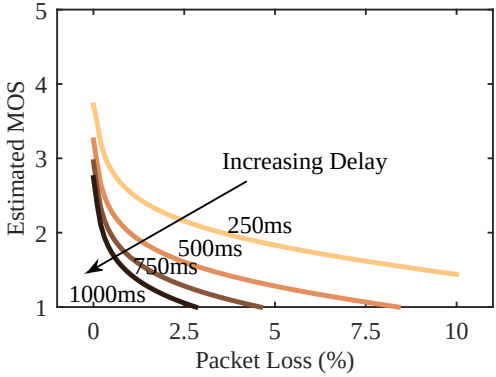


Figure 2.14: The Impact of Delay and Packet Loss on QoE for Photo Loading $C = 3750KByte/s, size = 218KBytes$ *in Table 2.3,* $D_{pg} \approx 0$ *in testbed*

depict the QoE behavior at higher delay. As shown in Figure 2.14, when the packet loss is not present in the network, the QoE for photo loading is better at smaller delay. However, the MOS value decreases dramatically with the increase of delay and packet loss. This is due to the retransmission of lost packets take longer and consequently, the time until information is successfully transmitted between server and client increases, which results in a rapid drop of MOS values as indicated in Equation (2.6).

From the equations (2.2), (2.3), (2.5), and (2.8), the trade-off between the size of photo and its geographical placement can be estimated. Figure 2.15 shows the relationship between the distance and the user perceived QoE represented by MOS values. We assume that the photos taken from Table 2.3 are transferred on a typical ADSL link, which has downstream rate of $8\,\mathrm{Mbit/s}$ following the ITU-T G.992.1 standard. Packet loss is assumed not to occur on the link, the round trip time is calculated by Equation (2.3) and (2.5). In the figure, the x-axis shows the various distances between server and client in kilometer, the y-axis indicates the corresponding estimated MOS values which represent the user perceived QoE.

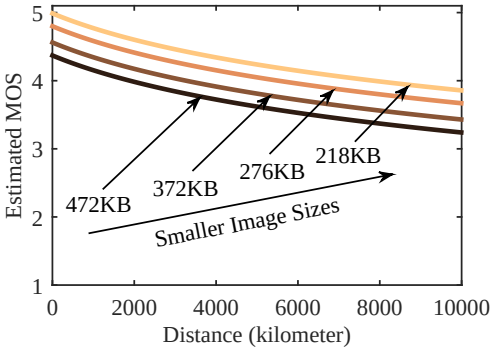


Figure 2.15: Estimated QoE for Photo Loading at Different Distances

The darker lines depict the QoE behavior of larger photo sizes.

The figure shows that the MOS values decline gradually at every longer distance and the smaller photo sizes (i.e., smaller photo resolutions as shown in Table 2.3) gain better QoE. We observe that the QoE for loading a photo $472\,\mathrm{KBytes}$ with $1658 \times 1105\,\mathrm{pixels}$ in resolution is acceptable if the distance between server and client is shorter than $4000\,\mathrm{kilometers}$. Besides, the photo has $218\,\mathrm{KBytes}$ in size with $1058 \times 705\,\mathrm{pixels}$ in resolution still gains a good QoE even it is transferred through a long distance. However, the packet loss may occur on the link and the probability of occurrence might be higher at longer distance. In this case, the MOS values will decrease rapidly as shown in Figure 2.14. To solve this problem, the service providers can rescale the photos resolution or reduce the photos size to meet the QoE as indicated in Equation (2.8). After all, if both adjusting photos quality and improving the network QoS do not meet the user satisfaction, a migration of the user photo album to the edge server next to the geographical location of the user is recommended.

2.4 Lesson Learned

Cloud services and SaaS products gained considerable interest recently as a replacement for traditional centralized infrastructure and locally installed software products. Although the Internet users can benefit enormously from the cloud services and the SaaS paradigm, the challenge is how to achieve a high user perceived QoE for these Internet-based applications.

This chapter focuses on the influence of different network conditions and content placement on the QoE of the two prominent cloud applications, Google Docs and photo album. In multi-tenancy architecture, Google Docs service is a more shared model where users are able to collaborate editing a document in a joint space. Despite the fact that a shared database reduces cost, the centralized architecture of this model affects the users if the service is down. In addition to this, users at different locations have different access network conditions. A low network QoS can therefore significantly influence the user expectation. Meanwhile, cloud-based photo album is a more isolated model where each user can have his/her own photo album. Even though the cost of ownership is higher, the content of the album can be dynamically migrated to edge network next to the user if the loading time of photos is high due to a long distance. By doing this, the user experiences a better provisioned service, so the higher QoE. To this end, is it important for the network providers to be aware of QoE for this service regarding the location of the user.

When studying both cloud applications and doing measurements in the testbed, we draw two major findings as follows.

Firstly, to assess the impact of network impairment on the performance of Google Docs, we consider two scenarios, which are derived from common Google Docs use cases. In the single scenario, we considered a single user editing a document. In the collaborative use case, with one user editing the document and the second user observing the changes. To objectively quantify the impact of network delay and packet loss, we measure the time it takes to complete the whole process as well as certain parts of it, e.g., the login or the creation of

the document. The measurements were performed using a local testbed which is connected to the Internet via a NetEm network emulator. The user interactions were automated using Selenium. Our results show that in both scenarios, packet loss below 4% does not influence the duration of sub-processes, if there is no network delay. In contrast to this, network delay negatively influences the performance of Google Docs, even in the absence of packet loss. Hereby, the login process as well as the process of creating a new document are the most delay-sensitive subprocesses. Furthermore, we also analyzed the impact of combined delay and packet loss. Here the results show a significant degradation of the Google Docs performance even for small values of delay and packet loss, if both occur at the same time. This outcome can help to shed a first light of the behavior of Google Docs as an exemplary SaaS solution under varying network conditions from an objective point of new. This in turn can later be used to evaluate the impact of this application behavior on the QoE perceived by the user. Here especially the obtained linear regression models can be used in analytical models for optimization and trade-off analysis network resources, energy consumption and QoE.

Secondly, while considering the cloud-based photo album service, we found out that the placement of content is one of the key factors that affects the user satisfaction. A long distance access is characterized by a high delay and possible packet loss which results in a longer data loading time. Thus, the user perceived QoE for the service is dramatically dropped. To increase the performance of services, the placement of content must be considered. The closer the content to the user geographically is, the faster it will be delivered to the user that will also increase the user perceived QoE. To achieve this perception, we propose in this chapter a trade-off between the size of photo, its placement, and network QoS to acquire a high QoE for photo loading time in a particular usage of a cloud-based photo album service. We first validate a TCP throughput model and use it to calculate the photo loading time from a given photo size and network QoS. Thereafter, we map a QoE logarithmic function to the TCP throughput model. From this mapping function, we can estimate QoE for photo

loading time from a given photo size, its placement, and network QoS. Our results show that, with the presence of packet loss smaller than $2.5\,\%$, the MOS value decreases dramatically at the delay larger than $250\,\mathrm{ms}$, even with a small photo that has only $218\,\mathrm{KBytes}$ in size. However, without packet loss, the QoE for photo loading time only decreases gradually at every longer distance. The QoE level is still acceptable when a big photo that has $472\,\mathrm{KBytes}$ in size and $1658 \times 1105\,\mathrm{pixels}$ in resolution, is transfered over the distance from 6000 to $8000\,\mathrm{kilometers}$. From the results, we can achieve a good QoE of photo loading time by optionally adjusting the size of photo, improving network QoS, or moving the content next to the user. Our contribution may help cloud service providers to have another method to estimate the behavior of QoE for photo loading time based on various parameters.

3 VNF-based QoE Monitoring in the Cloud

While the previous chapter focused on QoE assessment and QoE-awareness for content placement of different cloud applications, this chapter is concerned with one layer below, the network layer. In this chapter, QoE monitoring in network for HTTP Adaptive Video Streaming (HAS) is done as a prerequisite for QoE management. Although the rapid growth of video streaming provides the video producers with a great opportunity to increase their revenues, it is presented the challenges for the network operators in the aspect of managing the high volume of video traffic and a large number of subscribers. As users expect a good service, they may stop watching the video if there are interruptions during playback and consequently, the QoE perceived by the users for the video service drops. Therefore, to ensure the user experience, the network providers must be aware of the video quality that prevails on the user device as well as the source of video quality degradation. On the one hand, this not only helps the network providers to be aware of the QoE for the video service, but also gives them the ability to improve their service. On the other hand, it is the mandatory prerequisite to perform QoE management in the network.

In order to assess the quality of video streaming on the end user side, a video provider can utilize feedback given by the video player. A network operator, however, needs his own monitoring mechanism in the network to estimate the actual video quality at the end user device. To monitor the video quality in the network, video flows must be analyzed at packet level. This can be achieved by using the Network Function Virtualization (NFV) paradigm [20]. The basic idea

of NFV is to separate software from its underlying physical hardware. By using virtualization technology, the Virtual Network Function (VNF) can be consolidated on commodity servers. This helps the providers to quickly deploy a service on multiple hardware platforms. In addition, VNFs can be instantiated, operated, and/or migrated automatically on one or more commodity servers in an NFV architecture without having to install new hardware [57–59]. These benefits provide us with a solution for the development and implementation of a VNF for video monitoring at different locations in the network apart from using a dedicated physical device.

Despite of the advantages of applying the NFV paradigm for the video quality monitoring, this approach also faces several problems. Firstly, since the VNF for video monitoring might be placed at different Point of Presence (PoP) in the network, the video quality measured at these locations may result in different degrees of accuracy. It is primarily due to a long distance to the user among the others. Secondly, with the tremendous growth of smart devices [60], more and more clients are using mobile network to browse the videos. As a result, the mobile access network may also influence the accuracy of video quality estimation. On the other hand, to be aware of the user satisfaction with the video service, QoE for video streaming must be measured accurately. Since the QoE is estimated mainly by considering stalling frequency and length extracted from the video monitoring process [16, 17, 61–63], these key influence factors might be overestimated due to the performance of the VNF, the impact of different VNF placements and the mobile network as well. Thus, a new study is required to investigate how the VNF placement and the mobile network influencing the accuracy of the video quality and the QoE estimation.

To tackle these problems, in this chapter we conduct a study consisting of two stages. First, we design a VNF to analyze video flows in the network by using deep packet inspection and an algorithm to estimate the video quality based on video header parameters. In this stage, we set up a dedicated local testbed with a client, a network emulator, and a middle-box where the VNF is installed. This testbed allows us to examine the functional operation of the VNF and the

influence of different network conditions on the performance of video quality and QoE estimation.

In the second stage, to improve the reliability of our approach, we investigate the feasibility of deploying a VNF for video buffer and QoE monitoring in the Amazon Web Service (AWS) cloud. The reason of using cloud infrastructure for the VNF monitoring is that it is possible to provide the monitoring on demand and in a reactive way by instantiating the function on a data center server. In addition to simple instantiation, efficient use in the cloud brings improved scalability and cost savings. In this stage, we study the influence of different PoPs and a high mobility environment on the accuracy of the VNF for monitoring the video quality and QoE perceived by the user. Our findings show that the accuracy of the VNF for video buffer monitoring decreases with the distance of the PoP to the client. This is not only due to the delay and bottleneck between the monitoring point and the client, but also due to the mobile access network. This means, with increasing distance of the PoP to the client in a mobile environment, the probability of detecting stalling events also decreases, which is a key factor to evaluate the QoE for video streaming.

The contribution of this chapter is threefold. First, we propose a VNF-based video quality and QoE monitoring that can be deployed in the network. In addition to this, we investigate the performance of the VNF under different scenarios and network conditions. Second, we propose a cloud-based NFV architecture on the example of deploying the QoE monitoring VNF on top of the AWS cloud. In this approach, we study the influence of different VNF placements in the cloud and a high mobility access network on the accuracy of the VNF for monitoring the video buffer and the QoE. Lastly, we validate our findings with experiments in a real scenario with a typical moving user in a vehicle.

The content of this chapter is mainly taken from [4]. The remainder of it is structured as follows. Section 3.1 introduces background of the study and related work. Subsequently, in Section 3.2 we present the first study where we evaluate the impact of different network QoS and VNF placements on the accuracy of video quality and QoE estimation. In Section 3.3, we extend our study to a real

deployment of the VNF in the AWS cloud. Here, we investigate the impact of different PoP and a high mobility access network on the performance of the VNF. Finally, Section 3.4 concludes this chapter with lesson learned.

3.1 Background and Related Work

This section is divided into five subsections. In Section 3.1.1 we introduce HAS and the technology behind. Subsequently, we present several QoE assessment and QoE monitoring methods in Section 3.1.2 to 3.1.4. Finally, we summarize several typical works about the NFV and its combination with cloud infrastructure in Section 3.1.5.

3.1.1 HTTP Adaptive Video Streaming

HAS was developed to overcome the traditional inefficient streaming technology such as real-time streaming protocol or progressive download. The main idea of HAS is decomposing a video into one or more consecutive non-overlapping periods that are seamlessly streamed to the client web browser over HTTP [64]. These periods are so-called video segments or chunks that are described in a manifest file named Media Presentation Description (MPD). By streaming separated video chunks to the client, the streaming server can adapt the quality of each video chunk to the network condition.

The MPD is transmitted to the client after a video request via HTTP. This file contains information about all video chunks, such as their lengths, resolutions or frame bit rates. Before the client playing back the video, at least the first video chunk has to be fully downloaded, while downloads of subsequent chunks can still be ongoing. All the downloaded chunks are stored in the client video buffer. These buffered data are consumed during the video playback and filled up when a new chunk is fully downloaded. If the video buffer is empty during playback, an interruption of the video playback occurs. This interruption is called stalling event. The authors in [16, 17, 61] state that the stalling frequency and length are

the main influence factors on the QoE for HAS. The other QoE influence factors, from technical perspectives might be network QoS, video adaptation rate or segment size, to the user perceptual like initial delay or image quality [65].

3.1.2 QoE Assessment Methodologies

As defined in [12], "Quality of Experience (QoE) is the degree of delight or annoyance of the user of an application or service". The definition is possibly suitable in the context of multimedia applications or services. In this context, QoE is the level of user satisfaction or enjoyment with an application or a service. Being able to assess QoE helps network providers to react on the network quality degradations. The following subsections introduce subjective and objective assessment methods for QoE.

Subjective QoE Assessment

Subjective QoE is the degree of expectation or satisfaction with a service experienced by human [66–68]. In HAS, subjective QoE is the feeling or perception of a viewer (or a "subject") with the quality of image, stalling frequency, or stalling length. Subjective QoE assessments are psychophysical experiments where participants are asked to designate their opinion on the quality of a specific application or service (e.g., a video) through a given set of stimuli. The most popular method to quantify subjective QoE is to use Mean Opinion Score (MOS) defined in [69]. The MOS can take the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent, which represent the corresponding degree of user satisfaction with the service. Subjective QoE assessment is considered to have high reliability results, since the tests are implemented in real scenarios with recruited participants. This method, however is expensive and time consuming. In addition, the set of stimuli must be well-defined and test sessions also require organization efforts that increases the cost of tests. In [70], Hossfeld et al. introduce an alternative to subjective testing, called QoE crowdtesting. This method leverages the advantage of crowdsourcing [71] by submitting assessment tasks to a global

worker pool through a web-based aggregator platform such as Microworkers[1] or Amazon Mechanical Turk[2]. The forefront advantage of this method is reducing cost. This is achieved from a virtual laboratory with a low capital expenditure for a single test and limited number of participants. Thanks to a large diversity and high availability of participants, a test can be done or repeated quickly in one or different aggregator platforms. However, one of the major disadvantages of crowdtesting is unreliability of the user rating. This is due to the fact that cheat users may only try to maximize their payments with minimum effort. Additionally, the test conditions and environments of the workers are unknown in most of the cases. Thus, the impact of test conditions on the result is different from worker to worker.

Objective QoE Assessment

Objective QoE assessing refers to an attempt to predict the user behavior based on analytical and/or statistical models [18]. Similar to traditional subjective QoE assessment, this method also outputs the quantitative result reflecting the user expectation and satisfaction with an application or a service under test. The derived QoE model is especially useful for service quality monitoring in the network where providers can easily estimate the QoE perceived by the end user from validated input parameters (e.g., [3, 14–16, 72]). However, in contrast to the ease of use, objective QoE can only provide estimated results with a specific correlation with perceptual quality measured by subjective assessment.

In [73], the authors categorize objective assessment methodologies for IPTV into five types of models. These models are media-layer, parametric packet-layer, parametric planning, bit stream layer models, and hybrid. Each type of model can exploit the input parameters for QoE models at different network layers. For instance, the media-layer model uses video signals to estimate QoE, while the bit stream layer model captures the packet header and payload to derive input parameters for a QoE model. In this chapter, we use the combination

[1] `https://microworkers.com`
[2] `https://www.mturk.com`

of the parametric packet-layer model and the bit stream layer model in the form of a VNF to monitor video quality and QoE for HAS.

3.1.3 QoE Monitoring Methodologies

QoE monitoring is typically based on software solutions to collect the information of how a user experiencing a particular application or service. In network management, network providers can use different QoE monitoring methods to assess the user expectation to accordingly react and improve the service. In the following, we introduce several QoE monitoring methods at the user, application, and network layers.

QoE Monitoring at User Layer

Monitoring QoE for a service on the user layer refers to statistical information collected actively or passively from the user while using the service. Active QoE monitoring at the user layer can either be done by dedicated subjective user studies or by customer feedback. QoE monitoring which utilizes subjective assessment methods can be implemented in laboratories, field studies, or using crowdsourcing as described above. A popular active QoE monitoring method is listening to customer feedback by integrating surveys into the service. Such quality feedback integrator is widely used in speech and video services where the user is asked to rate the quality of a conversation after they hang up the phone (e.g., a Skype call). This method is considered to be lower cost and easy to deploy for different applications and services. However, the feedback or rating might be less accurate, since the user may be annoyed by receiving the feedback dialog at each time he uses the service. In addition, the reasons for a good or bad rating is hard to be traced. Nevertheless, the combination of the active monitoring with other methods may increase the accuracy of QoE monitoring.

In contrast to active monitoring, passive QoE monitoring at the user layer does not interact directly with the customers, but probes their behavior passively through different measurement techniques. For example, providers can

investigate the ratio of user engagement, the volume of sale products or the number of hotline complaint calls, etc., to estimate the degree of the user satisfaction with the service. This monitoring method is completely transparent with the customer activities. Moreover, some existing information of the customers can also be obtained from business administration departments which can be analyzed to estimate the customer behavior with offered services. However, QoE estimated by this method has limited accuracy since the user engagement or sales volume are not only influenced by the customer but also by other objective causes. The reasons for a high or low QoE is hard to be traced as well.

QoE Monitoring at Application Layer

QoE monitoring at application layer can be done either inside a service or at the end user device. *In-service QoE monitoring* is ordinarily implemented by service providers who can monitor QoE influence factors within the service. These influence factors are served as input parameters for a QoE model. Afterward, QoE scores are signaled back to the provider to adapt the service in order to meet the user expectation. *End-device QoE monitoring* relies on a monitoring function installed at the end user device. This additional software is used to collect the performance indicators of a service or application which can be translated into QoE scores. The monitored information might be useful for the user or can be signaled to network operators. Based on this, traffic management can be applied to ensure a high QoE level perceived by the user. However, the drawback of this approach is high complexity and cost to deploy such a solution at the user end device.

QoE Monitoring at Network Layer

At the network layer, QoE monitoring can be categorized into active and passive methods, in which an active monitoring mechanism uses probe nodes to perform measurements. Whereas, in passive monitoring, network traffic passing through a measuring point is captured and analyzed.

To monitor QoE actively for a specific service or application at network layer, one or multiple probe nodes are distributed across end points in the network. These nodes send extra data traffic to server and measure the service quality based on pre-defined QoE metrics. The QoE scores are then signaled to a collector server or directly to providers to adjust service in order to improve user experience. Active QoE monitoring mechanisms depend on the service. In [74, 75], the authors present a method for monitoring the quality of video streaming using a Pseudo Subjective Quality Assessment (PSQA) technology. This active monitoring method uses probe nodes distributed in the network. At first, the authors capture a relation between the parameters that cause video quality degradations (i.e., I, B, P video frames loss rate) and user perceived quality using PSQA technique. Thereafter, several experiments are done in a testbed where simulated probe nodes periodically send statistical information about frame loss rate and mean loss burst size to a data collector server. The QoE for a video delivery network is estimated at run time using trained data from PSQA technique. The accuracy of the platform is determined by the loss rate of video frames and mean size of loss bursts. This method is limited in accuracy and efficiency. This depends on the fact that the number of probe nodes is finite and that these nodes produce more overhead into the network or might change the outgoing data traffic for probing purposes.

Passive QoE monitoring approach can use *Deep Packet Inspection (DPI)* technique to capture data traffic passing through a measuring point in the network. Payload data is then parsed for service performance indicators. In HAS, payload data contains video request time, segment download time, MPD file, which can be used to estimate video buffer and other video quality influence factors. Passive QoE monitoring using DPI technique does not change the data traffic or require any end device. This method is promising to be highly accurate, since QoE is measured directly from service key influence factors extracted from payload data. A softwarized monitoring function can be deployed at any PoP in the network. However, to be able to measure QoE, one must thoroughly understand the service and its communication in the network. In addition, it is more difficult

to monitor QoE for encrypted data traffic. In this situation, a *service flow classification* method can be used. This method can leverage machine learning to classify flows of a service to monitor network layer parameters such as packet size, packet inter-arrival time, or throughput. This network layer parameters can be used to directly estimate QoE or to estimate application layer parameters. Thereafter, a QoE model is used to map application layer parameters to QoE. An example of *service flow classification* is presented in [76].

3.1.4 QoE Monitoring for HTTP Adaptive Video Streaming

The authors in [62, 63, 77] introduce QoE monitoring for YouTube video using end device application, namely YoMoApp. This application layer monitoring method is promising to have high accuracy in QoE measuring, since it can extract video quality parameters directly from the video player on the user device. This method however only work at the client side, while our approach is deployed in the network.

Concerning video monitoring based on packet analyzing, in [78], Wamser et al. model the YouTube stack at three levels, which are transport, application, and user. These models may help service providers to understand the functionality of YouTube from controlling video data flows to user perceived QoE. Our study is also similar to studies in [79–81] to some extent, where the authors analyze the YouTube video flows in the network using DPI to estimate the QoE based on extracted packet traces. Nevertheless, the authors focus more on QoE monitoring rather than deployment location. In our study, we however concern to the accuracy of the VNF for video monitoring depending on different placements in the cloud.

In [82–85], the authors introduce methods and present the results of evaluating the quality of HAS in mobile environments. The authors observe that most video streams in mobility scenarios will gain a smooth playback if the adaptation process is improved. In our work, we also deploy a real scenario of moving client, however, we focus more on the accuracy of VNF monitoring in the cloud.

3.1.5 NFV Cloud Infrastructure for VNF-based QoE Monitoring

In the NFV paradigm, network functions are virtualized and deployed on top of the virtualization layer which is also provided by existing cloud management systems like Amazon Web Service (AWS). In [86], Oechsner et al. present an algorithm for deploying VNF on an existing cloud infrastructure. The algorithm is designed based on adapted zone concept and aims to optimize resilience and performance of the VNF in the context of OpenStack cloud. In [87], Carella et al. propose a possible deployment of IP Multimedia Subsystem (IMS) software on top of the OpenStack cloud. The idea of leveraging a telco cloud environment to manage Service Function (SF) is presented in [88]. The authors introduce the Cloud4NFV platform that is built on cloud, SDN, and WAN technologies to manage SFs as a service. This platform is promising to improve the management of SFs in the cloud. However, the use of open source platforms like OpenStack or OpenDaylight for the virtual infrastructure management plane will need more assessment to ensure the security and feasibility in practice.

