Universidade de Aveiro
2018
Departamento de Eletrónica,
Telecomunicações e Informática

David Manuel
Ribeiro Santos

Implementação de Serviços em Ambientes Multi-Access Edge Computing

Service Deployment on Multi-Access Edge Computing Environments

Universidade de Aveiro
2018

Departamento de Eletrónica,
Telecomunicações e Informática

David Manuel Ribeiro Santos

Implementação de Serviços em Ambientes Multi-Access Edge Computing

Service Deployment on Multi-Access Edge Computing Environments

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor Daniel Nunes Corujo, Investigador Doutorado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Rui Luís Andrade Aguiar, Professor Professor Catedrático do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor João Paulo Silva Barraca

Professor Auxiliar no Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Doutor Sérgio Figueiredo

Consultor/Engenheiro Sénior, Altran Portugal

Doutor Daniel Nunes Corujo

Investigador Doutorado no Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos /
acknowledgements

First of all, I would like to thank my supervisor, Professor Daniel Corujo for his continuous support, exceptional supervision, mentoring and for reviewing my thesis with expert opinion. I am grateful to all my colleagues at IT for providing technical assistance and friendly working environment. Last but not the least, my heartiest gratitude goes to my girlfriend and all my family for their love, trust, affection, patience and support throughout this study period. I dedicate this humble work to my honorable parents Manuel and Rosário.
Esta dissertação de mestrado foi realizada com o apoio do Instituto de Telecomunicações em Aveiro.

Palavras Chave

MEC, SDN, NFV, Redes 5G, Computação em cloud, Computação em Fog, Edge, 4G, Virtualização.

Resumo

Impulsionados pelas visões da quinta geração de redes móveis, e com uma crescente aceitação das tecnologias de redes baseadas em software, tais como funções de redes virtualizadas (NFV) e redes definidas por software (SDN), encontramo-nos perante uma transformação na infraestrutura nas redes de telecomunicações, assim como no modo como estas são geridas e implementadas. Uma das alterações mais significativas é a mudança no paradigma de computação na cloud, passando de uma implementação centralizada para uma ramificada na direção das extremidades da rede. Este novo ambiente, que possibilita uma plataforma de computação na extremidade da rede, é denominado de Multi-Access Edge Computing (MEC). A principal característica do MEC é fornecer computação móvel, armazenamento e recursos de rede na extremidade da rede, permitindo que terminais móveis com recursos limitados tenham acesso a aplicações exigentes em termos de latência e computação. Na presente tese, é apresentada uma solução de arquitetura MEC, que suporta ligações a redes de acesso heterogéneas, servindo de plataforma para a implementação de serviços. Alguns cenários MEC foram aplicados e avaliados na plataforma proposta, de forma a demonstrar as vantagens da implementação MEC. Os resultados demonstram que a plataforma proposta é significativamente mais rápida na execução computação intensiva, maioritariamente devido à baixa latência, quando comparado com os tradicionais datacenters centralizados, resultando numa poupança de energia e redução de tráfego no backhaul.

Keywords MEC, SDN, NFV, 5G Networks, Cloud Computing, Fog Computing, Edge, 4G, Virtualization

Abstract Driven by the visions of the 5th Generation of Mobile Networks (5G), and with an increasing acceptance of software-based network technologies, such as Network Function Virtualization (NFV) and Software Defined Networks (SDN), a transformation in network infrastructure is presently taking place, along with different requirements in terms of how networks are managed and deployed. One of the significantly changes is a shift in the cloud computing paradigm, moving from a centralized cloud computing towards the edge of the network. This new environment, providing a cloud computing platform at the edge of the network, is referred to as Multi-Acess Edge Computing (MEC). The main feature of MEC is to provide mobile computing, network control and storage to the network edges, enabling computation-intensive and latency-critical applications targeting resource-limited mobile devices. In this thesis a MEC architecture solution is provided, capable of supporting heterogeneous access networks, to assist as a platform for service deployment. Several MEC use case scenarios are evaluated on the proposed scheme, in order to attest the advantages of a MEC deployment. Results show that the proposed environment is significantly faster on performing compute-intensive applications, mainly due to lower end-to-end latency, when compared to traditional centralized cloud servers, translating into energy saving, and reduced backhaul traffic.

Contents

List of Figures

List of Tables

Glossary

AAA Authentication, Authorization, and Accounting

AAC Advanced Audio Coding

ADASs Interactive Advanced Driver-Assistance systems

AP Access Point

API Application Programming Interface

APN