In [89], Yu et al. introduce the network function-enabled cloud computing (NeFuCloud). This platform was proposed with the idea of separating the control plane from the data plane underlying cloud infrastructure which is the principle of Software-Defined Networking (SDN). The authors believe that NeFuCloud, with separated control plane will gain better performance and flexible management. However, the control plane might be centralized or distributed across the network. A centralized control plane may lead to a bottleneck when the network is scaled out and a distributed control plane increases the control overhead. In our concept, we propose to build an NFV on top of an existing cloud platform (i.e., AWS) that provides flexible management and implementation.

In an NFV architecture, the deployment location is one of the most important VNF configurations. Where to place the VNF to meet the requirements of resources, performance, network efficiency is an emerging topic in research community. In [90], Clayman et al. contribute a placement engine software installed

in the orchestration layer of an NFV architecture. This algorithm decides, where to place the virtual routers to meet an adaptive resource utilization. The placement problem of virtual Deep Packet Inspection (vDPI) functions is presented in [91] for SDN and in [92] for NFV, respectively. In SDN, the authors use genetic algorithms to solve the placement problem, while in NFV, they use integer linear program with cost constraints. The authors conclude that an appropriate placement of vDPI depends on functional targets, operation cost, and the number of instances as well.

3.2 Impact of Network QoS on the Accuracy of QoE Estimation for HAS

In this section, we examine the accuracy of the monitoring function depending on its placement in the network. We use a predefined mapping function to estimate the video QoE based on the number of stalling events calculated from the function. To this end, we carry out a study in several steps. First, we design a VNF as a plain software that exploits a Python library, namely Scapy[3] to capture the video flows at the network interface. Then, we use an algorithm to estimate the video buffer and detect the stalling events based on timestamps of streamed packets. In the second step, we set up a local testbed with two scenarios to assess the impact of the function placement on its accuracy. In both scenarios, we validate the accuracy of the function by comparing its estimate to the actual video quality that is monitored at the client web browser. The result is the accuracy of the estimation depending on the placement of the VNF in the network.

The remainder of this section is structured as follows. In Section 3.2.1, we first present our research methodology, estimation algorithm, and measurement setup for different scenarios. Thereafter, the impact of bandwidth and packet reordering on the accuracy of video buffer and QoE estimation is discussed in Section 3.2.2 and Section 3.2.3, respectively.

[3] http://www.secdev.org/projects/scapy/

3.2.1 Methodology and Measurement Setup

In this subsection, we first highlight the research methodology of the study. Then, we present the description of algorithm that is used to estimate the video buffer and stalling events from extracted network layer information. Finally, we describe the measurement setup for different scenarios.

Methodology

To monitor QoE for video streaming in the network, we use a dedicated testbed including a network emulator and a middle-box to compose two scenarios, namely, the Edge Server (ES) scenario and the Data Center (DC) scenario, depicted in Figure 3.1.

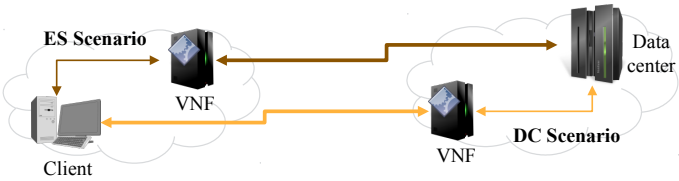


Figure 3.1: Overview of Scenarios

From the figure, video content is assumed to be stored in a data center to the right. In the ES scenario, our function is deployed on an edge server to monitor all the video flows passing through from the streaming data center to the client. The edge server is an essential instance of the edge computing paradigm, which refers to the enabling technologies allowing computation to be performed at the edge of the network close to the user [93]. In addition, edge servers are also the migration target when a service provider has to move the video content to the edge in order to improve quality. In the DC scenario, we deploy the monitoring function near to the data center. In this scenario, the function can utilize a high data rate traffic from data center network. However, it is placed far away from

the client, which may cause some performance problems due to the degradation of network conditions on the path to the client.

In both scenarios, we deploy the same VNF monitoring. The VNF is installed on the middle-box to sniff the video flows and outputs necessary video information to feed the estimation algorithm. This sniffing function exploits a Python library, namely Scapy. This open source software provides real time packet sniffing and decoding. In fact, we can parse the video flows on the fly. In our measurements however, we do the analyzing task afterwards to avoid the interference with the performance of the sniffing task.

We assume that it is a long distance between the server and the edge with various network segments and routers. The degradation of the network can be considered as the combination of high round trip time and congestion. We therefore shape the video traffic by using a network emulator (NetEm) [51]. This Linux-based software can adjust different network parameters to evaluate the impact of network QoS on the service or application in general. To validate the video quality estimated by the function, we compare it to actual video quality simultaneously obtained from the client. This actual video quality is measured by using a Javascript-based web API. The discrepancy between the estimated and actual video buffer shows the level of accuracy of the VNF monitoring.

Video Estimation Algorithm

The video estimation algorithm is designed for the evaluation of performance and accuracy of video streaming with unencrypted HAS traffic. Although there are other techniques available for encrypted traffic, we focus on the investigation of the accuracy of such an VNF-based monitoring and the side-effects of network QoS and VNF placement in the network.

In [17], Seufert et al. describe the QoE influencing factors of HAS, which are initial delay, stalling, quality adaptation among others. To achieve the goal of monitoring QoE for the video, we design an algorithm to estimate these parameters based on the extracted packets of the video flows. To feed the algorithm, the sniffing task of the VNF outputs all necessary information such as packet re-

ceiving time, source and destination IP addresses, source and destination ports, TCP acknowledgment numbers, TCP sequence numbers as well as the length of packet and HTTP payload. Figure 3.2 schematically depicts the algorithm of estimating video buffer based on analyzing video flows.

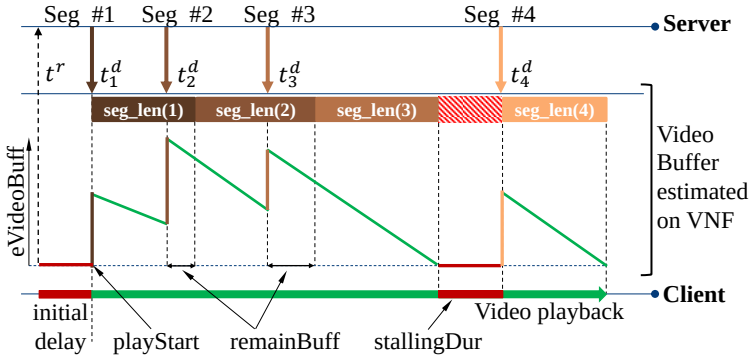


Figure 3.2: Graphical View of Estimating Video Buffer Algorithm

The top of the figure shows an example of transferring four video segments in sequence. The video flows are detected by matching pre-defined keywords contained in HTTP payload. After a video request from the client has been sent, the streaming server provides a Media Presentation Description (MPD) which contains video segment name and the respective durations. This information is stored for further analysis. Based on the MPD, the client seamlessly requests segments in sequence. The downstream packets that compose one segment are grouped and analyzed based on their common TCP acknowledgment number. Thereby, the timestamps of fully downloaded segments are recorded.

From the left side of the figure, t^r is the time when the client requests the video, t_i^d is the time when segment i is fully downloaded, which is represented by each vertical arrow. The solid rectangles with faded colors depict the corre-

sponding segment length `seg_len`(i) extracted from the MPD. The solid zig-zag line represents the amount of estimated video buffer `eVideoBuff` over playback time. It drops steadily over video playback and is filled up when a new segment is successfully downloaded. The `initial_delay` is the amount of time measured from the client requesting the video at the time t^r until the video starts to play out at the time `playStart`, while `playStart` $\approx t_1^d$. In fact, through measurements on the client, we observe that the client starts the playback almost instantaneously after the first segment is fully downloaded. The `remainBuff` is the amount of video time remaining in the buffer before filling up with the next segment. If the next segment arrives later than expected, then an interruption of the video playback occurs. This is called the video stalling event. In [16], the authors state that the longer stalling occurs, the lower QoE perceived by the user. In the Algorithm 3.1, the `stallingDur` is the length of a stalling event which serves as an input parameter for QoE estimation. The simplified algorithm is presented in the following.

In the algorithm, the `samplingRate` variable is defined at first. Sampling rate is the amount of time in milliseconds between two sampled video buffer values. Since our algorithm estimates the video buffer based on segment arrival time, to avoid missing a buffer value, the sampling rate must be smaller than the download time of a segment. Through a local experiment and based on analysis at packet level presented in Section 3.3.4, we observe that the average download time of a segment is about $350\,\mathrm{ms}$. Based on this analysis, we choose a sampling rate of $100\,\mathrm{ms}$ at the VNF and the client to ensure all buffer values are captured. Next, the algorithm estimates `playStart` with the time when the first segment arrives. The `avlPlayBackTime` is the available playback time which is built up by the duration of segments obtained from MPD, named `segmentLength`. This amount of time is cumulatively summed by every new segment that is fully downloaded. To estimate the video buffer, the amount of video that has been played out must be calculated first. `playOut` is calculated by the difference of the time between a new segment arrived and the `playStart` subtracting with `stallingTotal`. The video buffer `remainBuff` is calculated by the difference

Algorithm 3.1 Estimating Video Buffer Algorithm

Define `samplingRate` *and the time when the video starts the playback and to sample video buffer*
`timestamps` $=$ `playStart` $= t_1^d$
Available playback time is built from first segment
`avlPlayBackTime` $=$ `segmentLength(1)`
for *Next segment* **do**
    Amount of video time has been played out when segment i *comes*
    `playOut` $= t_i^d -$ `playStart` $-$ `stallingTotal`
    Remaining video buffer before next segment arrives
    `remainBuff` $=$ `avlPlayBackTime` $-$ `playOut`
    if `remainBuff` $<= 0$ **then**
        Record duration of one stalling event
        `stallingDur`$(i) = t_i^d - t_{s(i-1)}^{end}$
        `stallingTotal` $+=$ `stallingDur`(i)
        Record stalling states with pre-defined sampling rate
    else
        Amount of video time has been played out when segment $i-1$ *comes*
        `ite` $=$ `hasBeenPlayedOut`
        Estimate video buffer and timestamps cumulatively from the time t_{i-1}^d
        while `ite` $<=$ `playOut` **do**
            Timestamps of sampling video buffer
            `timestamps` $+=$ `samplingRate`
            Video buffer state is sampled every amount of `samplingRate`
            `eVideoBuff` $=$ `avlPlayBackTime` $-$ `ite`
            `ite` $+=$ `samplingRate`
            Save one record [`timestamps`, `eVideoBuff`]
        end while
    end if
    `hasBeenPlayedOut` $=$ `playOut`
    Updated available playback time
    `avlPlayBackTime` $+=$ `segmentLength(i)`
    $t_{s(i)}^{end} = t_i^d +$ `segmentLength(i)`
end for

of `avlPlayBackTime` and the amount of video which has been played out. This parameter plays a role as a trigger to detect whether a stalling event occurs. If the video buffer is depleted, we record one stalling event and its length. Then, we sample the stalling state with the corresponding variable `timestamps`. Conversely, if there remains video in the buffer, we record the video buffer values and its timestamps to the variable `eVideoBuff` and the `timestamps`, respectively. The variable `timestamps` is increased until it reaches the last millisecond of the video.

Measurement Setup

The measurement setup for the ES scenario is schematically depicted in Figure 3.3, and consists of one *NetEm* server and three Personal Computers (PCs). One PC is used to install the VNF monitoring named *VNF Moni*, another PC is used for the *Client* that browses the videos via a testbed network. Additionally, we use a *Control PC* that is connected remotely to the testbed via a dedicated control network to avoid interference with the experiment. The *NetEm* is running on a SUN FIRE X4150 server. This server uses Ubuntu 12.04 LTS while other PCs use Ubuntu 14.04 LTS as operating system. The testbed is connected to the Internet via a research network. To achieve high reliability in timestamps calculation, in both scenarios we synchronize the clocks of *Client* and *VNF Moni*

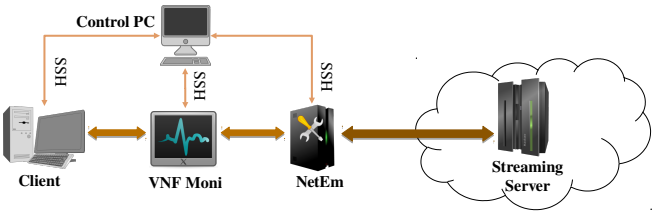


Figure 3.3: Overview of the Testbed in Edge Server Scenario

machines to the same NTP server. For the DC scenario, we use all the same devices, only changing the positions of the *VNF Moni* and the *NetEm*. Thus, the *VNF Moni* is placed near the data center as shown in Figure 3.4.

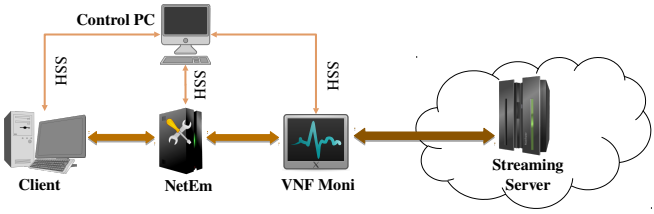
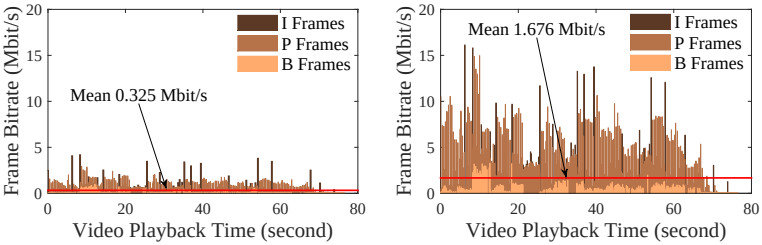


Figure 3.4: Overview of the Testbed in Data Center Scenario

At the implementation, a Python measurement script is used to start the monitoring function at the *VNF Moni*. Then we use the Selenium Webdriver[4] to automatically browse the video on the *Client* after adjusting the pre-defined link capacity parameters on the *NetEm*. This ensures the network to be configured before the client starts the video playback.



(a) 240P *(b) 720P*

Figure 3.5: Frame Bit Rate vs. Playback Time

[4]http://www.seleniumhq.org

We choose *metacafe.com*[5] as video streaming source, since they provide unencrypted HAS service. We implement measurements on 10 different videos with various types of content and lengths, offering the same three levels of resolution, i.e., 240 p (428 × 240), 360 p (570 × 320), and 720 p (1280 × 720). Figure 3.5 shows the frame bit rate over playback time of a typical video[6] at 240 p and 720 p. The figure shows that at the lowest quality of 240 p, the video has mean bit rate of 0.325 Mbit/s. This means if the available link capacity is below this number, stalling event might occur during the video playback. Since we want to investigate the accuracy of the function in all possible behaviors of video playback. We shape traffic on the *NetEm* at three levels that may cause some possible influences on the video playback as shown in Table 3.1.

Table 3.1: Shaped Link Capacities on NetEm

Link Capacity	Possible impacts on video playback
512 Kbit/s	Stalling occurs sporadically
1 Mbit/s	No stalling with fluctuated video quality and buffer
10 Mbit/s	No stalling with highest video quality and buffer

3.2.2 Impact of Bandwidth on the Accuracy of Video Buffer and QoE Estimation

Based on the methodology and the testbed setup presented in Section 3.2.1, we have implemented several measurements between September and October 2016 at the University of Würzburg. To secure the stability of the monitoring function, we tested with 10 different videos and 5 replications each. We observed that the performance of the function was similar with different videos. Therefore, in the following we present the graphical results of the video with frame bit rate as shown in Figure 3.5. This video has a total length of 75 s. To increase

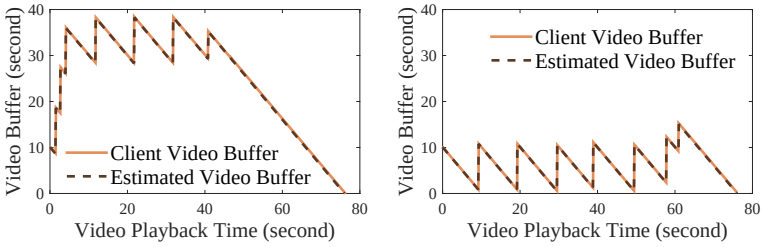
[5] http://metacafe.com
[6] http://www.metacafe.com/watch/11419883/homeless-video-mp4

statistical significance of the measurements for this video, we produced 30 replications at each link capacity configuration that is shown in Table 3.1. After the experiment, we collected 180 log files given by the sniffing task of the function in both scenarios and a corresponding number of video buffer sampling logs on the client. To minimize missing the state of video buffer, we did sampling on the client every 100 ms as described in Section 3.2.1. Using the Algorithm 3.1, we conducted the estimation task with all extracted log files. To compare the estimated and actual video buffer, we use the timestamps of the first video segment extracted from sniffing logs as the time reference for all graphs that have video playback time x-axis.

The Behavior of Video Buffer Estimation in Different Scenarios

Figure 3.6 and Figure 3.7 show the behavior of video buffer at different link capacities in the ES and DC scenarios. In all sub-figures, the x-axes indicate the video playback time while the y-axes show the video buffer. The dashed lines depict the estimated video buffer provided by the function and the solid lines show the actual video buffer extracted from the client browser.

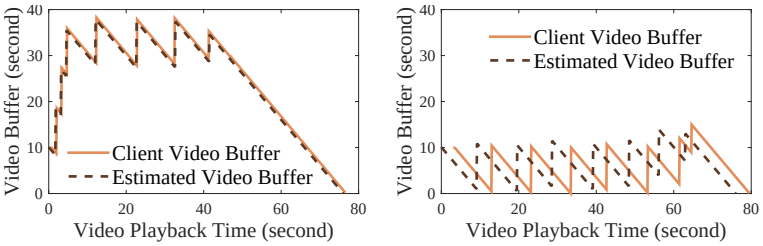
From the figures, it is clear that the video buffer is smaller at lower bandwidth. Considering our estimation function in comparison with the actual video buffer



(a) Link capacity of 10 Mbit/s *(b) Link capacity of 512 Kbit/s*

Figure 3.6: Video Buffer over Playback Time in Edge Server Scenario

extracted from the client browser. Figure 3.6 shows that our function has high accuracy in estimating the video buffer as well as detecting the stalling events in the ES scenario. Due to the placement of the function close to the client, the difference in arrival time of video flows is negligible at both machines, even at a low bandwidth of 512 Kbit/s.



(a) Link capacity of 10 Mbit/s *(b) Link capacity of 512 Kbit/s*

Figure 3.7: Video Buffer over Playback Time in Data Center Scenario

Similarly, Figure 3.7a also shows a good estimation at high bandwidth of 10 Mbit/s in the DC scenario. Although the function is placed far from the client, it can still estimate approximately the buffer of video playing on the client. This is because the bandwidth in this scenario is high, such that the packet arrival time at both machines is almost the same, which provides us a good estimation with small error. However, Figure 3.7b shows a bad fit in the DC scenario for a bandwidth of 512 Kbit/s. As the function is located near the streaming server, it can utilize a high data rate and receive packets of the video segments shortly after the client requests them. Due to the bandwidth limitation of the link after the VNF, the reception of these packets at the client is delayed. This induces the estimation error since the function calculates the video buffer based on the timestamps of downloaded segments.

The Accuracy of Video Buffer Estimation

To evaluate the difference between the estimated and actual video buffer at both scenarios, we calculate the Root Mean Square Error (RMSE) as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{3.1}$$

where n is the number of video buffer samples, $\hat{y}_i$ and y_i are the video buffer values, sampled by the VNF and client browser as a baseline, respectively. Figure 3.8 shows the Cumulative Distribution Function (CDF) of the RMSE between the estimated and actual video buffer at different bandwidth in two scenarios. The figure points out that $90\,\%$ of estimated samples in ES scenario have errors less than $1\,\mathrm{s}$ compared to baseline values.

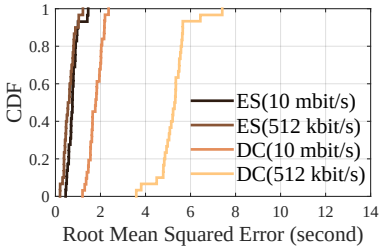


Figure 3.8: RMSE Between Estimated and Actual Video Buffer

In the DC scenario, the accuracy of the function is acceptable as there are $80\,\%$ of estimated samples have an error less than $2\,\mathrm{s}$ at the bandwidth of $10\,\mathrm{Mbit/s}$. This demonstrates that at high link capacity, our function can still approximately estimate the buffer of the video played at the client. Conversely, at lower link capacity of $512\,\mathrm{Kbit/s}$, the function estimates the video buffer with a high error where most of the estimated values have an error larger than $4\,\mathrm{s}$.

Regarding the measurement results of the other videos, Table 3.2 shows the

mean errors with a $95\,\%$ confidence interval between the estimated and actual buffer of the other videos with various lengths. However, we only present the results of measurements at the bandwidth of $512\,\mathrm{Kbit/s}$ in this table. This is because at higher bandwidth, the mean errors are negligible similar to our conclusion above. The table indicates that through different videos, our function can still estimate the videos buffer with high accuracy in the ES scenario. In the DC scenario, the mean errors are higher and various over different videos due to the differences in the number of segments and frame bit rates among themselves.

Table 3.2: Mean Error of Video Buffer at Bandwidth of $512\,\mathrm{Kbit/s}$

Video ID	Length	ES Scenario	DC Scenario
11419867	201 s	$0.84\,\mathrm{s} \pm 0.033$	$2.81\,\mathrm{s} \pm 0.051$
11420085	126 s	$0.65\,\mathrm{s} \pm 0.039$	$2.94\,\mathrm{s} \pm 0.059$
11419885	95 s	$0.46\,\mathrm{s} \pm 0.037$	$2.54\,\mathrm{s} \pm 0.071$
11419888	71 s	$0.57\,\mathrm{s} \pm 0.041$	$2.21\,\mathrm{s} \pm 0.085$
11419938	126 s	$0.52\,\mathrm{s} \pm 0.035$	$2.78\,\mathrm{s} \pm 0.065$
11420011	67 s	$1.13\,\mathrm{s} \pm 0.068$	$2.57\,\mathrm{s} \pm 0.078$
11420017	158 s	$0.54\,\mathrm{s} \pm 0.029$	$4.47\,\mathrm{s} \pm 0.047$
11420019	305 s	$0.99\,\mathrm{s} \pm 0.036$	$2.94\,\mathrm{s} \pm 0.058$
11420073	202 s	$0.52\,\mathrm{s} \pm 0.029$	$3.12\,\mathrm{s} \pm 0.051$

In [16], Hoßfeld et al. contribute a study of QoE for YouTube video using subjective crowdtest. They investigated the impact of stalling parameters (i.e., frequency and length) on the user perceived QoE. The authors proposed exponential fitting functions to quantify the QoE impact of stalling as

$$f_1(N) = 3.26 \cdot e^{-0.37 \cdot N} + 1.65, \tag{3.2}$$

$$f_3(N) = 2.99 \cdot e^{-0.96 \cdot N} + 2.01, \tag{3.3}$$

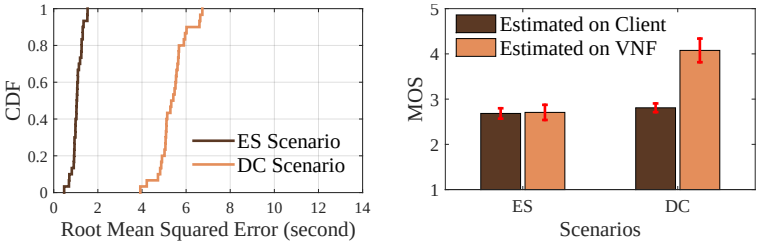
where $f_1(N)$ and $f_3(N)$ are the functions of MOS given by the number of stalling events N with stalling length of $1\,\mathrm{s}$ and $3\,\mathrm{s}$, respectively. It can be seen

from the equations that the MOS only depends on the number of stalling events and length. In previous measurements, we observe that there is only constant time shifting in estimated video buffer. Thus, the monitoring function can estimate almost the same number of stalling events and length that occur at the client. As a result, the VNF is able to approximately estimate the QoE in both scenarios. Indeed, by using Equation (3.2) to calculate MOS values based on number of stalling events, we observe that there is negligible mean error between estimated and actual MOS in both scenarios. Specifically, in the ES scenario, the mean error between estimated and actual MOS values is $0.53\,\%$, while in the DC scenario is $8.52\,\%$. Herein, the MOS is averaged over 30 replications with a $95\,\%$ confidence interval.

3.2.3 Impact of Packet Re-Ordering on the Accuracy of QoE Monitoring for HAS

In the following, we consider another scenario where the network condition is unstable, i.e., a scenario with packet reordering. In this scenario, packets belonging to one video segment may arrive at the client out of order. Packet reordering is also reported in several studies. In [94], Leung et al. describe five major causes of packet reordering, including packet level multipath routing, route fluttering, inherent parallelism in modern high-speed routers, link-layer retransmissions, and router forwarding lulls. In [95], Gao et al. argue that the packet reordering occurs with probabilities usually more than $30\,\%$ in concurrent multipath transfer system. The network heterogeneity and the use of multiple links in wireless networks also causes packet reordering which is described in [96].

To investigate the impact of packet reordering on the VNF monitoring for QoE, we have done several measurements with the same testbed. Specifically, in both scenarios we configure the NetEm to immediately dequeue $25\,\%$ of the packets, the others are delayed by $500\,\mathrm{ms}$. There is a correlation of $50\,\%$ influencing on the next packet and the link is set to $512\,\mathrm{Kbit/s}$. Figure 3.9a shows the CDF of the RMSE between the estimated and actual video buffer. It can be



(a) CDF of RMSE *(b) QoE Estimation*

Figure 3.9: The Accuracy of VNF at 512 Kbit/s with Packet Reordering

seen that the error is similar to our previous measurements shown in Figure 3.8.

However, in these measurements, the function additionally fails to correctly estimate the number of stalling events. Figure 3.9b depicts the QoE estimation based on the number of stalling events. The x-axis shows the measurements in two scenarios, the y-axis indicates the estimated MOS values calculated by Equation (3.2). The MOS can take the following values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent. The darker bars describe MOS values calculated from the actual number of stalling events counted at the client, while the other MOS values are estimated by the function. The MOS values are averaged over 30 replications with a 95 % confidence interval. The figure shows that the MOS calculated in the ES scenario and on the client are smaller than 3. While in the DC scenario, the function estimates a smaller number of stalling events, which is represented by a high average MOS value of 4.07, and higher than actual value by 31.11 %. It is due to the fact, that in the DC scenario packet reordering only happens in the user access network. The result is that some packets arriving at the client not in order are buffered in the memory to wait for reassembly. Additionally, the bottleneck of the link also produces more delay to packet arrival time. As a consequence, the video segment is decoded at the client later than the one arrives at the VNF, since packets arriving at the VNF even faster to fill up the video buffer to avoid an occurrence of stalling. Conversely, in the ES scenario, both the client

and the VNF are influenced by the same packet reordering configuration. Thus, they have similar behavior of the QoE. It can be concluded that, in the case of packet reordering occurred in the user access network, the function does not work properly if it is placed far away from the client.

3.3 Study on the Accuracy of VNF-based QoE Monitoring in the Cloud

The previous insights show that the VNF-based monitoring for HAS can be deployed in the middle-box to estimate video buffer and QoE. The results show that the accuracy of the estimation depending on the placement of the VNF in the network. Nevertheless, the VNF is installed in a middle-box in a controlled testbed that still has limitation in reliability. In fact, in the NFV paradigm, VNFs are required to be automatically deployed in a scalable and flexible ways with the availability of network appliance multi-version and multi-tenancy [20].

To overcome the limitation of testbed-based VNF and to find the best solution for our approach, in this section, we investigate the feasibility of placing a VNF-based video buffer monitoring in the cloud. First of all, we propose an architectural design of the deployment of VNF monitoring in the AWS cloud infrastructure. Then, we describe the monitoring process in the form of a VNF. We provide results that show the feasibility of monitoring application layer parameters within video traffic with DPI technique. The monitored parameters have a high correlation with the user-perceived QoE. We compare the performance of VNF monitoring at different PoP in the AWS cloud with regard to the actual quality obtained from the client device. Next, we add an investigation of the estimation accuracy where the user is streaming a video in a high mobility scenario within a mobile network.

The rest of this section is structured as follows. First, in Section 3.3.1 we introduce the architectural design for VNF monitoring in the cloud. Then, in Section 3.3.2 and Section 3.3.3 we describe the research methodology and measure-

ment setup with different VNF placements in the AWS cloud, respectively. Subsequently, in Section 3.3.4 and Section 3.3.5, we present the measurement results of the video buffer and QoE estimation in two scenarios of the VNF deployment. Lastly, in Section 3.3.6, we describe the behavior of video buffer estimation in a high mobility environment in a real measurement scenario.

3.3.1 Architecture for VNF QoE Monitoring in the Cloud

In this section, we propose the architecture of QoE monitoring in the cloud that utilizes an existing cloud infrastructure to deploy VNFs on the top. This architecture is used to implement our QoE monitoring measurements described in Section 3.3.3. Figure 3.10 gives an overview of the architecture.

The design can be divided into two main blocks, the cloud infrastructure plane and the NFV management plane. This architecture is aligned with the ETSI-NFV architecture framework [97]. However, VNFs, Management and Orchestration (MANO) are built on top of an existing cloud infrastructure.

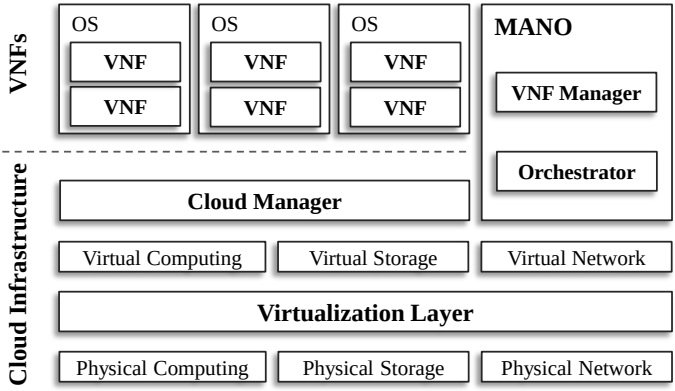


Figure 3.10: Overview of VNF QoE Monitoring Architecture

NFV Management Plane

The NFV management plane provides the environment and management for deploying VNFs. Aligning with the ETSI architecture, it consists of MANO and VNF instances.

VNF Instance: In the NFV management plane, one or more VNFs are installed in a virtual machine or a container (e.g., Docker [98]) and are managed by the MANO. Each block that consists of a virtual machine with operating system (OS) built-in and one or more VNFs installed, is called a *VNF Instance*. An instance is allocated with sufficient resources at run time and can be dynamically scaled-up or out by an orchestrator. In our measurement setup, the VNF for video quality monitoring is installed on a virtual machine. This VNF instance is initiated at different working regions of the AWS cloud.

MANO: In this architecture, the MANO is more lightweight compared to the ETSI-MANO due to the handover of infrastructure management to a cloud manager. It consists of VNF manager and orchestrator. Aligning with the ETSI standardization, the *VNF Manager* is in charge of VNFs life cycle management. The *Orchestrator* is responsible for automated provisioning necessary resources and network for the VNFs. The MANO can also work with the cloud manager via a REST API to initiate a new instance of VNFs from templates. By doing this, the VNFs can be scaled out or migrated across different working regions. In this work, we only present functional overview of the architecture and leave implementation for future work.

Cloud Infrastructure Plane

The underlying cloud infrastructure plane provides all necessary hardware and software resources to build a virtual environment where VNFs are deployed.

Virtualization Layer: Virtual resources consist of virtual computing, storage, and network which are created by a virtualization layer. This layer ensures the isolation of virtual machines from the underlying physical resources. Thus, many virtual machines or containers can share limited physical resources using

a resources sharing scheme.

Cloud Manager: Since the NFV infrastructure is no longer exist in this architecture, this module is built to manage the underlying cloud infrastructure and dynamically allocate virtual resources for VNF instances. For example, a new VNF instance will be allocated with initial virtual resources like computation, storage, memory, and networking. In addition, a specific geographical region under consideration and other security policies of the instance are also configured before launching. Then, the instance is initiated to host a virtual machine or a container (e.g., Docker). The instance can be automatically scaled-up with additional resources without any interruption. It also can be duplicated and migrated to another working region.

In this work, we choose AWS as the cloud infrastructure for QoE monitoring VNF, since AWS provides an Infrastructure as a Service (IaaS) platform with multiple management options. AWS is a collection of different online services and a subsidiary of Amazon.com, Inc [99]. The most popular services are Elastic Compute Cloud (EC2)[7] and Simple Storage Service (S3)[8]. S3 is mainly used to store data in AWS data centers. While, EC2 provides computing nodes on demand, where user can launch instances with different hardware configurations depending on his requirements. Instances are multiple virtualized containers that can be installed on the same physical machine. In addition, AWS provides a Virtual Private Cloud (VPC) with security groups for each user, this ensure the isolation of the containers from the other users.

When launching an instance, the user can select a desired operating system and a region where the instance is operating. AWS provides EC2 services located in 16 regions that are totally independent from each other. In each regions, there are availability zones that are also isolated and connected through low-latency connections. In this work, we choose EC2 service to deploy our VNF for video quality and QoE monitoring at two different regions, namely *eu-central-1b* that is located in Frankfurt, Germany and *us-west-2b* that is located in Oregon, USA.

[7] `https://aws.amazon.com/en/ec2/`
[8] `https://aws.amazon.com/en/s3/`

3.3.2 Methodology

Our study aims to monitor the quality and the QoE of HAS in the cloud. To this end, we first choose AWS EC2 as cloud environment where we place the VNF at a PoP. To capture the video traffic passing through the PoP, we reuse the VNF as described in Section 3.2.1. To analyze the video flows, we also reuse Algorithm 3.1 which provides us with an estimated video buffer based on download timestamps of video segments. To validate the estimated values, we simultaneously sample the video buffer at the client browser by using a Javascript-based web API. The discrepancy between the estimated and actual video buffer shows the level of accuracy of the VNF monitoring.

To evaluate the influence of different placements on the accuracy of the VNF in high mobility user access networks, we deploy the function at two AWS instances. One located in the central of Europe which is near the client, the other located at the West of the USA which is far away from the client. Both of these locations are close to the streaming server or its CDN server. This means the network in between the PoPs and the streaming server has a high quality. For the high mobility user access network, we divide experiments into two scenarios, *testbed* and *real scenarios*. In the *testbed scenario*, we emulate a wireless environment based on pre-defined LTE bandwidth traces measured in real life. With the network emulation, we can evaluate the behavior of the VNF running in the cloud in multiple experiments. In this scenario, we provide the results of VNF accuracy in estimating video buffer and QoE for video streaming using a reference QoE model. In the *real scenario*, we use a vehicle to drive the client machine around a city and in country side regions. The results provided in this scenario are used for the validation of our simulation.

3.3.3 Measurement Setup

In this section, we present the measurement setups of testbed and real scenarios at both local testbed and in the AWS cloud. The installed hardware, network configuration, and method of implementation are described in the following.

AWS Cloud Infrastructure Setup

As mentioned in Section 3.3.2, we divide experiments with the VNF deployed in the AWS cloud into two scenarios. In the *testbed scenario*, the user access network is emulated by a network emulator. Meanwhile, in the *real scenario*, the user access network is provided by a German telecommunication operator. Nevertheless, in both testbed and real scenarios, the VNF is installed in an AWS cloud instance. The instance acts as a PoP for the VNF where a proxy program is used to route video traffic passing through the PoP to the client. During the measurement, the location of the client is varied. In the *testbed scenario*, the client is placed in the local testbed, while during the measurements of *real scenario*, the client is placed in a vehicle moving around the city and country side. Thereby, this scenario is also used to investigate the influence of the mobile network on the accuracy of the VNF. In both scenarios, the VNF is installed in an AWS *t2.micro* server with Ubuntu Server 16.04 LTS as operating system. Each instance has 1 virtual CPU clocked up to 3.3 GHz, 1GB of RAM and low-to-medium network performance. These instances are located in different regions as mentioned in Section 3.3.2. The aim of this location distribution is to evaluate the influence of difference VNF placements on its accuracy. Table 3.3 shows the average Round Trip Time (RTT) and the network throughput in between the AWS regions and the client in the testbed.

Table 3.3: Mean RTT and Throughput of AWS Cloud Regions

AWS Region	Mean RTT	STD	Mean Throughput	STD
eu-central-1b	6 ms	0.50 ms	258 Mbit/s	14.00 Mbit/s
us-west-2b	170 ms	0.11 ms	109 Mbit/s	11.60 Mbit/s

Table 3.3 shows that the mean RTT of the client and the *eu-central-1b* region is only 6 ms and it is much smaller than the *us-west-2b* region. The mean throughput in *eu-central-1b* region also doubles the *us-west-2b* region. This is due to a longer distance between the client and the *us-west-2b* region that may

have undesired network congestion compared to a shorter distance between the *eu-central-1b* region and the client. Regarding the streaming server, all measurements are conducted with the use of the video streaming platform *metacafe.com*[9]. This platform has the benefit that it provides unencrypted traffic for HAS. The analysis of an MPD file of this video platform has been introduced in Section 3.2.1.

Measurement Setup for the Testbed Scenario

The testbed scenario measurements are conducted in a testbed located at the University of Würzburg. Figure 3.11 shows the topology of the testbed.

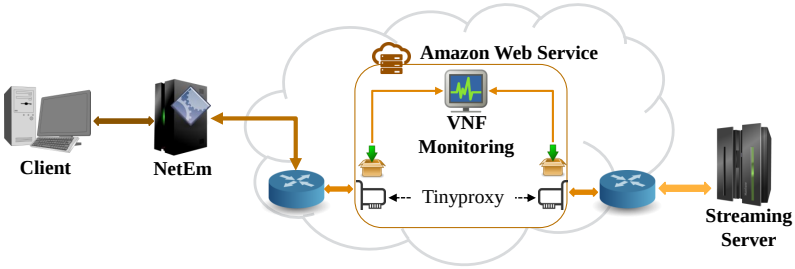


Figure 3.11: Overview of Testbed Scenario Topology

The *Client* is a Fujitsu PC with 1 Gbit/s NICs. Ubuntu Desktop 14.04 LTS is used as operating system. The *NetEm* is a SUN FIRE X4150 server with Ubuntu Server 16.04 LTS as operating system. The testbed is connected to the Internet via the German research network backbone for universities (DFN - Deutsche Forschungsnetz). For the sake of traffic tracing, we deploy the VNF monitoring at both the client and the AWS cloud. This allows us to compare the QoE estimation in the cloud and at the client device as well as to analyze the traffic at both end

[9] `http://www.metacafe.com`

points. In addition, the *Tinyproxy*[10] software is used to route video traffic passing through the PoP from the streaming server to the client. To secure the reliability in timestamps calculation, we synchronize the system clock of all testbed devices with the same NTP server.

In this scenario, the *Client* is fixed in the testbed and measurements are implemented during daytime. We simulate homogeneously a single user watching a video in every experiment replication. The interval between two replications is 60 s. To this end, we use a Python library, called Selenium Webdriver[11] for Google Chrome to automatically browse the video. In the setup, we emulate a high mobility user access network with a network emulator (NetEm) [51]. This Linux-based software can adjust different network configurations to evaluate the impact of network QoS on services or applications in general. To start the measurement, at the *Client* we use a Python script to remotely activate the VNF at the AWS instance through SSH protocol. Then, the *Client* automatically browses the video after the *NetEm* is configured with pre-defined bandwidth. This ensures the network is shaped before the client starts the video playback. The bandwidth traces were measured from a real mobile scenario where it was sampled every 1 s. The duration of the bandwidth traces is about 500 s. In the measurements, we configure the *NetEm* continuously to ensure different network condition in each replication. Video stalling events therefore only occur sporadically over all replications. The bandwidth measurements were performed in LTE networks along several routes in and around the city of Ghent, Belgium, during the period of Dec. 2015 to Feb. 2016 (more details can be found in [100] and the website below[12]). This setup allows us to produce enough experiments to evaluate the influence of wireless network on the accuracy of the cloud-based VNF as well as QoE estimation for video streaming.

[10] https://wiki.ubuntuusers.de/Tinyproxy/
[11] http://www.seleniumhq.org
[12] http://users.ugent.be/~jvdrhoof/dataset-4g

Measurement Setup for the Real Scenario

In the real scenario, we use the mobile network for Internet access. Figure 3.12 shows the network topology of this scenario.

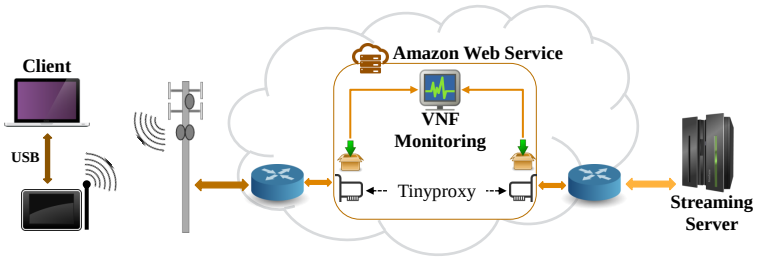
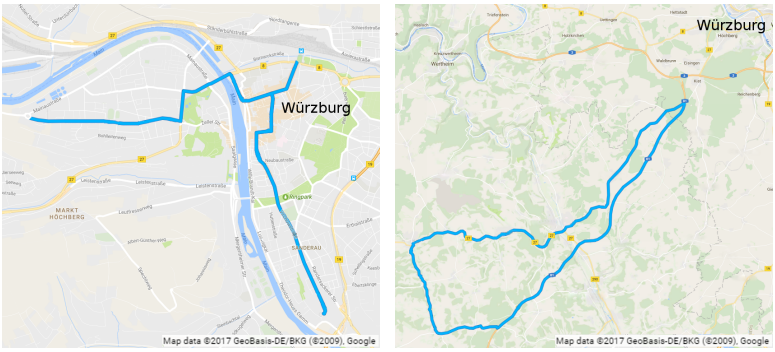


Figure 3.12: Overview of Mobility Client Scenario Topology

The *Client* is a notebook with Ubuntu 14.04 LTS as underlying operating system. The network connection is provided by a tablet via a USB hub. The tablet connects to the Internet using its built-in LTE modem which supports up to 150 Mbit/s downstream and 50 Mbit/s upstream. The network signal is provided by a major German telecommunication operator with a maximum downstream bandwidth up to 300 Mbit/s. The measurements in this scenario are implemented in two different configurations of location. Figure 3.13 shows the routes of the vehicle in both city and country side regions.

In the first configuration, as shown in Figure 3.13a, we implemented the measurements while taking the public transport system (electric tram) as our vehicle in the city center of Würzburg. In the second configuration, as shown in Figure 3.13b, we pre-defined a fixed route over country roads and the Highway A81 and we used car as the vehicle.



(a) City *(b) Countryside*

Figure 3.13: Routes of Vehicle in Mobility Scenario

3.3.4 Video Quality Monitoring in the Testbed Scenario

Based on the research method presented in Section 3.3.2 and measurement setup in Section 3.3.3, we have implemented experiments between January and February 2017 at both the University and the city of Würzburg. The video we present in the following measurement results has a typical content with total length of 200 s. The evaluations consider the accuracy of the network function, where the accuracy is measured as the difference between estimated video buffer by the VNF and actual video buffer obtained by the client browser. To minimize the number of missed video buffer states, we use a sampling rate of 100 ms at the VNF and the client as described in Section 3.2.1. Since the video buffer is measured and estimated at different devices and locations, we choose the video starting time calculated by the VNF as the time reference for all graphs that have video playback time x-axis.

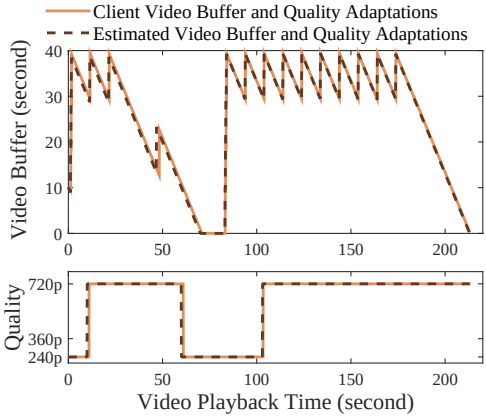
The measurement results in this section are used to evaluate the impact of VNF placement and high mobility environment on the accuracy of video buffer and QoE estimation. The client device is located in a testbed where the mobile

network environment is emulated by a network emulator as described in Section 3.3.3. The monitoring VNF is placed at different regions of the AWS cloud in Europe and USA. We also deploy a similar VNF at the client machine to capture all incoming packets from the streaming server. This VNF is referred to the *local VNF*. To increase statistical significance of the measurements, we produce 60 replications at each PoP.

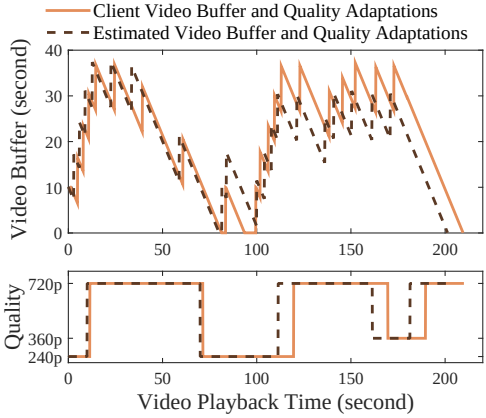
Influence of VNF Placement on Video Buffer Estimation

To intuitively compare video buffer and stalling events estimated by the VNF and obtained from the client, we select a typical video experiment that consists of different video states (i.e., high buffer, stalling, quality adaptations). Figure 3.14 shows the behavior of the video buffer and quality in the testbed scenario. In all sub-figures, the x-axes show video playback time in seconds. In the upper parts of the sub-figures, the y-axes indicate the amount of video buffered in the memory. In the lower parts of the sub-figures, the y-axes show the video quality adaptations to different network conditions. The solid lines depict the behavior of video buffer and quality sampled by the client browser, while the dashed lines represent video buffer and its quality estimated by the VNF.

The results show that in both PoPs, there is a correlation between the quality adaptations and video buffer dynamics. The client requests a low quality video segment at first, due to the unawareness of network conditions. As shown in Figure 3.14a, between the playback time of $71\,\mathrm{s}$ and $83\,\mathrm{s}$, the video is stalling. This is a result of a connection loss. The video buffer is therefore depleted before new video segment arrives. The video quality also drops, since the client requests a smaller segment size. This typical behavior of HTTP adaptive video streaming is also described in [64, 65]. Considering the accuracy of the video buffer estimation, the upper part of Figure 3.14a shows a good fit of estimated and actual video buffer as well as detecting stalling events. The VNF estimates the video quality adaptations with high accuracy as shown in the lower part of the figure. In contrast to this, the VNF placed at the US-PoP estimates the video buffer with higher error as shown in Figure 3.14b. In this case, there is a short



(a) EU-PoP



(b) US-PoP

Figure 3.14: Video Buffer and Quality Monitoring at Different VNF Placements

time shifting at the beginning of the video playback and after the stalling event. The video buffer estimation is fluctuated and less accurate.

The Accuracy of Video Buffer Estimation

As described above, there is a time shifting in the estimated video buffer and quality adaptation compared to the actual value obtained from the client. Indeed, to evaluate the discrepancy between the estimated and actual video buffer at both PoPs, we calculate the RMSE using Equation (3.1) described in Section 3.2.2. Figure 3.15 depicts the CDF of RMSE between estimated and actual video buffer. The lines with different colors represent the CDF of RMSE at different PoPs.

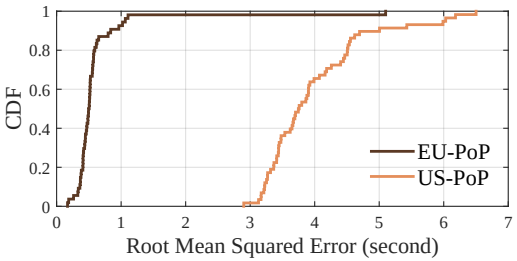


Figure 3.15: RMSE Between Estimated and Actual Video Buffer

Figure 3.15 shows that the estimated values at the US-PoP have errors larger than 3 s. The EU-PoP errors are smaller than 0.7 s in 90 % of the cases. Due to the placement of the VNF at EU-PoP near the client, the difference in arrival time of video flows is negligible at both machines. In case of packet loss, the retransmitted packets are received almost instantaneously at both machines as well. Therefore, high accuracy estimations are obtained at the EU-PoP which is geographically closer. Conversely, the longer distance in between the client and the VNF at the US-PoP may cause delay and congestion. This produces a time shifting in estimating the video buffer when the VNF is placed at this location. This means, the estimation error is highly dependent on the RTT. Thus, we be-

lieve that, the error will increase linearly at every higher RTT or longer distance or between streaming server and the user. In the next subsection, we present an analysis at the packet level to understand the cause of the estimation error.

Packet Delay Analysis

In fact, the monitoring function may add a delay caused by the forwarding and processing of the proxy. This overhead is relatively small compared to the delay caused by the distance induced by the placement of the monitoring function, which is the scope of our study. Therefore, we do not consider the overhead of the video buffer estimation process in this work. To understand the influence of different VNF placements on its accuracy, we analyze the delay of video segment arrival through a scheme as depicted in Figure 3.16.

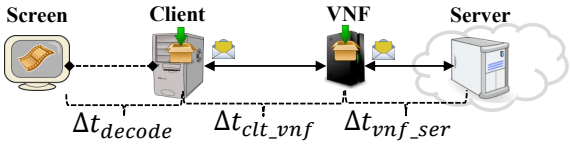


Figure 3.16: Overview of Packet Delay Scheme

Although packets traveling across the network are influenced by queuing delay or processing delay, we only consider the download time and decoding time of each video segment. In addition, we analyze the download time of video segment as measured by the VNF, since analyzing the download time of individual packet is not straight forward. Because of the TCP congestion control algorithm [101], packets passing through the proxy at the AWS PoPs may be re-encapsulated to adapt the network condition in between the client and the AWS PoPs. As shown in Figure 3.16, Δt_{decode} is the amount of time between a segment is fully captured at the client and it is filled up to the video buffer. The interval Δt_{clt_vnf} is the segment download time from the VNF to the client. Note

that, as described above, we also deploy a similar *local VNF* at the client device for traffic analysis purpose. The duration Δt_{vnf_ser} is the segment download time from the streaming server to the VNF. This network delay, however, is not considered in this study. Figure 3.17 shows the average decoding and download time calculated from measurements at different monitoring points.

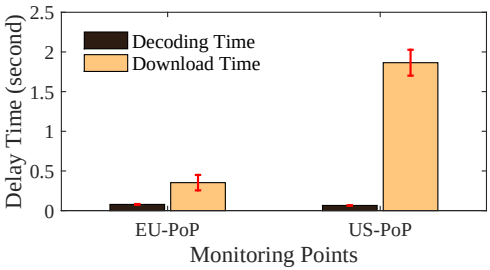


Figure 3.17: Average Decoding and Download Time at Different VNF Placements

In Figure 3.17, the x-axis shows monitoring points at the EU-PoP and the US-PoP. The y-axis indicates the delay time. The bars with different colors depicts the mean of decoding and download time over 60 replications with a $95\,\%$ confidence interval. The result shows that, the decoding duration is negligible in the measurements at both PoP. However, the average download time of video segment measured at the US-PoP is much higher than at the EU-PoP. This means, a long distance between the client and the US-PoP with additional unstable user mobile access network are the main factors influencing the accuracy of the VNF for video buffer monitoring.

3.3.5 Influence of VNF Placement on QoE Estimation

In Section 3.2.2, we employ Equation (3.2) as the reference model for evaluating the accuracy of QoE monitoring. This model is introduced by Hoßfeld et al. in [16], and has only one argument number of stalling events for a given

stalling length 1 s or 3 s. In this section, we refer to another QoE model presented in [61], where the authors combine both stalling frequency and length in one formulation to quantify QoE perceived by the user as follow

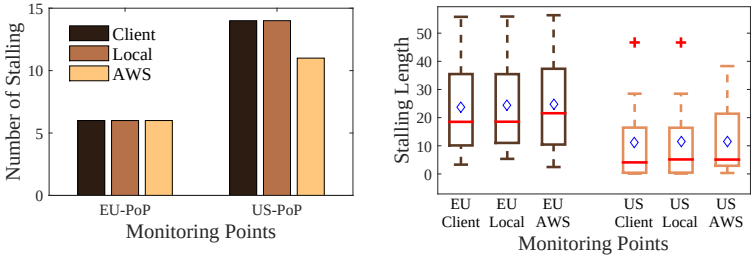
$$f_{MOS}(L, N) = 3.50 \cdot e^{-(0.15L+0.19)N} + 1.50, \tag{3.4}$$

where f_{MOS} is the function of the Mean Opinion Score (MOS) given by the average number of stalling events N and stalling length L. The equation shows that, the QoE for video streaming only considers the number of stalling events and length as main influence factors. In the following, we present experimental results that show the behavior of QoE estimation under the influence of VNF placement and mobility environment.

Stalling Frequency and Length Analysis

Figure 3.18 shows an overview of the video stalling behavior over all measurement replications at the EU- and US-PoP.

In Figure 3.18, the x-axes show the monitoring points, which are at the client, at the *local VNF* and at the VNF deployed within the AWS cloud instance. The y-axes indicate the total number of stalling events and the stalling length in



(a) Number of Stalling Events *(b) Stalling Length*

Figure 3.18: Video Stalling Behavior at Different VNF Placements

Figure 3.18a and Figure 3.18b, respectively. In Figure 3.18b, each box shows the maximum, median, and minimum durations of stalling events at a measurement point. The bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers. The outliers are plotted individually using the '+' symbol. The diamond symbol in the middle of the box shows its mean value.

The result shows that in the measurements at the EU-PoP, less stalling events occur than at the US-PoP. In fact, when the VNF is placed at the EU-PoP, the requests from the client will be sent to the CDN streaming server which is also placed in Europe. Similar to the other scenario, when we place the VNF at the US-PoP, the client will request the video at a central streaming server in the USA. Thus, this explains that we receive more stalling events at the US-PoP scenario due to a longer distance and possible network impairments in between the client and the streaming server. Considering the stalling length in the EU-PoP scenarios, due to the lower probability of stalling occurrence than at the US-PoP, the average stalling length in the EU-PoP scenario seems to be higher. In fact, in the US-PoP scenario, we also observe several stalling events with long duration. However, due to a higher number of stalling events with several short stalling lengths, the average stalling length in the US-PoP is lower. Nevertheless, the most important result is that the *local VNF* always delivers an exact estimate for both stalling frequency and length. Whereas, the VNF deployed in the AWS cloud shows estimation errors. In the next subsection, we present the QoE estimation result based on stalling frequency and length described above.

QoE Estimation

Figure 3.19 shows the QoE estimation for video streaming over the course of the measurements and at different locations. For comparison between the QoE measured at the client and estimated at the VNF, we only present the measurement replications that have stalling events, since stalling frequency and length are the main metrics for QoE estimation. In Figure 3.19, the x-axis indicates various monitoring points where QoE is measured and the y-axis shows the MOS

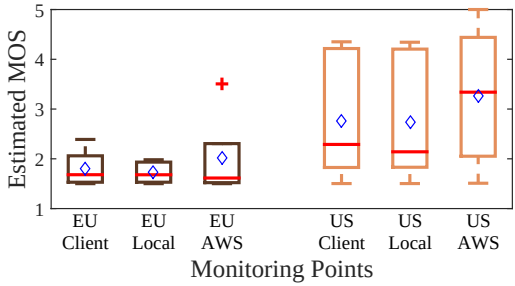


Figure 3.19: QoE Estimation at Different VNF Placements

values representative the QoE levels. Each box shows the maximum, median in the red horizontal line, and minimum MOS values. The bottom and top edges of the box plot indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers. The outliers are plotted individually using the '+' symbol. The diamond symbol in the middle of each box represents the average MOS value. The MOS is calculated from Equation (3.4) based on stalling frequency and length as presented above. The MOS can take the values: (1) bad; (2) poor; (3) fair; (4) good; (5) excellent.

Figure 3.19 shows that the local VNF can estimate the QoE with high accuracy in both the EU- and US-PoP. It is reasonable since the video flows arrive at the local VNF and show up almost instantly on the web browser due to a negligible decoding time as shown in Figure 3.17. In the EU-PoP, the VNF can estimate QoE accurately in most of the cases. Due to the VNF placement at the EU-PoP near the client, the difference in arrival time of video flows is negligible at both machines. Additionally, although the mobile network can cause packet loss that induces video stalling, lost packets are retransmitted to the machines almost at the same time. This explains a high accuracy of estimated QoE at the EU-PoP.

Conversely, we observe an overestimation of QoE measured at the US-PoP as indicated in the most right box in Figure 3.19. In this monitoring point, the VNF estimates a higher average MOS compared to the actual value obtained from the

client and from the local VNF as well. In fact, at the US-PoP, the VNF receives the video segments faster to fill up the virtual video buffer (i.e., eVideoBuff variable in Algorithm 3.1). Due to the congestion of the user mobile access network and a long distance between the US-PoP and the client, the video flows arrive at the client slower. In the worst case, the video buffer at the client has been depleted while the video segment is on the fly. In the next subsection, we investigate the influence of video segment arrival time error on the accuracy of QoE estimation.

Impact of Video Segment Arrival Time Error on QoE Estimation

Figure 3.20 shows an analysis of the video segment arrival time error between the US-PoP and the client. The x-axis shows video playback time in second. The left y-axis shows the video buffer and the right y-axis indicates the segment arrival time error measured from an analysis at packet level. Note that, the arrival time of the last packet of a segment is considered as the arrival time of that segment. The saw-tooth lines show estimated and actual video buffer as described in the previous section. The black line shows the segment arrival time error, which is the difference between the estimated and actual segment arrival time.

The figure shows that, the segment number 10 of the video arrives at the VNF 15.6 s faster than at the client. Thus, it fills up the virtual video buffer of the VNF

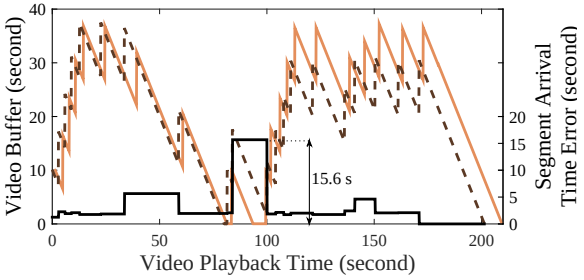


Figure 3.20: Segment Arrival Time Error Between the US-PoP and the Client

earlier where a stalling event is avoided, while the client video buffer has been empty before it arrives. This is a typical reason of QoE overestimation at the VNF in this scenario. To achieve a high accuracy in estimating QoE for video streaming, it is recommended to place the monitoring function at the client in high mobility environments or at the edge network.

3.3.6 Behavior of the Video Buffer Monitoring VNF in the Real Scenario

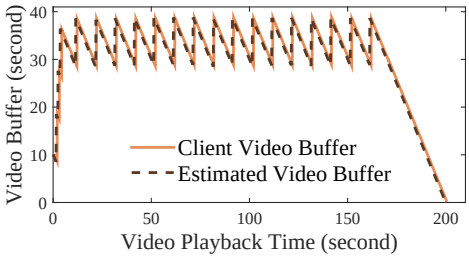
To validate our findings from the *testbed scenario*, we have implemented several experiments where the client browses video while moving in a vehicle in the *real scenario* as described in Section 3.3.3. First, we measure the mobile network condition in between the client and the AWS cloud. Table 3.4 shows the average RTT and downstream throughput of different PoPs.

Table 3.4: Mean RTT and Throughput in AWS Mobility Scenario

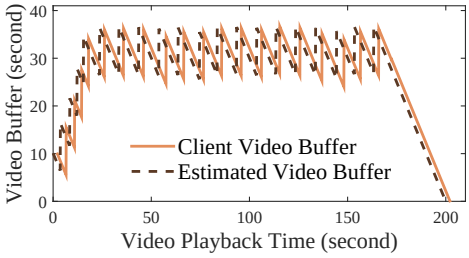
PoP	Mean RTT	STD	Mean Throughput	STD
EU-PoP	59 ms	15.4 ms	17.9 Mbit/s	8.9 Mbit/s
US-PoP	218 ms	22.5 ms	35.5 Mbit/s	20.5 Mbit/s

The table shows that, the network between the client and the EU-PoP has a lower delay. However, both network measurements have high standard deviation values. It is due to our limited number of replications and the instability of the high mobility environment. In fact, we have implemented measurements for the *real scenario* in both city and countryside. We, however only present a typical experiment in country side scenario, since the difference in result between these two locations is negligible. Figure 3.21 shows the behavior of estimated and actual video buffer obtained from the VNF and the client browser.

As expected, the results measured at the EU-PoP show a high video buffer level and high accuracy estimation as depicted in Figure 3.21a. The buffer level



(a) EU-PoP



(b) US-PoP

Figure 3.21: Video Buffer Monitoring at AWS with Mobility Client in Country Side

keeps stable in between about 39 s and 29 s, in which the higher level is the maximum amount of video buffered in the memory. The lower level is the threshold at which a new segment is requested. Conversely, the measurement results at the US-PoP show a time shifting in estimated and actual video buffer as shown in Figure 3.21b. To evaluate the accuracy of the VNF monitoring, we calculate the RMSE between estimated and actual video buffer from Equation (3.1). Figure 3.22 shows the CDF of RMSE between estimated and actual video buffer in all mobility scenarios.

At first, the figure shows that all errors of estimated and actual video buffer are larger than 2 s. Meanwhile, this error in the testbed scenario at the EU-PoP is generally smaller. It is clear that the instability of the wireless environment

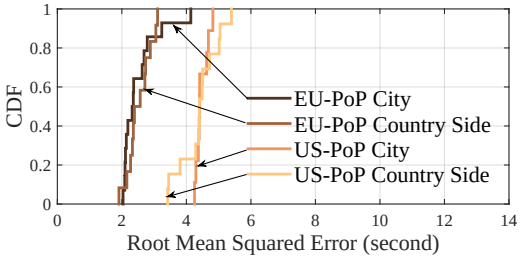


Figure 3.22: RMSE Between Estimated and Actual Video Buffer

influences the accuracy of the VNF monitoring. Furthermore, with additional high delay and congestion in the mobile network, the estimation error is doubled in case of the US-PoP scenario where most of the discrepancy between estimated and actual video buffer is larger than 4 s. Concerning the accuracy of VNF monitoring when the client moving at different regions, we observe that the difference between city and country side is small. Note that the density of mobile stations in the city is higher than in the country side. This means, the influence of mobile signal strength on the video buffer estimation is negligible.

3.4 Lesson Learned

The problem tackled in this chapter is to implement a VNF for video quality and QoE monitoring on an existing cloud infrastructure. Thereby, video buffer, quality adaptation and stalling events are monitored based on capturing video flows in the network by using DPI. The QoE for video streaming was also estimated using QoE models from previous studies where the input parameters are the extracted stalling frequency and length. To conduct the study, we first develop a VNF that can estimate the video quality based on download timestamps of video segments. Thereafter, to evaluate the performance of the function and its accuracy under the side-effects of different network conditions and function

placement, we divided the study into two stages.

In the first stage, we set up a local testbed with a network emulator to compose two scenarios of VNF placement, called ES and DC scenarios. In this stage, we evaluate the impact of bandwidth and packet reordering on the accuracy of the estimation. Our results show that, in the ES scenario, where the function is placed near the client device, we can estimate the video buffer and stalling events with a high accuracy. In contrast, when the function is placed near the streaming server in the DC scenario, we can accurately estimate the video buffer only at high link capacity of $10\,\mathrm{Mbit/s}$. At lower link capacities of $512\,\mathrm{Kbit/s}$, the buffer estimation has a constant error. This buffer error can be reduced by calculating the packet delay within the estimation algorithm. However, in case of an unstable network with the presence of packet reordering, additional errors in estimating the video stalling occur in the DC scenario and lead to an inaccurate QoE estimation. Wherein, the VNF results in a higher MOS than actual value obtained from the client by $31.11\,\%$, while in the ES scenario, this error is less than $1\,\%$. To avoid QoE overestimation in this situation, it is recommended to migrate the VNF for QoE monitoring near to the client.

In the second stage, we analyze and evaluate the accuracy of a VNF for video buffer monitoring in the AWS cloud. Through multiple testbed and real scenario experiments, we observe that the VNF monitoring for video buffer gains a high accuracy if it is placed at the EU-PoP near to the client. Specifically, in the *testbed scenario*, the estimation error is less than $0.7\,\mathrm{s}$ in $90\,\%$ of the cases compared to the actual value. However, in the *real scenario*, this error is higher than $2\,\mathrm{s}$. This result reflects a negative influence of mobile access network on the accuracy of the estimation. Regarding QoE monitoring at the EU-PoP, the VNF can estimate a similar MOS score with the client in most of the case. Nevertheless, the highest performance is achieved by the *local VNF* which is deployed at the client device.

Conversely, when the VNF is placed at the US-PoP far away from the client, it estimates the video buffer with higher error. Particularly, the discrepancy in video buffer between the VNF and the client is more than $3\,\mathrm{s}$ in both scenarios. Based on this result, we believe that the video buffer estimation error is highly

dependent on RTT and it trends to linearly grow up with an increasing distance between the client and monitoring point. In addition to this, the instability of mobile network induces QoE overestimation at the US-PoP. In the *testbed scenario*, we observe that the VNF estimates a higher MOS value compared to the one obtained from the client. This may cause a misleading performance of HAS service at the client device.

To conclude this chapter, we believe that the unify of NFV and cloud computing is promising to simplify the deployment of such a VNF QoE monitoring for HAS on an existing cloud infrastructure. Through multiple experiments, our VNF shows a high accuracy in estimating video quality and QoE if it is operating at the edge network, near to the user. However, placing the VNF at other PoP in the cloud has several issues that one must take into account.

- First, the architecture of NFV-Cloud must be well defined. For instance, since the NFV infrastructure is no longer exist in this approach, the allocation of virtual resources for VNF need to be thoroughly considered.
- Next, the video buffer estimation error is highly dependent on the distance between the client and monitoring point. This problem, however, can be solved by calculating the delay within the estimation algorithm.
- Finally, the mobile access network is an important obstacle that negatively impacts the accuracy of QoE estimation. In this situation, migrating the VNF near to the client or at the client device is highly recommended.

All in all, the insights in this chapter may help the network operators to learn the pitfalls and drawbacks of such a VNF-based QoE monitoring in the cloud.

4 Performance Evaluation of SFC Placement Algorithms in the Edge Cloud

In the previous chapter, we evaluated the influence of geographical placement on the performance of VNF QoE monitoring in the cloud. There, the monitoring function is a combination of DPI and video quality estimation in a single VNF. In fact, in the NFV architecture, a network service like QoE management typically consists of separate VNFs in a chain [21]. Thereby, corresponding VNFs can be placed in one server or distributed over NFV nodes across the network, improving the flexibility of this paradigm. For instance, the DPI part of the VNF QoE monitoring could be placed next to the user and the QoE manager could be flexibly distributed inside a large points of presence of data center.

In this chapter, we focus on the placement problems of VNF chains within data centers in the edge cloud. As today's networks typically consist of middle-boxes that perform individual network functions such as firewall, load balancer, or network address translation. These hardware appliances are statically installed in monolithic platforms in data centers or at the edge of the carrier network. Relying on this architecture, ingress data traffic is sequentially processed at these boxes before arriving at a destination. The ordered set of required boxes for a specific network service or application is referred to a function chain.

However, the hardware-based function chain has several drawbacks. First, since a function chain is statically provisioned by the network operator for a particular service, its architecture is fairly stable and only little changes are pos-

sible. Additionally, a network device provided by one vendor mostly can not be replaced by another. Thus, if one box of the chain is malfunction then the whole chain breaks. In this situation, replacement of a box is time consuming and complicated. Next, each middle-box is typically designed for a dedicated network function like a firewall. Thus, the consolidation of multiple boxes in one place significantly increases capital and operational expenditures (CAPEX and OPEX). Moreover, with the rapid growth of global Internet traffic, the network operators must cope with the increasing demand for flexibly provisioned and scalable services. If a new service requires a new function chain, it is often costly and time consuming since a fixed function chain requires new hardware installation. As a consequence, time to the market of that service is high and the economic scalability is not given.

To overcome the limitations of the hardware-based function chain, a potential solution is the use of the NFV paradigm. With the concept of Service Function Chain (SFC) [22], multiple VNFs are flexibly chained together to provide a specific service. The expected benefit of SFC is the reduction in the complexity when deploying heterogeneous network services. However, deploying such a SFC in an NFV system has several challenges. On the one hand, distributing VNFs in multiple servers will increase the length of the chain if the servers are placed in different data centers. This will considerably rise the latency within the SFC itself and can reduce the QoE. In this situation, placement problems of the SFC must be well-defined and optimized to minimize total latency. On the other hand, since the VNFs can be placed in different servers, their resource utilization must be taken into account as well. In fact, complex optimization problems can be formulated for a given system, but their solution can be time consuming regarding the complexity theory [102, 103]. Thus, heuristics are required to quickly obtain solutions which achieve a close-to-optimal performance.

To address these problems, we propose and evaluate two heuristics for distributing VNFs of service chains in data centers of an edge cloud, named *centralization* and *orchestration* algorithms. These placement algorithms aim to minimize total latency or server utilization. They are evaluated against optimal so-

lutions for the placement problems, which are formulated and solved by using Integer Linear Programing (ILP). For the performance evaluation, we extend the event-based EdgeNetworkCloudSim simulator [24] with the inclusion of the CPLEX Optimizer toolbox[1]. This toolbox uses Optimization Programming Language (OPL) to express the ILP mathematical model. Then, the OPL model is solved using CPLEX Optimizer.

In fact, the SFC placement problem is widely studied in different directions. The objective of existing works has focused on cost reduction [104, 105], virtualization trade-off between different objectives [106, 107], or optimizing energy consumption [108–110] among others. Our study is different from these works, since we consider the SFC placement problem in the context of edge cloud computing, wherein the user is close to a data center in an edge cloud. In addition to this, the user device is included as the end point of the whole chain.

In this study, we use EdgeNetworkCloudSim to simulate a fixed network topology in an edge cloud. In the simulation, users randomly request service chains consisting of three VNFs that can be placed on different servers. The performance of all placement algorithms is evaluated with respect to QoE in terms of service response time and resource consumption in terms of number of utilized servers. Herein, the response time or a service may influence the QoE perceived by the user as described in Chapter 2. Therefore, an efficient placement algorithm helps to increase the user satisfaction with the service. Our simulation result shows that the optimized solutions obtained by using ILP model achieve lowest service response time and least server utilization rate compared to the others. However, the heuristic algorithms are able to come close to the optimum by simple placing rules. Besides, we observe that the service response time linearly grows with the increasing number of VNFs in the chain. The obtained linear regression models of the service response time depending on the number of VNFs in a chain can later be used as analytical models for optimization or reference models for QoE monitoring. Our insights may help network operators and the research community to quickly compute good SFC placements

[1] https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer

for NFV infrastructures in an edge cloud context.

The content of this chapter is mainly taken from [6]. The remainder of it is structured as following. First, Section 4.1 presents background and the state of the art in SFC. Subsequently, the description of four placement algorithms are presented in Section 4.2. Thereafter, we describe the extension of EdgeNetworkCloudSim, edge cloud topology, and several performance metrics in Section 4.3.1 to 4.3.4. Next, the outcome of our study is detailed in Section 4.3.5. Finally, Section 4.4 concludes this chapter with lessons learned.

4.1 Background and Related Work

In this section, we first introduce NFV and its architecture in Section 4.1.1. Note that, this subsection is different from Section 3.1.5 in the Chapter 3, since NFV Cloud is another approach where the NFV infrastructure is replaced by a cloud infrastructure. Next, in Section 4.1.2, we present the definition of simulated SFCs. Subsequently, we give an overview of cloud computing simulators in Section 4.1.3 and the state of the art in the SFC research area in Section 4.1.4.

4.1.1 The Emergence of Network Function Virtualization

In this subsection, we first introduce the concept of NFV, its advantages and challenges. Subsequently, we present the NFV architectural framework and the definition of VNF forwarding graph.

Transformation of Legacy Approach into NFV

The tremendous growth of global Internet traffic has been forcing network operators to struggle with reducing CAPEX and OPEX [60]. Additionally, they must cope with the increasing demand for flexibly provisioned services. Thereby, the network service must be provided by the network operators with a high QoE in order to achieve high customer satisfaction and to avoid user churn [11].

To handle these problems, a more innovative and agile network technology has been emerged, called Network Function Virtualization (NFV). The concept of NFV was first introduced in the conference on Software-Defined Networking (SDN) and OpenFlow in October 2012 presented in a white paper [20]. The main idea of this paradigm is to decouple the network functions from their physical hardware. Figure 4.1 shows a vision for NFV where physical network devices are transformed into VNFs by leveraging virtualization technology.

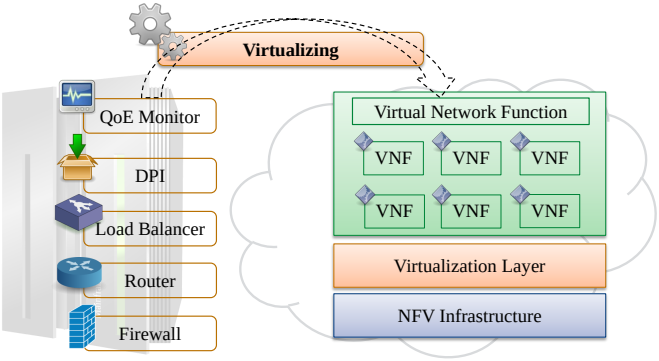


Figure 4.1: NFV Transformation from Legacy Network Functions

In this concept, a routing function, for example, can be detached from its expensive hardware router to become a plain software, which can run on any commodity server. Similarly, other legacy network appliances like firewall, load balancer, DPI, or QoE monitoring can also be virtualized as applications, see also Chapter 3. Such kind of application is called Virtual Network Function (VNF) [20]. A VNF fulfills a designated task and can be installed on any standard industry server. Based on this innovation, NFV is promising to feature a series of benefits opposed to classical hardware-oriented approaches.

Benefits of the NFV Paradigm

Reduce CAPEX and OPEX: Since VNF is a piece of software, multiple types of VNF can be consolidated in a high volume server. Herein, each VNF performs a particular network function like firewall or routing. Thus, instead of installing new and expensive network appliances in server racks, the consolidation of VNFs in a server significantly reduces CAPEX. In addition to this, VNFs can be migrated to another data center without the installation of new physical hardware. Next, as plain software can be easily installed, tested, and updated on the same infrastructure, this benefit helps to decrease the development cost of a VNF. Besides, since the deployment process of a VNF can be automated across a large network, time to market of a service is considerably reduced. This leads to a substantial reduction of OPEX [111, 112].

High Flexibility: With the characteristics of an application, a VNF can be flexibly distributed over various network nodes or at the end user device. It also can be migrated from a data center to the edge network to avoid high latency by a long distance to the user. Moreover, VNF can be easily allocated with virtual resources like CPU or memory without the need to upgrade the underlying physical hardware [57, 58]. This reduces downtime and thereby prevents potential loss in revenue.

Easy Scaling: Since the traditional network is built from dedicated appliances, it must be optimally dimensioned to be able to properly handle traffic in peak hours. Nevertheless, unexpected traffic overload still can occasionally happen that may negatively influence QoS, and thus degrade the user perceived QoE. Moreover, a well-dimensioned network device for peak hours will not be efficiently utilized on average, since high traffic demand rarely occurs. Therefore, in off-peak times, hardware resources and energy might be wasted. To overcome this limitation, the NFV paradigm allows to quickly scale up a virtual machine or scale out by deploying additional VNFs to handle peak traffic load in real time [113]. Note that, elastic scalability in NFV only requires server resources like computation or memory that is conventionally inexpensive compared to dedicated network appliances. Furthermore, in the NFV architecture, hardware

resources are shared between VNFs. This means, resources released by a terminated VNF can be re-allocated to another one. This dynamic allocation allows administrators to efficiently exploit the capability of the NFV infrastructure.

Challenges in the Deployment of NFV

Performance: Despite the many benefits of innovation, the deployment of NFV may be constrained by several challenges which need to be addressed. First, a VNF is expected to have as high performance as traditional hardware-based network appliance. Since a VNF is software-based, its performance depends on the virtual environment it is operating on. A bottleneck at the processor or shared memory may cause a high latency or packet loss [114]. Additionally, several VNFs like virtual DPI may require a fast packet processing mechanism to reduce delay caused by multiple copying processes between buffers or memories [115]. A solution for this issue is the use of a Data Plane Development Kit (DPDK) [116]. A DPDK is implemented as a kernel bypass, thus packets are processed completely in user space without involving the kernel. This mitigates the problems of memory allocation per packet and context switching.

Legacy Support: One of the most important selling points of NFV is reducing CAPEX and OPEX, since its infrastructure only requires standard servers and switches. This advantage becomes a significant challenge keeping it compatible with the legacy network. It is due to the fact that traditional network products might not be upgraded to support the co-existent of NFV. Therefore, a well-defined migration path toward NFV is necessary while maintaining traditional network infrastructure in one place.

Orchestration and Automation: In an NFV architecture, VNFs must be automatically initiated and allocated resources on demand. To this end, an orchestration algorithm is required to flexibly allocate and re-allocate resources for the VNFs. Note that a VNF should only receive sufficient resources it needs. There, a major challenge is to dynamically react on changing resource demands without causing any service interruptions or performance delays. To solve this problem, the resource utilization can either be checked periodically or in any

other timing pattern by the orchestration algorithm or upon the notification of the under-provisioned Virtual Machine (VM). Thereby, resource utilization of the whole infrastructure is monitored and unused resources are reallocated to other processes or shut down to save energy.

NFV Architecture

Figure 4.2 shows an overview of an NFV architectural framework. The NFV framework consists of three main components, which are *VNF*, *NFV Infrastructure (NFVI)* and *Management and Orchestration (MANO)* [59]. A VNF is an actual network function that is virtualized and can be automatically deployed on top of the NFVI. An example of VNFs are the Evolved Packet Core (EPC) network elements like Mobility Management Entity (MME) or Serving Gateway (SGW) [117]. An Element Management (EM) is responsible for the functional

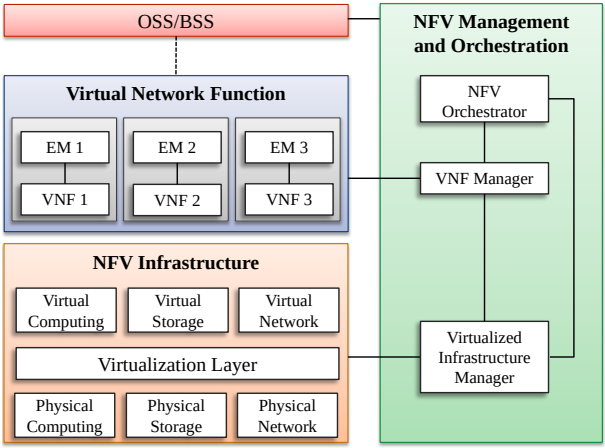


Figure 4.2: Overview of the NFV Architecture

management of one or several VNFs.

The NFVI is the whole physical hardware and software which creates an environment where VNFs are deployed and managed. From the bottom, it consists of physical resources such as computing, storage, or network. These components are generally built up from standard commodity servers. The connectivity to the VNFs is handled by the virtualization layer. This layer is responsible for the process of abstracting physical resources, enabling the implementation software of VNFs and providing them with virtualized resources. Based on this, virtual computing, storage or network are allocated to the VNFs in the form of hypervisors and VMs. The virtualization layer plays an important role in the principle of NFV since it ensures the decoupling of VNF software and isolating the VNF lifecycle management from the underlying physical hardware.

The MANO consists of three main components, which are the NFV Orchestrator (NFVO), the VNF Manager (VNFM), and the Virtualized Infrastructure Manager (VIM). The NFVO has the responsibility for the global management of the NFV infrastructure and network services. It manages the topology of network services and can create an end-to-end service over multiple VNFs. In the MANO component, depending on requirements, the NFVO can connect either with a VIM or directly with the NFVI to coordinate, authorize, release, and engage them. The VNFM is responsible for the lifecycle management of VNF instance. Herein, a VNFM is able to initiate, scale, or terminate a VNF instance. The VIM is responsible for the dynamic allocation of virtual resources to the VNFs. Another function of a VIM is to support the management of VNF Forwarding Graph (VNFG) by organizing virtual links, networks, subnets, and ports [118].

VNF Forwarding Graph

In a traditional network architecture, different network devices can be interconnected by bidirectional links to perform a specific service together. For instance, a chain of appliances on the path to a web server tier typically consists of a firewall, a NAT, and a load balancer. In the NFV paradigm, the virtual form of these network devices can also be interworked by logical links. ETSI NFV

ISG describes this interconnectivity in [97, 119] as a VNFG. Specifically, a VNFG defines a sequence of VNFs and virtual links that packets traverse, which is kind of like a virtual network. The VNFs in a VNFG are constituent functional components of a network service. Thereby, the behavior of the network service associates with the functionality of each VNF. However, the order or sequence of the VNFs in a VNFG is not required [120]. A similar concept to VNFG is a Service Function Chain (SFC) defined by IETF SFC WG in [22]. Typically, in a SFC, each VNF executes a certain function and all VNFs must be processed in a specific order. Thus, VNFs can be dynamically deployed in a server or distributed over different data centers to meet various requirements. For example, multiple VNFs can be instantiated for a particular function in order to increase resilience or for load balancing.

4.1.2 Definition of Simulative Service Function Chain

In this work, we simulate three types of personal services, *Video Streaming (STR)*, *Web* and *Database (DB)*. These services are characterized by their own virtual machine resource demands and traffic characteristics, and are requested and used by only one single user. Note that these services do not resemble real cloud applications, but were mainly specified in order to have different service chain characteristic. Each service is requested and processed as a chain of VNFs. Thus, in the remainder of this chapter the terms SFC and service will be used interchangeably. In the simulation, we define a SFC consisting of three VNFs that must be executed in order to provide full service functionality. This definition is also consistent with the SFC described in [22].

Figure 4.3 shows an overview of communication between a user and a SFC. There, each VNF is installed at a VM. The arrows represent the direction of the data flow and also the sequence of VNF execution in the SFC. Specifically, when the user requests a service, the request is processed at VNF1, followed by VNF2, and VNF3. Then, the response message will also be sent back in sequence from VNF3 to the user. Therefore, the placement of VMs primarily influences

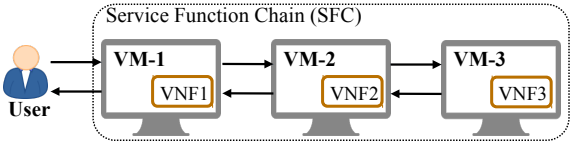


Figure 4.3: Overview of Service Function Chain Communication

the latency between the VNFs in the chain, and thus also the service response time. This makes it crucial to design placement strategies, which quickly find a good possible placement of SFCs in terms of QoE and resource utilization.

In the SFC, the type of a VM also must correspond to the VNF requirements. The VM types are predefined based on instance types introduced by Amazon Elastic Compute Cloud[2]. Table 4.1 presents the definition of the used VM types in EdgeNetworkCloudSim. Here, we use three types of VMs which are *T2Nano*, *T2Small*, and *T2Large* with different computing and memory capacities. Based on this, three types of simulated personal services are characterized by their own set of VM types which will be detailed in Section 4.3.

Table 4.1: VM Type Definition

VM Type	CPU	RAM
T2Nano	1	1024 MB
T2Small	2	2048 MB
T2Large	4	4096 MB

[2]https://aws.amazon.com/ec2/instance-types/

4.1.3 Cloud Computing Simulator

In [121], Calheiros et al. present their work on an event-based simulator for cloud computing infrastructure, called CloudSim. This simulation framework is extensible and enables data center resource scheduling and provisioning. An extension of CloudSim with a GUI is CloudAnalyst, introduced in [122]. CloudAnalyst is designed to overcome the limitation of CloudSim that does not provide a user GUI. This simulator allows users to easily create their own cloud with geographically distributed servers over the world map. Based on this, a user can evaluate the behavior of cloud applications in a large scale. Another extension of CloudSim that enables networking is NetworkCloudSim [123]. With the inclusion of various network topologies and modeled network communication, it provides users with a general view of data processing and transferring under different network configurations in the cloud.

In [24], Seufert et al. introduce EdgeNetworkCloudSim as an extension of NetworkCloudSim. Thereby, EdgeNetworkCloudSim moves from batch-like processing of computation jobs to persistent and personalized cloud services that are implemented in an edge network cloud. To this end, the authors allow to define several characteristics of cloud services which can be processed in a chain of virtual machines. The framework is able to simulate service requests and user requests to services and can compute response times and resource utilization among others. In this chapter, we extend EdgeNetworkCloudSim to simulate and evaluate the performance of different SFC placement algorithms.

4.1.4 State of the Art in Service Function Chaining

The architecture for the specification of SFCs is officially described in [22] by IETF. Since SFCs are promising to decrease the complexity of the service deployment in an NFV architecture, they receive a considerable interest from research community. In [124], John et al. introduce several research topics on network service chaining. In [104], Savi et al. study the impact of processing capability on the placement of service chains. They model a set of NFV nodes hosting one

or more VNFs that are used to form a service chain. This model is formulated using an ILP and solved by CPLEX. The main objective of the model is to minimize the number of active NFV nodes with regard to upscaling and context switching costs of processing task. The authors conclude that with the increasing number of service chains, the context switching costs strongly influence the implementation cost of NFV. Conversely, the number of service chains does not change how the upscaling costs translate into the cost for NFV implementation.

In [106], Luizelli et at. consider the placement of SFC where the placement problem is formulated by using an ILP model. The main objective of the model is to minimize the number of VNF instances mapped to infrastructure. The authors conclude that the ILP model leads to a reduction of up to $25\,\%$ in the end-to-end delay compared to a baseline. However, their baseline solution is another ILP model, where the objective function is changed to minimize the chain length. The authors propose a heuristic algorithm to guide the search for a solution. This contribution helps to reduce the solving time of the CPLEX Optimizer.

The authors in [107] formalize the chaining of VNFs using a context-free language. Then, they formulate the placement problem of chained VNFs as a Mixed Integer Quadratically Constrained Program (MIQCP) with regard to three different objectives. Their results show a trade-off between optimizing the remaining data rate, latency, and number of used nodes. In [125], the authors propose the Merge-RD algorithm to place service functions with respect to minimizing energy consumption. They use GreenCloud to simulate a data center topology and to evaluate the performance of the proposed algorithm. Their algorithm performs better than StEERING, PowernetS and Algorithm H [108, 126, 127]. However, with $O(n^4)$, the complexity of the algorithm is quite high and time consuming. Other approaches in energy-efficient and bandwidth-efficient SFC placement are presented in [109] and [110], respectively. In [105], Bari et al. propose a heuristic algorithm and an ILP model for optimization for SFC placement. The algorithms aim to reduce CAPEX and OPEX of VNF deployment in the operator network. They conclude that the heuristic algorithm outperforms the optimal solution in the aspect of execution time.

The method of using an ILP model to formulate the placement problem of VNFs is widely studied. In [92], Bouet et al. propose a study that aims to minimize the overall cost of vDPI deployment in a NFV infrastructure. The authors conclude that the network structure and costs strongly influence the execution time of a vDPI function. However, the authors do not consider the chaining form of vDPI in their ILP model. Similar studies on solving the placement problem of VNFs formulated by using an ILP model are presented in [91, 107, 128].

The existing studies use similar methods as our study to some extents in the aspect of optimizing SFC placement. Most of the works are based on mixed integer programming with different objective functions and constraints. Although, their approaches can deliver optimal solutions for individual problems, the performance of their proposals still need to be assessed in a real scenario. Moreover, their result is mostly obtained by solving the ILP model using CPLEX. The authors in [125] use GreenCloud simulator for examining their algorithm, however their focus is on energy consumption only.

Our study is different from the mentioned research works since we compare four different approaches for SFC placement. This helps the network operator to have a comparative view of the advantages and drawbacks of each solution. Moreover, we consider an edge cloud context where the user is close to the data center. In our heuristic algorithms and ILP model, the user device is also included as the end point of the whole chain. Furthermore, to the best of our knowledge, this study is the first that uses EdgeNetworkCloudSim to simulate and evaluate the influence of different SFC placement strategies on service response time or server utilization. Wherein, user requests and SFC placement are done consecutively on the same platform that increases the practical of our approach.

4.2 SFC Placement Algorithms

In this section, we present four SFC placement strategies, which are *Centralization (CEN)*, *Orchestration (ORC)*, *Service Time Optimization (STO)*, and *Resource Optimization (RO)*. Table 4.2 shows the summarize of the four algorithms.

Table 4.2: Summarize of the Placement Algorithms

Reference	Algorithm Name	Objective
CEN	Centralization	To find the closest servers for VMs
ORC	Orchestration	To shorten the length of the chain
STO	Service Time Optimization	To optimize the service response time
RO	Resource Optimization	To optimize the server utilization

On the one hand, CEN and ORC are heuristic approaches. These algorithms are designed and included within EdgeNetworkCloudSim to search quickly for a proper placement of SFCs. On the other hand, STO and RO use the optimization model with the objective of minimizing service response time and server utilization, respectively. Thereby, the placement problems of SFC are formulated using an ILP model. Then, the problems are solved by using CPLEX Optimizer to provide the simulator with specific locations of all VMs.

4.2.1 Centralization Algorithm

The centralization (CEN) placement algorithm tries to place all VMs of a SFC as close as possible to the user, meaning to have the lowest delay between the VMs and the user. This algorithm is useful in classical cloud computing where independent VMs of a user should be placed close to him. However, it is unclear how the CEN performs in case of service chains with communicating VMs. Algorithm 4.2 shows the simplified pseudo-code of the centralization approach.

Data provided for the algorithm are an ordered list of VMs in a chain, *UserVM* location, and type of service. When receiving requests to create VMs in a data center (*DC*), the simulator starts to find a closest *DC* to the user for each VM. If there is still an available *DC*, then the VM is placed in sequence and the service can be processed. All the VMs can be placed in one *DC* or distributed over different data centers. If at least one VM could not be placed, there are not enough resources (i.e., CPU or RAM) in the system to deploy the entire SFC. In this case,

Algorithm 4.2 Centralization

```
1:  Input: List of VMs, requested service, and UserVm
2:  Initialization;
3:  for each vm in chain do
4:      DC = findClosestDcToUser(uservm);
5:      if DC == -1 then
6:          abandonService();
7:      else
8:          createVmInDc(vm, DC);
9:      end if
10: end for
```

the requested SFC will be discarded or blocked.

In the CEN approach, all VMs of the service chain will be placed around and close to the *UserVM*, but the algorithm does not try to reduce the length of the chain. A large VM might be placed far away from the *UserVM* due to the lack of resources at the closer *DC*, but a smaller VM later in the chain may fit. As a consequence, the length of the chain is increased and data oscillate between data centers. This might significantly influence the efficiency of the algorithm with respect to service response time.

4.2.2 Orchestration Algorithm

The orchestration (ORC) algorithm differs from the CEN, where only the first VM in the chain is attempted to be placed as close as possible to the user. For the subsequent VMs, the aim is to place them close to the previous VM. Based on this, ORC tries to shorten the length of the chain that reduces the latency. Algorithm 4.3 shows the simplified pseudo-code of the ORC algorithm.

Similar to CEN, this algorithm is provided with the ordered list of VMs, *UserVM* location, and type of service. At first, it tries to find a closest *DC* to the *UserVM* for the first VM in chain. Afterwards, the algorithm tries to place the next VM in chain as close as possible to the previous one in sequence. Additionally, the next VM is only sent if the previous one has been successfully placed.

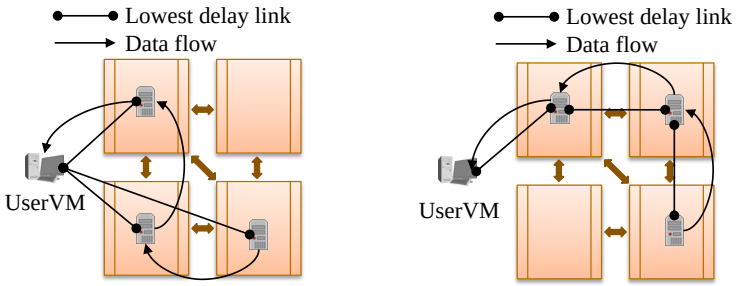
Algorithm 4.3 Orchestration

```
1:  Input: List of VMs, requested service and UserVm
2:  Initialization;
3:  DC = findClosestDcToUser(uservm);
4:  if DC == -1 then
5:      abandonService();
6:  else
7:      createVmInDc(firstVm, DC);
8:      for next vm in chain do
9:          DC = findClosestDcToPreviousVm();
10:         if DC == -1 then
11:             abandonService();
12:         else
13:             createVmInDc(vm, DC);
14:         end if
15:     end for
16: end if
```

Thus, ideally all VMs are placed in the same *DC* if it has enough resources. If no *DC* is found in any case, the requested service is abandoned.

Figure 4.4 shows a comparative view of the CEN and the ORC algorithms. As indicated in Figure 4.4a, CEN tries to search all closest servers to place VMs. As a consequence, the data flow may not travel in a shortest path that potentially increase service response time. Conversely, ORC overcomes the limitation of CEN, as it avoids placing next VMs close to the user rather than close to the previous VMs, as shown in Figure 4.4b. Based on this mechanism, data flow can pass in a possible shortest path that helps to reduce the delay within the SFC.

However, prioritizing the closest *DC* for the first VM is not always the best option. Especially, when this closest *DC* has only resources for some VMs and the other VMs must be distributed to another farther *DC*. This leads to an increased hop count within the chain and increased latency. In this case, placing all VMs in the farther *DC* might be better, since the delay within the chain would be zero. However, a heuristic for this idea is not evaluated in this work. In the next subsection, we present an optimized approach with the use of an ILP model.



(a) Centralization Algorithm *(b) Orchestration Algorithm*

Figure 4.4: Graphical View of CEN and ORC Algorithms

4.2.3 Service Response Time and Resource Optimization

To formulate the problem and align with the definition of the infrastructure and SFC described before, we consider that each server is a member of a data center. We assume that all servers within a data center are fully connected with links of practically infinite bandwidth and zero delay. The first node of a network inside a data center is considered to be the network gateway. A *UserVM* is declared as statically allocated in a server within a dedicated *UserDC*. The *UserVM* only acts as the source of requests to a particular service chain. Table 4.3 shows the notations that are used for the formulation of the optimization problem. Based on the provided notations and the considerations mentioned before, the optimization problem is formulated as follows.

Given:

$$R = \{CPU, Memory\}$$
$$R' = \{Bandwidth\}$$
$$M = \emptyset$$

Table 4.3: Summary of Parameters Used in the ILP Model

Parameters	Description
T	Set of application components
C	Set of channels between application components, $C \subseteq T \times T$
H	Set of hosts
L	Set of links between hosts, $L \subseteq H \times H$
S	Set of user application components statically allocated at hosts, $S \subset T$
H^S	Set of hosts where static user application components are placed, $H^S \subset H$, $f : S \to H^S \mid \forall h' \in H^S, \exists t' \in S : h' = f(t')$ (f is subjective)
R	Set of unique resources offered by hosts
R'	Set of unique resources offered by links
M	Set of monitored metrics at hosts
M'	Set of monitored metrics at links
a_t^r	Amount of resource r demand by application component t
i_h^r	Capacity of resource r at host h
β_h^r	Amount of resource r available at host h
m_h^k	Measured value of metric k at host h
c_{sd}^r	Amount of resource r demand required by channel (s, d)
b_{uv}^r	Amount of resource r available at link (u, v)
μ_{uv}^k	Measured value of metric k at link (u, v)

Minimize:

- ***Objective 1:*** Minimize service response time (STO) of service chain

$$Objective_1 = \sum_{(s,d)\in C} \pi_{uv,sd}\mu_{uv}^{Delay}, \quad \forall (u, v) \in L. \tag{4.1}$$

- ***Objective 2:*** Minimize resource utilization (RO)

$$Objective_2 = \sum_{h\in H} (min\{\sum_{t\in T} \sigma_{ht}, 1\} \frac{100\beta_h^{CPU}}{i_h^{CPU}}). \tag{4.2}$$

Objective 2 aims to minimize the product of the number of servers used in a placement and the percentage of available CPU. This optimization procedure will attempt to collocate VMs in a server but will have the preference to an already utilized server. Thus, the servers that have zero utilization will not be activated until the others have been fully utilized. By doing this, unused servers can be put in the idle state to save energy. However, at the initial placement all servers will have the same probability to be selected.

Subject to:

$$\pi_{uv,sd} \in \{0,1\}, (s,d) \in C, (u,v) \in L. \tag{4.3}$$

In Equation (4.3), $\pi_{uv,sd}$ is a decision variable and equals to 1 if task channel (s,d) is routed from link (u,v), 0 otherwise.

$$\sigma_{ht} \in \{0,1\}, \quad h \in H, t \in T. \tag{4.4}$$

In constraint (4.4), σ_{ht} is a decision variable and equals to 1 if task t is assigned to host h, 0 otherwise.

$$\sum_{h \in H} \sigma_{ht} = 1, \quad \forall t \in T, \tag{4.5}$$

$$\sigma_{ht} = 1, \quad h \in H^S, t \in S, \tag{4.6}$$

$$\sum_{t \in T \setminus S} \sigma_{ht} = 0, \quad \forall h \in H^S. \tag{4.7}$$

Constraint (4.5) ensures that a task (or application component) is assigned only to one host. The static placement of the user task is defined in Equation (4.6), it is given as an input to the problem and not decided. Whereas, constraint (4.7) specifies that user applications are only placed in hosts assigned for them.

$$\sum_{t \in T} \sigma_{ht} \alpha_t^r \leq \beta_h^r, \quad \forall r \in R, \forall h \in H. \tag{4.8}$$

Equation (4.8) stipulates that the considered host h must have enough resources to allocate the application component t.

$$\sum_{(u,h)\in L} \pi_{uh,sd} + \sigma_{hs} = \sum_{(h,v)\in L} \pi_{hv,sd} + \sigma_{hd}. \tag{4.9}$$

Constraint (4.9) captures and expresses in one equation,

- the unsplittable flow constraint: A channel uses a single outgoing link from source and a single incoming link at destination and does not split,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = 1 \quad if \quad \sigma_{us} = 1,$$
$$\sum_{(h,v)\in L} \pi_{hv,sd} = 1 \quad if \quad \sigma_{vd} = 1,$$

- the collocation of tasks: A communication path is not required in the case that both s and d are assigned to the same host (and no capacity checking),

$$\sigma_{hs} = \sigma_{hd},$$
$$\pi_{uu,sd} = 0,$$

- the flow conservation constraint: No traffic is stored in a node unless this node is the source or the destination or collocated source and destination,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = \sum_{(h,v)\in L} \pi_{hv,sd},$$
$$\forall h \in H : \sigma_{hs} = 0, \sigma_{hd} = 0.$$

$$\sum_{(u,h)\in L} \sigma_{hs}\pi_{uh,sd} = 0. \tag{4.10}$$

Constraint (4.10) makes sure that there is no loop in the path before reaching

destination.

$$\pi_{uv,sd} = \pi_{vu,ds}, \quad (s,d),(d,s) \in C, (u,v),(v,u) \in L. \tag{4.11}$$

As determined in Equation (4.11), a bidirectional communication between two tasks is routed through the same bidirectional overlay path. In addition to this, upstream and downstream of a flow is not routed separately.

$$\sum_{(s,d)\in C} \pi_{uv,sd} c_{sd} \leq b_{uv}, \quad \forall (u,v) \in L. \tag{4.12}$$

Constraint (4.12) guarantees that the link (u, v) must have enough resource required by channel (s, d).

4.3 Simulative Performance Evaluation of SFC Placement Algorithms

In this section, we first present the EdgeNetworkCloudSim extension in Section 4.3.1, which is used to simulate and evaluate different SFC placement algorithms in the edge cloud. Subsequently, we introduce the edge cloud topology and simulation configurations in Section 4.3.2, followed by service chain characteristics used for the simulation in Section 4.3.3. Thereafter, in Section 4.3.4, we present several performance metrics to evaluate the performance of the algorithms. Finally, Section 4.3.5 presents the outcome of this chapter.

4.3.1 EdgeNetworkCloudSim Extension

In this work, we extend EdgeNetworkCloudSim to simulate a SFC running on the edge cloud and to evaluate the performance of different placement algorithms. Figure 4.5 shows the main components of the simulator. The extended components are highlighted by colored boxes.

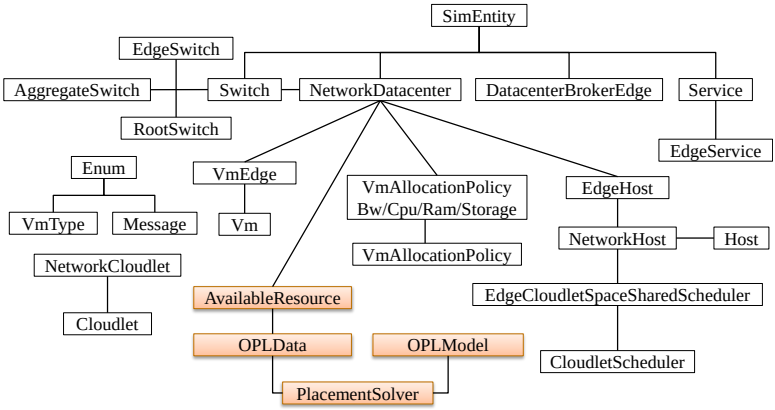


Figure 4.5: Overview of EdgeNetworkCloudSim Architecture

From the top of the figure, **SimEntity** models an entity in the simulation. The entity can send and receive messages as well as fire and handle events. Next, **NetworkDatacenter** models the core cloud infrastructure that is offered by a provider like Amazon Web Services. Each data center contains one or multiple servers that are modeled by the **NetworkHost** class. **Service** is a *NetworkCloudlet* running on a VM that represents a SFC in this study. Depending on the service type, each SFC has different resource demand and characteristics. In the simulation, resources are allocated for each VM by its demand and also depending on the amount of available resources at the data center. The resource allocation policy is in charge by **VmAllocationPolicy**. The detail of other components can be found in the original work [24, 121, 123].

In fact, placement algorithms for VMs can easily be implemented in EdgeNetworkCloudSim. However, the framework did not provide means to compute an optimal placement with the help of an ILP. Therefore, EdgeNetworkCloudSim is extended to be able to implement OPL models and solve them with a CPLEX

Optimizer. Specifically, the following new classes are added to the simulator to achieve the mentioned goal. First, the **AvailableResource** class is implemented whenever a new service is requested. It monitors resource utilization (e.g., CPU, RAM) of all servers in data centers. Additionally, the resource demand of the VMs in SFCs is also provided. This information is used for the **OPLData** class.

The **OPLModel** class contains the pre-defined OPL model with two objectives which are minimizing service response time and resource utilization. The mathematical expression of the model was presented in Section 4.2. The **OPLData** class contains the data of the OPL model. The data consists of static and dynamic information. The static information like topology and link resources between nodes is initially provided. The dynamic information consists of available resources of the servers (e.g., CPU, RAM), resource demand of the VMs, and location of the user. Note that the user in EdgeNetworkCloudSim is simulated as another VM, called *UserVM*. This *UserVM* is located in a server of a dedicated data center, called *UserDatacenter*, which can not be used to place virtual machines of SFCs. The dynamic information is gathered by the **AvailableResource** class and regularly updated whenever a new SFC is requested.

Finally, the **PlacementSolver** class is triggered when a new service has been requested and the *UserVM* has been already specified by the simulator. This class exploits the CPLEX Optimizer toolbox to solve the OPL model and returns one optimal solution each time. The solution is the specific location of each VM of the service chain. EdgeNetworkCloudSim can now make use of the solution and place these VMs accordingly to start the service. If no solution can be found due to insufficient available resources in the system, EdgeNetworkCloudSim will discard the incoming SFC request. Note that a new optimal placement will also be computed when a SFC has been terminated and allocated resources are released.

4.3.2 Edge Cloud Topology

Edge computing is a method of enhancing cloud computing systems by offloading applications, services and hardware resources to the edge of the net-

work [93]. Therefore, the edge cloud introduces a new intermediate layer at the edge of the network, which is physically placed in between the cloud data center and the user. This removes a major bottleneck and reduces high latencies in services due to long distances to the user. In this chapter, we assume that a user device is directly connected to an edge cloud with four data centers. The user requests a service in the edge cloud, and receives a response from servers located in these data centers. Figure 4.6 shows the topology of the simulation.

This topology is designed based on a real testbed of the EU H2020 INPUT project [23]. It consists of four data centers (*DC*) and four user data centers (*UserDC*). One of the *DCs* has two servers, the others have only one. These servers have different resource capacity. In the simulation, the VMs of SFCs are distributed over these five servers of the edge cloud depending on their avail-

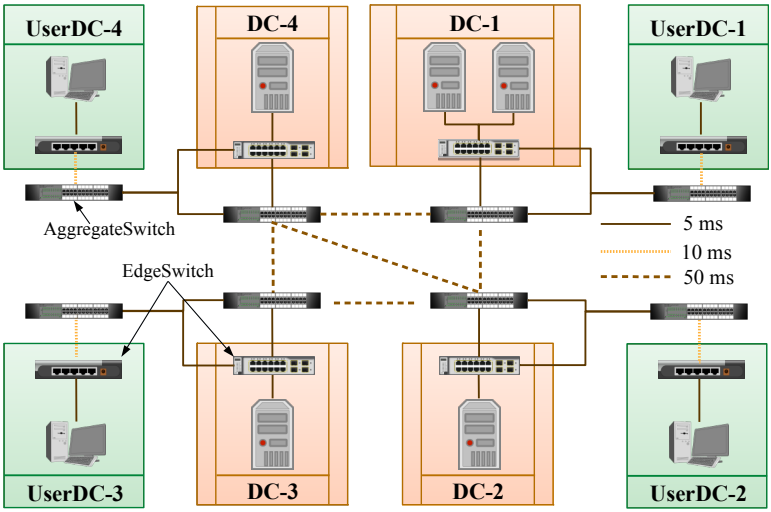


Figure 4.6: Overview of the Edge Cloud Topology in the Simulation

able resources. The user device is simulated as a *UserVM* that is installed within a server in the *UserDC*. Each *UserDC* is connected to a *DC*, thereby, representing all users which are close to that edge data center. Each *UserDC* is considered to have unlimited resources to host *UserVMs*. The interconnection of *DCs* and *UserDCs* in the edge cloud is operated by different link capacities via two types of switches. The server in each data center is connected to its EdgeSwitch with a $5\,\mathrm{ms}$ delay direct link. However, in *DC-1*, two servers are set to be interconnected with zero delay. The EdgeSwitch of a *DC* is connected to its AggregateSwitch via a $5\,\mathrm{ms}$ delay connection. While, in case of the *UserDC*, this connection has $10\,\mathrm{ms}$ delay. All AggregateSwitches are interconnected through a link with $50\,\mathrm{ms}$ delay. In the simulation, this topology is mapped with a BRITE file [129] for modeling link bandwidth and associated latency. Since we only consider link delay in the topology of the simulation, we configure the link bandwidth of the topology with a large number to ensure there is not any bottleneck in the network.

4.3.3 Service Chain Characteristics

As briefly described in Section 4.1, we simulate three types of personalized services. Each service chain has a single *UserVM* with a fixed location in a *UserDC* sending requests only to its particular service chain. Each service chain requires three VMs with various types. Table 4.4 summarizes the simulated services and their required VM types.

The table shows that the Streaming and Web services require a similar total size of VMs. However, their characteristics are different. The Streaming service

Table 4.4: Summary of Services and Their Required VM Types

Service	**VM-1**	**VM-2**	**VM-3**
Video Streaming	T2Small	T2Nano	T2Large
Web	T2Large	T2Small	T2Nano
Database	T2Nano	T2Nano	T2Small

is simulated to deliver video with different lengths. Specifically, the video length is distributed exponentially around a mean of 30 s. Each delivered video chunk has 2 s in length and 1100 KB in size. Consequently, when a Streaming service is requested, the number of responses is corresponding to the number of video chunks. That is the reason why a Streaming service has a much longer service response time than the other services. On the other hand, Web and Database services have only one response per request, but their response data size is distributed around a mean of 2000 KB and 50 KB, respectively. Besides, to simulate user-like behavior, we use an exponential distribution for service requests (i.e., the time at which the service is instantiated), service life time (i.e., the time until the service is terminated), and user request inter-arrival times (i.e., the time at which the user sends a request to the service) in the simulation.

4.3.4 Performance Metrics

The main goal of the simulation is to evaluate the performance of different SFC placement strategies. To this end, we define two metrics that are used to compare these strategies in different aspects.

Service Response Time and Hop Count

Service response time is the amount of time between the user request a SFC and the reception of its response. In the simulation, the service response time consists of the sum of all link delays and the processing times at all VNFs. Herein, the processing time at each VNF is set by default of 50 ms in the simulator for all services. Therefore, the total link delay is the main factor that impacts the service response time of a SFC. Additionally, the total *hop count* from the user to the SFC, which is the number of intermediate nodes including switches between the UserVM and the data center can be analyzed. It shows insights into the dispersion of the SFC over different servers in the topology.

Resource Utilization

Resource utilization is an important metric, since reducing power consumption saves energy. In that way, additionally greenhouse gas emission is reduced, decreasing carbon footprint. To this end, the idea is to place all VMs on the smallest set of servers capable to deal with all running tasks. Empty servers can then be shut down to save energy. Based on this, we consider the *number of servers*, which are utilized to provide a given number of services.

4.3.5 Performance Evaluation of SFC Placement Algorithms

In this section, we present the simulation results that compare the performance of four SFC placement algorithms. We simulated three types of personalized services with individual algorithms in the extended EdgeNetworkCloudSim. Based on the topology and metrics presented above, we implement 20 replications for each algorithm to increase the statistical significance. After the experiments, we collect 400 log files in total and analyze them regarding the mentioned metrics. In the next subsections, we evaluate the performance of the algorithms according to two criteria, *service response time* and *resource utilization*. The algorithms under evaluation are *Centralization (CEN)*, *Orchestration (ORC)*, *Service Time Optimization (STO)*, and *Resource Optimization (RO)*. Wherein, *STO* and *RO* are $Objective_1$ and $Objective_2$ formulated in Equation (4.1) and Equation (4.2), respectively.

SFC Placement Algorithms vs. Service Response Time

Figure 4.7 shows the average service response time of the three SFCs with different placement algorithms. The x-axis indicates the three services, the y-axis shows the service response time in milliseconds. The bar group with different colors of each service shows the mean service response time with a $95\,\%$ confidence interval of different placement algorithms.

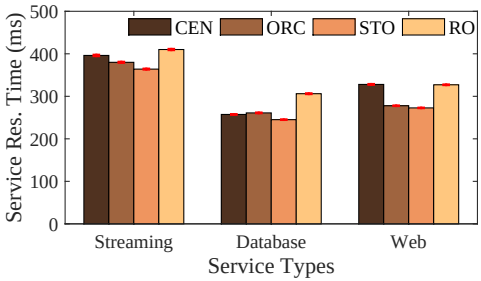


Figure 4.7: Average Service Response Time of Different SFCs

Figure 4.7 shows that the Streaming service has a higher service response time than the Database and the Web service. It is due to the fact that the Streaming service responds multiple video chunks per request, depending on the video length as described in Section 4.3.3. Each video chunk is processed at VNFs in the SFC before transferring to the user. The processing time at each VNF is $50\,\mathrm{ms}$ as default configuration in the simulator. Thus, the more video chunks delivered, the higher total processing time adding to the response time of the Streaming service. In fact, the average total response time of the Streaming service is about $6000\,\mathrm{ms}$. However, for the sake of comparability with other services, we only show in Figure 4.7 the average service response time of one reception. In other word, response time of one video chunk. The overhead of sending multiple video chunks increases the average response time of one chunk as shown in the figure. Conversely, the Database has the lowest average response time of about $250\,\mathrm{ms}$. Since it requires a lower total size of all VMs, the placement algorithms have a higher chance to place the VMs in a desired server which decreases the overall service response time.

Regarding the service response time produced by different placement algorithms, STO gains the lowest service response time of all services, followed by ORC, CEN, and RO. For instance, when STO is used as placement algorithm for

the Web service, it takes 272.80 ms for a user to request the service until receiving its response. Whereas, by using ORC, CEN, or RO algorithms, it takes 277.80 ms, 328.00 ms, and 327.10 ms, respectively. This does not come as a surprise, as STO was designed to compute a placement, minimizing the service response time by using the ILP model and considering the whole system state. Note that, since our topology has only four data centers, the processing time of CPLEX Optimizer is negligible. However, with a larger network topology, the solution space created by the CPLEX Optimizer is also large. As a consequence, the solving time for an optimal placement can negatively impact the overall service response time, since the optimal solution is a *NP-hard* problem.

Thus, heuristic approaches have to be investigated, such as ORC which reaches the second lowest service response time for Streaming and Web services. In this algorithm, all VMs in the chain are placed to have shortest distance between them. The first VM in the chain is placed as close as possible to the user. As a result, ORC constantly tries to minimize the length of the chain. In fact, ORC has to scan all servers with multiple loops to find the best placement for VMs. The chosen servers must have enough resources for all VMs as well. This operation is executed each time for a new coming service. Thus, with an increasing number of requests from the user, it also influences the overall service response time.

In fact, the ideal placement for the lowest service response time is the situation, where all VMs in the chain are placed in one server. In this case, the delay between the three VMs in the chain is zero, which substantially decreases the overall service response time. Figure 4.8 shows the probability of the occurrence of this situation. The x-axis indicates the three service types, the y-axis shows the probability of placing the whole chain in one server. The bars with different colors represent the mean probability for the different algorithms with a 95 % confidence interval.

It can be seen that RO exhibits a higher results compared to the other algorithms, since it is specially designed for resource optimization. This behavior is described in more detail in the next subsection. Regarding the other algorithms,

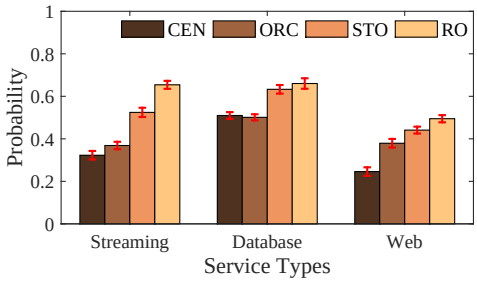


Figure 4.8: Probability of Placing the Complete Chain in one Server

Figure 4.8 shows that STO has a considerable higher value than ORC and CEN algorithms. In the Streaming service, there is $52.44\,\%$ chance that STO places the whole chain of the service in one server, while this number in case of ORC and CEN is $36.86\,\%$ and $32.29\,\%$, respectively. This tendency is also encountered in the Database and the Web services. This is reasonable, since STO calculates the SFC placement based on minimized total delay. Thereby, it is one option to place all VMs in one server. For instance, the selected server for the whole chain may not be the closest one to the user. Since the delay between the VMs in the chain is zero, the total delay is still smaller than the case where the VMs are distributed over different servers. This is the major difference between the optimized solution and the heuristics. Indeed, ORC and CEN choose the placement of a SFC based on iteratively scanning every server in data centers. They try to select the server as close as possible to the user. As a consequence, the closest servers are rapidly running out of resources. Afterwards, the VMs of the next SFC must be distributed over different servers. This is the reason why their probability of all VMs in one server is lower.

Figure 4.9 shows the mean hop count calculated from different placement algorithms. The x-axis shows the algorithms and the y-axis displays the mean hop count with a $95\,\%$ confidence interval.

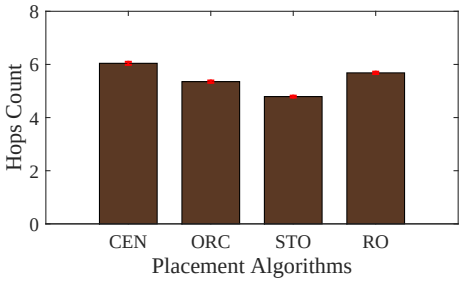


Figure 4.9: Average Hop Count of Different Placement Algorithms

The figure indicates that, CEN has the highest average hop count compared to other algorithms. The average number of intermediate nodes between the user and SFC calculated by CEN is 6.04, while in case of STO and ORC is 4.7 and 5.3, respectively. In contrast to ORC, the CEN algorithm always selects the closest server to the user to place VMs, without consideration of the length of the chain. Consequently, although the chosen servers are near to the user, the data flow might be transferred in a long chain that significantly increases the service response time. RO also shows a high hop count, only slightly lower than CEN. But still, the response time produced by RO is higher in the case of Streaming and Database services. This is reasonable, since RO is proposed to minimize resource utilization and service response time is not taken into account.

To conclude this subsection, we have shown that the STO placement strategy carried out by using ILP model has the highest performance, since it achieves the lowest service response time of all types. However, the performance of STO might be influenced by the processing time of CPLEX Optimizer in a larger topology. In this situation, the solving time for the optimization problem would be high and might considerably increase the overall service response time. The heuristic algorithm ORC shows an acceptable performance, since it attempts to minimize the length of the chain. Although RO has low performance in service

response time, it is particularly designed for resource optimization as presented in the next subsection.

SFC Placement Algorithms vs. Server Utilization

As presented in Section 4.2.3, the second objective of our ILP model is to specify the placement of a SFC, where resource utilization is minimized (i.e., RO algorithm). Figure 4.8 in the previous subsection indicates that RO has a noticeable higher probability that it places all VMs of a SFC in one server compared to the other algorithms. To this end, the objective function tries to minimize the number of servers used for SFCs regarding their available resources. This means, RO will place as much SFCs as possible in one server and has the preference to an already utilized server as well. By doing this, the unused servers can be put in the idle mode or shut down to save energy.

To evaluate the performance of this placement strategy, we calculate the number of utilized servers along with the number of concurrently instantiated SFCs. Herein, a server is considered as utilized when at least one CPU is allocated to a VM of a SFC. Figure 4.10 shows the correlation between the number of utilized servers and the number of concurrent services. The x-axis shows the number of concurrent services, the y-axis indicates the average number of corresponding used servers, meaning the Server Utilization (ServUtil) rate. The different colored lines display the mean ServUtil rate produced by different placement algorithms. The error bars on each line indicate the $95\,\%$ confidence interval of the mean values.

Figure 4.10 shows that STO, ORC, and CEN placement algorithms produce similar ServUtil rate, when all servers are handled by more than ten concurrently instantiated SFCs. This is reasonable, since these algorithms attempt to minimize the service response time by selecting the closest servers to the user. Since users are located at different *UserDCs* as shown in the topology, their nearest servers are quickly utilized. However, ORC, CEN, and STO are outnumbered by far by the RO placement algorithm, which has a much lower ServUtil rate as represented by the separate yellow line. It can be seen that ten concurrent services

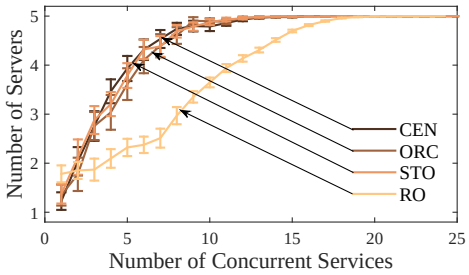


Figure 4.10: Number of Utilized Servers vs. Number of Concurrent Services

only use 3.6 servers on average. All servers are constantly utilized when there are 21 services instantiated at the same time. This maximum number of concurrent services doubles the other placement algorithms. This is due to the fact that the RO algorithm always prioritizes the collocation of VMs in one server before considering the others. Furthermore, when a SFC has finished the execution and releases server resources, but is still used by other services, it has a higher priority to be chosen for the next incoming service than the unused ones. Based on this, the number of utilized servers is minimized and the other servers can be put to standby state. This can significantly reduce the power consumption and save energy. To conclude, it could be seen that the presented heuristic algorithms show a decent performance in terms of service response time, but still need improvements in terms of resource utilization.

Influence of the Number of VNFs in a Chain on Service Response Time

A unique feature of a SFC in the NFV paradigm is enabling the operator to flexibly configure network services at the software level without changing the underlying hardware infrastructure. In such a configuration, a VNF can be involved in one or multiple SFCs depending on the service requirements. For instance, a virtual firewall can be part of both web and other value-added network services.

Besides, the flexible configuration also allows the operator to dynamically add or remove one or more constituent VNFs for a particular purpose. For instance, a virtual DPI may be added to a live streaming service for QoE monitoring.

Based on this, we consider another scenario where the Streaming service has a different number of constituent VNFs. We investigate the influence of this configuration on the service response time. In this subsection, we only consider the Streaming service and skip the result of the Web and Database service since they have similar behavior. Additionally, we only focus on the impact of the number of VNFs in the chain on the service response time. Therefore, we omit the measurement of the RO algorithm since it is particularly designed to minimize the server utilization as presented in the previous subsection.

Figure 4.11 shows the behavior of the response time of the Streaming service where the number of VNFs in the chain is varied. In the figure, the x-axis depicts the scenarios where, a SFC contains a different number of VNFs between 2 and 5. The y-axis indicates the mean service response time in each scenario with a 95 % confidence interval. In the figure, the service response time obtained by different placement algorithms is given as bars with different colors. Additionally, for each algorithm a linear regression of the service response time is performed, which is shown as a colored line, depending on the algorithm type.

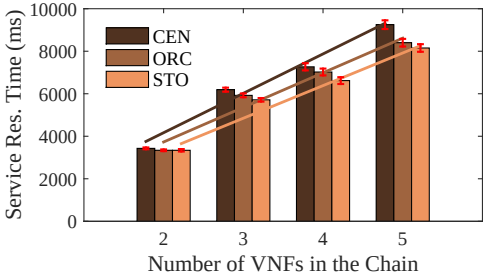


Figure 4.11: Influence of the Number of VNFs in Chain on Service Response Time

It can be seen from Figure 4.11 that the service response time of all placement algorithms linearly increases when the number of VNFs is incrementing. Specifically, with the CEN algorithm, the service response time increases by $45\,\%$ when the number of VNFs changes from 2 to 3 in the SFC. In case of ORC and STO, this number is $44\,\%$ and $42\,\%$, respectively. It is reasonable since a longer chain results in a higher end-to-end latency. Additionally, packets traversing the SFC will be processed at all VNFs in the chain. As a consequence, a SFC with more VNFs requires a longer processing time compared to a shorter one.

Regarding the performance of the algorithms, the service response time produced by CEN is higher compared to ORC and STO, as also described in the previous subsection. However, in the scenario with two VNFs in the chain, the difference in service response time between these placement algorithms is negligible. In fact, both CEN and ORC try to place the first VNFs as close as possible to the user. The only difference between them is the selection of the server for the second one. With ORC, it should be close to the first VNF, with CEN it will be close to the user. Consequently, the difference in service response time obtained by these two heuristic algorithms is small. Based on this, we believe that heuristic approaches are close to optimal if the number of VNFs in the chain is small enough. In this experiment, this number is two VNFs in chain.

Conversely, if the SFC contains more VNFs, the STO algorithm still gains a higher performance represented by a lower service response time. In addition to this, the velocity of an increase of service response time produced by CEN is faster than the others. Indeed, Table 4.5 shows the detailed results of the linear regression of the service response time depending on the number of VNFs in chain, including the Coefficient of Determination (CoD) r^2 as a measure for the goodness of fit. The table shows that CEN has the highest slope coefficient of 1853.14, with a smaller intercept of 49.12. This means, the service response time produced by CEN will raise faster with an increasing number of VNFs, compared to other algorithms. Nevertheless, both ORC and STO also produce a linear increase of service response time given by an incremental number of VNFs. This trend is represented by linear regression functions as shown in Ta-

Table 4.5: Linear Regression of the Service Response Time for Placement Algorithms

Algorithms	Service Response Time	r^2
CEN	$1853.14 \cdot x + 49.12$	0.97
ORC	$1628.08 \cdot x + 474.07$	0.96
STO	$1534.36 \cdot x + 586.16$	0.96

ble 4.5. With high CoD values, these functions can be reliably used as reference models for analyzing the impact of the number of VNFs in the chain on the service response time and also the QoE.

QoE Monitoring in the Context of SFC

In Chapter 3, we evaluate the influence of virtualization and geographical placement on the performance of the VNF for QoE monitoring. There, the VNF is a standalone monitoring function deployed in the AWS cloud. In this context, to efficiently deploy the VNF we can simply use CEN or ORC algorithm to find a closest server to the user. Thereby, we can achieve a high accuracy in video quality and QoE estimation. However, a QoE management architecture conventionally consists of different sub-functions. Figure 4.12 shows an example of a QoE management architecture for HAS. This architecture is also generally introduced in [21].

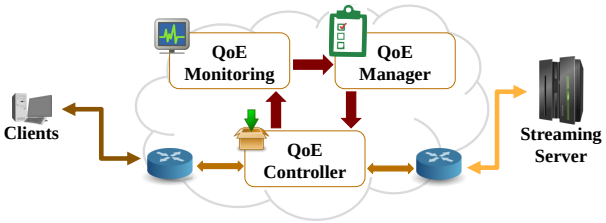


Figure 4.12: Overview of QoE Management Functions Chain

As shown in the figure, to perform QoE management, the *QoE Controller* acquires video traffic for the *QoE Monitoring*, where video quality and QoE are estimated based on a pre-defined QoE model. Thereafter, the estimates are forwarded to the *QoE Manager* with an awareness of the current network conditions. Finally, the *QoE Manager* triggers a network management decision into the operator network to improve the QoE accordingly. This architecture realizes a chain of QoE management functions where the network operator is the end point of the chain. We refer this concept is the QoE Monitoring SFC (QMSFC).

Regarding our monitoring approach in Chapter 3 for a single user, the placement problem of a QMSFC is easily solved since it only requires the lowest delay between the user and the monitoring point. In this case, the CEN algorithm can be used. However, in a multiple user scenario, the placement problem of the QMSFC must be well-defined to achieve a high efficiency in QoE management. Indeed, with an increasing number of users, more QMSFCs must be initiated across the network to handle an amplifying traffic load. In this situation, an efficient placement strategy for QMSFCs like ORC or STO is required to quickly deliver the management decisions into the operator network. By doing this, the operators have timely opportunity to react to improve the network and thereby to prevent user churn.

4.4 Lesson Learned

With the NFV paradigm, the use of SFCs is promising to reduce the complexity of heterogeneous services deployment. Nevertheless, the distribution of VNFs over different hosts increase the overall latency and server utilization rate. In this chapter, we evaluate four algorithms to efficiently place SFCs in the context of an edge network. These algorithms aim to decrease the service response time or minimize resource utilization. To evaluate and compare the performance of these placement algorithms, we use the event-based EdgeNetworkCloudSim simulator. Thereby, data centers are located in an edge cloud and the VNFs of the SFC are placed in the servers of a data center according to a specific algorithm.

While the CEN and ORC algorithms place the VNFs by heuristically scanning servers for proper locations, STO and RO are optimized solutions obtained by solving an ILP model.

Regarding the service response time, the result shows that STO performs better than the other algorithms in all types of services. This demonstrates that the use of an ILP model is able to compute an optimal solution. Especially, the probability of placing all VMs of a chain in one server is higher than with the CEN and ORC algorithms, which results in reduced service response time. However, the processing time of the optimizer is a considerable drawback as the placement problem is *NP-hard*. Thus, computation times might be unusably high when the topology is large with a huge number of servers and links connecting them.

Despite of producing higher service response times than STO, the ORC algorithm always tries to shorten the length of the chain. This algorithm can be an alternative method for STO in a large network topology, if the processing time of STO is high. However, as ORC iteratively scans all servers for the closest possible placement of a VM, the scanning time can increase in large topologies. The CEN algorithm produces higher overall service response time than STO and ORC, since it only places VMs as close as possible to the user without the consideration of the chain itself and the communication of the VMs.

In the context of QoE management, the service response time plays an important role since the management decisions must be delivered to the network operators in time. To achieve the goal, the VNFs in the QMSFC like QoE monitoring or QoE controller must be efficiently placed across the network to minimize overall response time to the network operators. For instance, the QoE controller should be placed close to the user to reduce the delay in estimating the video quality. Whereas, the QoE monitoring or QoE manager should be placed close to the network operator to quickly deliver management decisions. In this situation, an optimal placement is required to minimize the response time of the QoE manager into the operator networks. To this end, the STO algorithm is recommended. Nevertheless, to efficiently place the VNFs of the QoE management service using STO, the solving time of the ILP model must be taken into account.

The second objective of the ILP model is to minimize the server utilization. Herein, the placement of the SFC is optimized to utilize the least number of servers. Our result shows that with the optimized placement, ten concurrent SFCs only require half of all server resources. In contrast, the heuristics and STO utilize all data centers to handle the same number of concurrent SFCs. This insight shows that while the evaluated heuristics perform well in terms of service response times, they require improvements to have the ability to reduce power consumption, cost, and thus the carbon footprint of data centers.

Lastly, we extend the simulation to other scenarios, where a SFC consists of different number of VNFs. Our result shows that with both heuristic and optimized placement strategies, the service response time linearly varies with an increasing number of VNFs in the chain. This behavior is represented by a linear regression functions with high CoD values. These equations can later be used as analytical models for optimizing the service response time with respect to the number of VNFs in a chain.

To conclude, the findings in this chapter provide the network operators with different placement strategies, where they can decide to place the SFC in order to reduce the end to end latency or to save energy.

5 Conclusion

The Internet has exponentially evolved in the last decade. In particular, key developments in the network technology have led to an improvement in the user experience while using Internet services. There, one of the most important changes is the emergence of cloud computing. This technology is an efficient alternative for desktop-based software that is statically installed on the user machine. Nowadays, users are able to access an arbitrary type of cloud application from anywhere with a thin client. Thus, the benefits of the cloud paradigm have attracted a huge number of subscriptions in recent years. However, high service demands also challenge the network providers maintaining the user expectation since their resource capacity is limited. To cope with this problem, it is important to be aware of the user perception when an impairment of the service quality occurs. Based on this, traffic management can be performed in a timely manner to improve the network QoS and the service quality as well. With the establishment of the Quality of Experience (QoE) concept, the network providers have an ability to understand the user experience through a monitoring mechanism in the network.

This monograph has focused on different aspects of QoE research. From the study of QoE assessment of the prominent cloud applications, objective QoE monitoring was investigated for HTTP Adaptive Video Streaming (HAS). Additionally, by using Virtual Network Function (VNF) for QoE monitoring in the cloud, the feasibility of an interworking of Network Function Virtualization (NFV) and cloud paradigm was evaluated. In the NFV architecture, the implementation of a service chain is promising to decrease the complexity when deploying heterogeneous network services. Thus, different placement algorithms

of Service Function Chain (SFC) were proposed and compared in the context of the edge cloud. On the one hand, this research work targets the cloud service providers by providing them with methods of assessing QoE for cloud applications. On the other hand, the network operators can learn the pitfalls and drawbacks of applying the NFV paradigm for such a QoE monitoring mechanism. In the following, we summarize the main outcomes of this work and recommendations for the providers.

In the first part, we investigated the popular cloud services, which fall into two models of multi-tenancy cloud architecture. While Google Docs is a shared multiple user cloud application, a cloud-based photo service features an isolated multi-tenancy model where each user has his own album. Each service has different characteristics in performance with respect to QoE. Thus, for Google Docs, we considered the impact of delay and packet loss on different subprocesses such as login or creating document. To this end, we used a practical emulation testbed where the network is artificially changed according to pre-defined traces. Our results show that network delay differently influences the login, creating, and saving a document subprocesses, while low packet loss rate does not affect at all. As part of the outcome, the derived linear regression models in turn can later be used in analytical model for optimization or in a reference model for QoE monitoring.

Different from Google Docs, a cloud-based photo service typically provides the user with an isolated space to store pictures. Thereby, the whole photo album is able to be moved next to the user to decrease the latency due to a long distance access. Thus, QoE assessment for this type of service is necessary to determine when the photo loading time degrades the user perceived QoE. To this end, we considered the trade-off between the content size, geographical location of the service, and QoE. In this study, we validated and used a previous TCP throughput model to calculate the loading time of a photo given by different network QoS parameters and distance between the user and the service. Then, the TCP model is mapped with a QoE model to achieve the mentioned goal. We observed that the presence of delay and packet loss dramatically reduces

the QoE for photo loading time, while without packet loss, a long distance only slightly decreases the QoE. Based on the mapping model, the service and network operators can optionally adjust the photo size, improve the network QoS, or give decision to migrate the service next to the user to improve the QoE.

Second, we focused on one of the most popular and demanding services in today's Internet, HAS. Based on the previous studies about the influence factors of QoE for HAS, we designed a monitoring function exploiting Deep Packet Inspection (DPI) technique in the form of a VNF. This function is able to extract application layer parameters of the HAS such as video buffer and quality adaptation, stalling frequency, and length. Thereby, the video quality is observed and QoE perceived by the user is estimated using a pre-defined reference model. Considering the side-effects of the network QoS and the virtual environment on the accuracy of the estimation, we deployed the VNF QoE monitoring in different scenarios, either in a practical emulation testbed and on the Amazon Web Service cloud. We found out that the VNF gains a high accuracy in estimating video quality and QoE for HAS when it is deployed next to the user or operated with a high bandwidth connection. Conversely, there is a time shifting in estimating the video buffer if a high latency is present on the network path. Specially, packet re-ordering and mobile access network have negative influence on the accuracy of the monitoring function when it is placed far away from the user. This network behavior induces an overestimation of the QoE that leads to a misunderstanding for the network providers about the user expectation. Therefore, it is highly recommended for the network providers to properly place such a VNF monitoring to achieve a high accuracy in estimating QoE. In our study, it should be placed at the user device or in the edge cloud.

The third part of the thesis carries out a study about service function chaining in the edge cloud. In a SFC, separated VNFs are placed in a specific order to perform the service. For instance, in the QoE management system, QoE controller, QoE monitoring, and QoE manager must be executed in sequence to manage the network and to maintain the expected QoE perceived by the end user. Thus, it is necessary to have a good strategy of placing the VNFs in the SFC to obtain a low

end to end latency. Other concern is to minimize the server utilization hosting the VNFs. This contributes to the reduction in power consumption and therefore decrease carbon footprint. To this end, we proposed four placement algorithms for SFC with respect to minimizing service response time and server utilization. Wherein two heuristic approaches are designed against the optimized solution obtained by using the Integer Linear Programing (ILP) model. The SFC is simulated in the EdgeNetworkCloudSim simulator software and the algorithms are installed to place the VNFs whenever a service is requested. Our simulation results show that the optimized solutions produce lowest service response time and least server utilization rate compared to the heuristic approaches. This demonstrates that the ILP model can be used to efficiently compute an optimal placement for the SFC. However, one must consider the size of solution space when the network topology is large. This can become a bottleneck when searching for an optimal solution since the placement problem is *NP-hard*. In this situation, the heuristic algorithms can be the alternatives since they have simple placing rules that require less processing time. These insights may help the network operators to learn the strengths and weaknesses of different placement strategies. Based on this, they have an ability to quickly compute a proper placement for the SFC in the context of the edge cloud.

To summarize, this monograph covers different experimental methods for the QoE assessment and monitoring for cloud services in both fixed and mobile networks. These methodologies are not limited in the scope of this thesis but can later be applied for other researches in the field of QoE assessment and monitoring. This work provides the cloud service providers with different QoE assessment methods through the two exemplary cloud applications. Based on this, they can assess the performance of other cloud services with respect to QoE perceived by the user. Regarding the QoE monitoring, the network operators can learn the influence factors from this work on the deployment of the VNF for QoE monitoring in the cloud. There, to achieve an accurate QoE estimation, the side-effects of the virtual environment, network QoS, and geographical placement must be taken into account. Finally, we propose and evaluate the performance

of four placement strategies for SFC with respect to service response time or server utilization. By considering the characteristics of these algorithms, the network operators can decide whether placing their SFCs to reduce the end to end delay or to save energy in the context of the edge cloud.

Today's Internet is continuously changing its shape with the emergence of new network technologies. The advent of the NFV paradigm inspires the operators to transform the legacy network into a more flexible, scalable, and low cost approach. Thereby, a new concept of intelligent networks has established for the future Internet. Nevertheless, the performance of the NFV architecture is still on the early stage of the evaluation. Thus, more efforts in future research are required to actualize and commercialize this network paradigm also in combination with other paradigms like QoE monitoring.

Acronyms

ADSL Asymmetric Digital Subscriber Line. 45

API Application Programming Interface. 64, 81

AWS Amazon Web Service. 53, 54, 61, 77–84, 87, 90, 92, 93, 96, 97, 99, 137

BDP Bandwidth Delay Product. 41, 42

CAPEX Capital Expenditure. 102, 104, 106, 107, 113

CDF Cumulative Distribution Function. 42, 73, 75, 89, 97

CDN Content Delivery Network. 14, 81, 93

CEN Centralization. 114–118, 128–133, 136–139

CoD Coefficient of Determination. 27, 29, 32, 34, 136, 137, 140

CPS Citrix Presentation Server. 19

DASH Dynamic Adaptive Streaming over HTTP. 1

DB Database. 110

DC Data Center. 63, 69, 71–76, 99, 115–117, 125, 126

DPDK Data Plane Development Kit. 107

DPI Deep Packet Inspection. 59, 60, 77, 98, 101, 105, 107, 135, 143

EM Element Management. 108

EPC Evolved Packet Core. 108

ES Edge Server. 63, 68, 72–76, 99

ETSI European Telecommunications Standards Institute. 78, 79, 109

EU-PoP Europe Point of Presence. 89, 91, 93, 94, 96, 97, 99

GUI Graphical User Interface. 112

HAS HTTP Adaptive Video Streaming. 3–6, 9, 18, 51, 54, 55, 57, 59, 60, 64, 70, 100, 137, 141, 143

HTML Hypertext Markup Language. 19

HTTP Hyper Text Transfer Protocol. 17, 34, 54, 65

HTTPs HTTP Over TLS. 17, 34

IaaS Infrastructure as a Service. 80

IETF Internet Engineering Task Force. 110, 112

ILP Integer Linear Programing. 7, 9, 103, 113–115, 117, 119, 123, 130, 132, 133, 139, 140, 144

IMS IP Multimedia Subsystem. 61

IP Internet Protocol. 65

IPTV Internet Protocol Television. 56

ISG Industry Specification Group. 110

LTE Long Term Evolution. 81, 84, 85

MANO Management and Orchestration. 78, 79, 108, 109

MIQCP Mixed Integer Quadratically Constrained Program. 113

MME Mobility Management Entity. 108

MOS Mean Opinion Score. 18, 35, 42, 44–46, 49, 55, 74–76, 92–94, 99, 100

MPD Media Presentation Description. 54, 59, 65, 66, 83

NAT Network Address Translation. 109

NFV Network Function Virtualization. 3–6, 9, 51–54, 61, 62, 77–80, 100–102, 104–109, 112–114, 134, 138, 141, 142, 145

NFVI NFV Infrastructure. 108, 109

NFVO NFV Orchestrator. 109

NTP Network Time Protocol. 69, 84

OPEX Operational Expenditure. 102, 104, 106, 107, 113

OPL Optimization Programming Language. 103

ORC Orchestration. 114–118, 128–133, 136–139

PC Personal Computer. 22, 23, 25, 68

PoP Point of Presence. 4, 52–54, 77, 81, 82, 84, 87, 89–91, 96

PSQA Pseudo Subjective Quality Assessment. 59

QMSFC QoE Monitoring SFC. 138, 139

QoE Quality of Experience. 2–7, 9, 12–18, 20, 34–36, 42–49, 51–60, 62–64, 66, 74–78, 80, 81, 83, 84, 86, 91–96, 98–106, 111, 135, 137–139, 141–145

QoS Quality of Service. 2–4, 6, 7, 12, 14, 15, 17, 18, 34–36, 38, 42, 44, 46–49, 53, 64, 84, 106, 141–144

RDP Remote Desktop Protocol. 19

RFC Request for Comments. 39

RMSE Root Mean Square Error. 73, 75, 89, 97

RO Resource Optimization. 114, 115, 119, 128–130, 132–135, 139

RTT Round Trip Time. 36, 37, 39, 41, 42, 82, 89, 90, 96, 100

SaaS Software as a Service. 1, 7, 11, 14–17, 19, 47, 48

SDN Software-Defined Networking. 61, 62, 105

ServUtil Server Utilization. 133

SF Service Function. 61

SFC Service Function Chain. 4–7, 9, 102–104, 110–118, 122–125, 127–129, 131–136, 138, 140, 142–145

SGW Serving Gateway. 108

SSH Secure Shell. 38, 84

STO Service Time Optimization. 114, 115, 119, 128–133, 136, 138–140

STR Video Streaming. 110

TCP Transmission Control Protocol. 6, 17, 34–44, 48, 65, 90, 142

US-PoP United States of America Point of Presence. 87, 89, 91–95, 97–100

vDPI virtual Deep Packet Inspection. 62, 114

VIM Virtualized Infrastructure Manager. 109

VM Virtual Machine. 108–111, 115–117, 120, 123–126, 128–134, 139

VNF Virtual Network Function. 4–7, 9, 52–54, 57, 60–62, 64, 66, 68, 72, 73, 75–84, 86, 87, 89–111, 113, 114, 127, 129, 134–141, 143, 144

VNFG VNF Forwarding Graph. 109, 110

VNFM VNF Manager. 109

VoIP Voice over IP. 43

VPC Virtual Private Cloud. 80

WAN Wide Area Network. 19, 61

Bibliography and References

— Bibliography of the Author —

Journal Article

[1] V. Burger, T. Zinner, L. Dinh-Xuan, F. Wamser, and P. Tran-Gia, "A generic approach to video buffer modeling using discrete-time analysis", *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 2018.

Conference Papers

[2] F. Wamser, C. Lombardo, C. Vassilakis, L. Dinh-Xuan, P. Lago, R. Bruschi, and P. Tran-Gia, "Orchestration and monitoring in fog computing for personal edge cloud service support", in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, Washington, DC, USA, Jun. 2018.

[3] L. Dinh-Xuan, M. Seufert, F. Wamser, and P. Tran-Gia, "Qoe aware placement of content in edge networks on the example of a photo album cloud service", in *IEEE 6th International Conference on Communications and Electronics (ICCE)*, Ha Long, Vietnam, Jul. 2016.

[4] L. Dinh-Xuan, M. Seufert, F. Wamser, and P. Tran-Gia, "Study on the accuracy of qoe monitoring for http adaptive video streaming using vnf", in *1st IFIP/IEEE International Workshop on Quality of Experience Management (QoE-Management)*, Lisbon, Portugal, May 2017.

[5] L. Dinh-Xuan, C. Schwartz, M. Hirth, F. Wamser, and H. Truong Thu, "Analyzing the impact of delay and packet loss on google docs", in *7th International Conference on Mobile Networks and Management*, Santander, Spain, Sep. 2015.

[6] L. Dinh-Xuan, M. Seufert, F. Wamser, C. Vassilakis, A. Zafeiropoulos, and P. Tran-Gia, "Performance evaluation of service functions chain placement algorithms in edge cloud", in *30th International Teletraffic Congress (ITC30)*, Vienna, Austria, Sep. 2018.

— General References —

[7] ISO/IEC 23009-1:2012, *Information technology - dynamic adaptive streaming over http (dash) - part 1: Media presentation description and segment formats*, 2012.

[8] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, "Everything as a service (xaas) on the cloud: Origins, current and future trends", in *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, IEEE, New York, NY, USA, Jun. 2015, pp. 621–628.

[9] A. Ahmad, A. Floris, and L. Atzori, "Qoe-centric service delivery: A collaborative approach among otts and isps", *Computer Networks*, vol. 110, pp. 168–179, 2016.

[10] B.-H. Chu, M.-S. Tsai, and C.-S. Ho, "Toward a hybrid data mining model for customer retention", *Knowledge-Based Systems*, vol. 20, no. 8, pp. 703–718, 2007.

[11] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service", *IEEE Network Special Issue on Improving QoE for Network Services*, Jun. 2010.

[12] P. Le Callet, S. Möller, A. Perkis, *et al.*, "Qualinet white paper on definitions of quality of experience", *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, vol. 3, 2012.

[13] A. Karadimce and D. Davcev, "Perception of quality in cloud computing based services", in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, IEEE, Lisbon, Portugal, Jun. 2016, pp. 1–6.

[14] S. Egger, T. Hoßfeld, R. Schatz, and M. Fiedler, "Waiting times in quality of experience for web based services", in *Fourth International Workshop on Quality of Multimedia Experience (QoMEX)*, IEEE, Melbourne, Australia, Jul. 2012.

[15] S. Egger, P. Reichl, T. Hoßfeld, and R. Schatz, "Time is Bandwidth? Narrowing the Gap between Subjective Time Perception and Quality of Experience", in *2012 IEEE International Conference on Communications (ICC)*, Ottawa, Canada, Jun. 2012.

[16] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing", in *IEEE International Symposium on Multimedia (ISM2011)*, IEEE, California, USA, Dec. 2011.

[17] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming", *IEEE Communications Surveys and Tutorials*, vol. 17, Mar. 2015.

[18] ITU-T Recommendation J.144, "Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference", *International Telecommunication Union, Tech. Rep*, Mar. 2004.

[19] Cisco Systems, *Cisco visual networking index: Forecast and methodology, 2015-2020*, White Paper, 2016.

[20] M. Chios, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, and H. Deng, *Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action*, White Paper available at `https://portal.etsi.org/nfv/nfv_white_paper.pdf`, 2012.

[21] E. Liotou, D. Tsolkas, K. Samdanis, N. Passas, and L. Merakos, "Towards quality of experience management in the next generation of mobile networks", in *European Conference on Networks and Communications (EuCNC)*, Athens, Greece, Jun. 2016.

[22] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture", RFC, Tech. Rep. 7665, Oct. 2015.

[23] R. Bruschi, P. Lago, and C. Lombardo, "In-network programmability for next-generation personal cloud service support (input)", *Procedia Computer Science*, vol. 97, pp. 114–117, 2016.

[24] M. Seufert, B. K. Kwam, F. Wamser, and P. Tran-Gia, "Edgenetworkcloudsim: Placement of service chains in edge clouds using networkcloudsim", in *IEEE Conference on Network Softwarization (NetSoft 2017)*, IEEE, Bologna, Italy, Jul. 2017, pp. 1–6.

[25] U. Reiter, K. Brunnström, K. De Moor, M.-C. Larabi, M. Pereira, A. Pinheiro, J. You, and A. Zgank, "Factors influencing quality of experience", in *Quality of experience*, Springer International Publishing, 2014, pp. 55–72, ISBN: 978-3-319-02680-0.

[26] ITU-T Recommendation G.1030, "Estimating end-to-end performance in ip networks for data applications", *International Telecommunication Union*, Nov. 2005.

[27] G. Bertrand, E. Stephan, T. Burbridge, P. Eardley, and K. M. G. Watson, "Use cases for content delivery network interconnection", RFC 6770, Nov. 2012.

[28] M. Reiners and W. van der Bijl, “Content delivery networks”, *Communications Network*, vol. 3, no. 3, 2004.

[29] ITU-R Recommendation P.800, “Methods for subjective determination of transmission quality”, *International Telecommunication Union*, Aug. 1996.

[30] R. Jain, “Quality of experience”, *IEEE MultiMedia*, Jan. 2004.

[31] R. K. Mok, E. W. Chan, and R. K. Chang, “Measuring the quality of experience of http video streaming”, in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Dublin, Ireland, May 2011.

[32] P. Casas and R. Schatz, “Quality of experience in cloud services: Survey and measurements”, *Computer Networks*, vol. 68, pp. 149–165, 2014.

[33] H. Dan, “Google this!: Using google apps for collaboration and productivity”, in *SIGUCCS Fall Conference*, St. Louis, USA, Oct. 2009.

[34] D. Schlosser, A. Binzenhöfer, and B. Staehle, “Performance comparison of windows-based thin-client architectures”, in *ATNAC 2007*, Christchurch, New Zealand, Dec. 2007.

[35] D. Schlosser, B. Staehle, A. Binzenhöfer, and B. Boder, “Improving the qoe of citrix thin client users”, in *International Conference on Communications*, South Africa, May 2010.

[36] A. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, *et al.*, “OPTIMIS: A Holistic Approach to Cloud Service Provisioning”, *Future Generation Computer Systems*, vol. 28, 2012.

[37] P. Amrehn, K. Vandenbroucke, T. Hoßfeld, K. de Moor, M. Hirth, R. Schatz, and P. Casas, “Need for Speed? On Quality of Experience for File Storage Services”, in *Workshop on Perceptual Quality of Systems*, Vienna, Austria, Sep. 2013.

[38] G. Reig and J. Guitart, "On the anticipation of resource demands to fulfill the qos of saas web applications", in *Conference on Grid Computing*, Beijing, China, Sep. 2012.

[39] T. Hoßfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE Management for Cloud Applications", *Communications Magazine, IEEE*, vol. 50, no. 4, pp. 28–36, 2012.

[40] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "An Evaluation of QoE in Cloud Gaming Based on Subjective Tests", in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, IEEE, 2011, pp. 330–335.

[41] B. Emmert, A. Binzenhoefer, D. Schlosser, and M. Weiss, "Source traffic characterization for thin client based office applications", in *Workshop on Dependable and Adaptable Networks and Services*, Twente, Netherlands, Jul. 2007.

[42] O3B Networks, *What is network latency and why does it matter?*, White Paper available at `http://tinyurl.com/nv8agu8`, Nov. 2008.

[43] H. Ningning, L. Li, M. Z. Morley, S. Peter, and W. Jia, "A measurement study of internet bottlenecks", in *INFOCOM*, Miami, Florida, USA, Mar. 2005.

[44] S. Anmol, N. Sergiu, P. Rabin, S. Sonesh, B. Eric, and S. Lakshminarayanan, "Packet loss characterization in wifi-based long distance networks", in *International Conference on Computer Communications*, Anchorage, Alaska, USA, May 2007.

[45] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation", in *ACM SIGCOMM Computer Communication Review*, 1998.

[46] W. Bao, V. W. Wong, and V. Leung, "A model for steady state throughput of tcp cubic", in *Global Telecommunications Conference (GLOBECOM)*, Florida, USA, Dec. 2010.

[47] S. Poojary and V. Sharma, “Analytical model for congestion control and throughput with tcp cubic connections”, in *Global Telecommunications Conference (GLOBECOM)*, Texas, USA, Dec. 2011.

[48] J. Balej, O. Komnek, and M. Rajnoha, “Geographic distance estimation for ip geolocation”, in *Proceedings in EIIC-The 2nd Electronic International Interdisciplinary Conference*, Sep. 2013.

[49] L. Q. Chen, X. Xie, X. Fan, W. Y. Ma, H. J. Zhang, and H. Q. Zhou, “A visual attention model for adapting images on small displays”, *Multimedia Systems*, vol. 9, no. 4, pp. 353–364, 2003.

[50] R. Mohan, J. R. Smith, and C. S. Li, “Adapting multimedia internet content for universal access”, *Multimedia, IEEE Transactions on*, vol. 1, no. 1, pp. 104–114, 1999.

[51] S. Hemminger, “Network emulation with netem”, in *Australia’s National Linux conference*, Canberra, Australia, Apr. 2005.

[52] ITU-T Recommendation G.114, “One-way transmission time”, *International Telecommunication Union*, May 2000.

[53] ETSI-TR-101-329-6, *Actual measurements of network and terminal characteristics and performance parameters in tiphon networks and their influence on voice quality*, Jul. 2001.

[54] V. Paxson, M. Allman, J. Chu, and M. Sargent, “Computing tcp’s retransmission timer”, RFC 6298, Jun. 2011.

[55] R. Braden and V. Jacobson, “Tcp extensions for long-delay paths”, RFC 1072, Oct. 1988.

[56] J. Shaikh, M. Fiedler, and D. Collange, “Quality of experience from user and network perspectives”, *Annals of Telecommunications-Annales des Télécommunications*, vol. 65, no. 1-2, pp. 47–57, 2010.

[57] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges”, *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, 2015.

[58] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations", *IEEE Communications Magazine*, vol. 53, no. 2, 2015.

[59] ETSI-GS-NFV-002, *Network functions virtualisation (nfv); architectural framework*, Group Specification available at `http://www.etsi.org`, 2014.

[60] Cisco Systems, *Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021*, White Paper, 2017.

[61] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in youtube: From traffic measurements to quality of experience", in *Data Traffic Monitoring and Analysis. Lecture Notes in Computer Science, vol 7754*, Springer, Berlin, Heidelberg, 2013, pp. 264–301.

[62] B. Staehle, M. Hirth, F. Wamser, R. Pries, and D. Staehle, "Yomo: A youtube application comfort monitoring tool", University of Wuerzburg, Tech. Rep. 467, Mar. 2010.

[63] B.Staehle, M. Hirth, R. Pries, F. Wamser, and D. Staehle, "Yomo: A youtube application comfort monitoring tool", Jun. 2010.

[64] T. Stockhammer, "Dynamic adaptive streaming over http: Standards and design principles", in *Proceedings of the Second annual ACM Conference on Multimedia Systems*, ACM, Santa Clara, CA, USA, Feb. 2011.

[65] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hobfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming", *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[66] ITU-T Recommendation P. 910, "Subjective video quality assessment methods for multimedia applications", *International Telecommunication Union, Tech. Rep*, 2008.

[67] ITU-T Recommendation P.913, "Methods for the subjective assessment of video quality, audio quality and audiovisual quality of internet video and distribution quality television in any environment", *International Telecommunication Union, Tech. Rep*, 2016.

[68] M. H. Pinson and S. Wolf, "Comparing subjective video quality testing methodologies", in *Visual Communications and Image Processing 2003*, International Society for Optics and Photonics, 2003, pp. 573–582.

[69] ITU-T Recommendation P.10/G.100, "Vocabulary for performance and quality of service", *International Telecommunication Union, Tech. Rep*, 2006.

[70] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best practices for qoe crowdtesting: Qoe assessment with crowdsourcing", *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 541–558, 2014.

[71] J. Howe, "The rise of crowdsourcing", *Wired Magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[72] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service", *IEEE Network*, vol. 24, no. 2, 2010.

[73] A. Takahashi, D. Hands, and V. Barriac, "Standardization activities in the itu for a qoe assessment of iptv", *IEEE Communications Magazine*, vol. 46, no. 2, 2008.

[74] D. De Vera, P. Rodriguez-Bocca, and G. Rubino, "Qoe monitoring platform for video delivery networks", in *International Workshop on IP Operations and Management*, Springer, San José, USA, Nov. 2007, pp. 131–142.

[75] K. Piamrat, C. Viho, J.-M. Bonnin, and A. Ksentini, "Quality of experience measurements for video streaming over wireless networks", in *Sixth International Conference on Information Technology: New Generations*, IEEE, Las Vegas, NV, USA, Apr. 2009, pp. 1184–1189.

[76] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "Youtube qoe estimation based on the analysis of encrypted network traffic using machine learning", in *Globecom Workshops (GC Workshops)*, IEEE, Washington, DC, USA, Dec. 2016, pp. 1–6.

[77] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks", in *European Conference on Networks and Communications (EuCNC)*, Paris, France, Jun. 2015.

[78] F. Wamser, P. Casas, M. Seufert, C. Moldovan, P. Tran-Gia, and T. Hoßfeld, "Modeling the youtube stack: From packets to quality of experience", *Computer Networks*, Dec. 2016.

[79] R. Schatz, T. Hoßfeld, and P. Casas, "Passive youtube qoe monitoring for isps", in *The Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012)*, IEEE, Palermo, Italy, Jul. 2012.

[80] P. Casas, R. Schatz, and T. Hoßfeld, "Monitoring youtube qoe: Is your mobile network delivering the right experience to your customers?", in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, Shanghai, China, Apr. 2013.

[81] P. Casas, M. Seufert, and R. Schatz, "Youqmon: A system for on-line monitoring of youtube qoe in operational 3g networks", *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, pp. 44–46, Sep. 2013.

[82] J. Yao, S. S. Kanhere, I. Hossain, and M. Hassan, "Empirical evaluation of http adaptive streaming under vehicular mobility", in *10th International*

IFIP TC 6 Networking Conference, Springer, Valencia, Spain, May 2011, pp. 92–105.

[83] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over http in vehicular environments", in *Proceedings of the 4th Workshop on Mobile Video*, ACM, Chapel Hill, NC, USA, Feb. 2012, pp. 37–42.

[84] M. Seufert, F. Wamser, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "Youtube qoe on mobile devices: Subjective analysis of classical vs. adaptive video streaming", in *6th International Workshop on Traffic Analysis and Characterization (TRAC)*, Dubrovnik, Croatia, Aug. 2015.

[85] P. M. Eittenberger, M. Hamatschek, M. Großmann, and U. R. Krieger, "Monitoring mobile video delivery to android devices", in *Proceedings of the 4th ACM Multimedia Systems Conference*, ACM, Oslo, Norway, Feb. 2013, pp. 119–124.

[86] S. Oechsner and A. Ripke, "Flexible support of vnf placement functions in openstack", in *1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, London, UK, Apr. 2015, pp. 1–6.

[87] G. Carella, M. Corici, P. Crosta, P. Comi, T. M. Bohnert, A. A. Corici, D. Vingarzan, and T. Magedanz, "Cloudified ip multimedia subsystem (ims) for network function virtualization (nfv)-based architectures", in *IEEE Symposium on Computers and Communication (ISCC)*, IEEE, Madeira, Portugal, Jun. 2014, pp. 1–6.

[88] J. Soares, C. Gonçalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions", *IEEE Communications Magazine*, vol. 53, no. 2, pp. 98–106, 2015.

[89] R. Yu, G. Xue, V. T. Kilari, and X. Zhang, "Network function virtualization in the multi-tenant cloud", *IEEE Network*, vol. 29, no. 3, pp. 42–47, 2015.

[90] S. Clayman, E. Maini, A. Galis, A. Manzalini, and N. Mazzocca, "The dynamic placement of virtual network functions", in *2014 Network Operations and Management Symposium (NOMS)*, IEEE, Krakow, Poland, May 2014.

[91] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn", in *MILCOM 2013 IEEE Military Communications Conference*, IEEE, San Diego, CA, USA, Nov. 2013.

[92] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures", *International Journal of Network Management*, vol. 25, no. 6, 2015.

[93] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges", *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[94] K. C. Leung and D. Yang, "An overview of packet reordering in transmission control protocol (tcp): Problems, solutions, and challenges", *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, 2007.

[95] C. Gao, Z. Ling, and Y. Yuan, "Packet reordering analysis for concurrent multipath transfer", *International Journal of Communication Systems*, vol. 27, no. 12, 2014.

[96] D. Kaspar, K. Evensen, A. F. Hansen, P. Engelstad, P. Halvorsen, and C. Griwodz, "An analysis of the heterogeneity and ip packet reordering over multiple wireless networks", in *IEEE Symposium on Computers and Communications(ISCC)*, IEEE, 2009.

[97] ETSI-GS-NFV, "Network functions virtualisation (nfv): Architectural framework", *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.

[98] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment", *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[99] J. Varia and S. Mathew, *Overview of amazon web services*, White Paper available at `https://media.amazonwebservices.com/AWS_Overview.pdf`, Amazon Web Services, 2014.

[100] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks", *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[101] M. Allman, V. Paxson, and E. Blanton, "Tcp congestion control", RFC Editor, Tech. Rep. 5681, Sep. 2009.

[102] D. E. Knuth, "Postscript about np-hard problems", *ACM SIGACT News*, vol. 6, no. 2, pp. 15–16, 1974.

[103] D.-Z. Du and K.-I. Ko, *Theory of computational complexity*. John Wiley & Sons, 2011, vol. 58.

[104] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization", in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*, IEEE, San Francisco, CA, USA, Jan. 2015.

[105] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions", in *11th International Conference on Network and Service Management*, IEEE, Barcelona, Spain, Nov. 2015.

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

[107] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions", in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, IEEE, Luxembourg, Dec. 2014.

[108] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, *et al.*, "Steering: A software-defined networking for inline service chaining", in *21st IEEE International Conference on Network Protocols (ICNP)*, IEEE, Goettingen, Germany, Oct. 2013, pp. 1–10.

[109] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard, "Energy-efficient service function chain provisioning", *Journal of Optical Communications and Networking*, vol. 10, no. 3, pp. 114–124, 2018.

[110] C.-H. Hsieh, J.-W. Chang, C. Chen, and S.-H. Lu, "Network-aware service function chaining placement in a data center", in *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, Kanazawa, Japan, Oct. 2016, pp. 1–6.

[111] E. Masanet, A. Shehabi, J. Liang, L. Ramakrishnan, X. Ma, V. Hendrix, B. Walker, and P. Mantha, "The energy efficiency potential of cloud-based software: A us case study", Ernest Orlando Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA (United States), Tech. Rep., 2013.

[112] I. Chih-Lin, J. Huang, R. Duan, C. Cui, J. X. Jiang, and L. Li, "Recent progress on c-ran centralization and cloudification", *IEEE Access*, vol. 2, pp. 1030–1039, 2014.

[113] J. Martins, M. Ahmed, C. Raiciu, V. Olteanu, M. Honda, R. Bifulco, and F. Huici, "Clickos and the art of network function virtualization", in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, USENIX Association, Seattle, WA, USA, Apr. 2014, pp. 459–473.

[114] S. Lange, A. Nguyen-Ngoc, S. Gebert, T. Zinner, M. Jarschel, A. Köpsel, M. Sune, D. Raumer, S. Gallenmüller, G. Carle, *et al.*, "Performance benchmarking of a software-based lte sgw", in *Network and Service Management (CNSM), 2015 11th International Conference on*, IEEE, Barcelona, Spain, Nov. 2015, pp. 378–383.

[115] R. Kelly and J. Gasparakis, “Common functionality in the 2.6 linux network stack”, *Whitepaper, Intel, Apr*, 2010.

[116] D. Scholz, “A look at intel’s dataplane development kit”, *Network*, vol. 115, 2014.

[117] ETSI-123-002, “Digital cellular telecommunications system (phase 2+); universal mobile telecommunications system (umts); lte; network architecture (3gpp ts 23.002)”, *Universal Mobile Telecommunications System (UMTS)*, pp. 2005–06,

[118] J. Donovan and K. Prabhu, *Building the Network of the Future: Getting Smarter, Faster, and More Flexible with a Software Centric Approach*. CRC Press, 2017, ISBN: 9781138631526.

[119] ETSI-GS-NFV-001, “Network functions virtualisation (nfv); use cases”, *V1*, vol. 1, pp. 2013–10, 2013.

[120] Y. Xie, Z. Liu, S. Wang, and Y. Wang, “Service function chaining resource allocation: A survey”, *arXiv preprint arXiv:1608.00095*, 2016.

[121] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[122] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, “Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications”, in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, Perth, WA, Australia, Apr. 2010, pp. 446–452.

[123] S. K. Garg and R. Buyya, “Networkcloudsim: Modelling parallel applications in cloud simulations”, in *Fourth IEEE International Conference on Utility and Cloud Computing*, IEEE, Victoria, Australia, Dec. 2011.

[124] J. Wolfgang, P. Konstantinos, A. George, J. Eduardo, K. Mario, M. Antonio, R. Fulvio, S. Dimitri, S. Rebecca, and M. Catalin, "Research directions in network service chaining", in *SDN for Future Networks and Services (SDN4FNS)*, IEEE, Trento, Italy, Nov. 2013.

[125] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers", in *Global Communications Conference (GLOBECOM), 2016 IEEE*, IEEE, Washington, DC, USA, Dec. 2016, pp. 1–6.

[126] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis", in *INFOCOM, 2014 Proceedings IEEE*, IEEE, Toronto, ON, Canada, Apr. 2014, pp. 2598–2606.

[127] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters", *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, 2015.

[128] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions", in *Network and Service Management (CNSM), 2014 10th International Conference on*, IEEE, Rio de Janeiro, Brazil, Jan. 2014, pp. 418–423.

[129] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation", in *Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE, Cincinnati, OH, USA, Aug. 2001, pp. 346–353.

ISSN 1432-8801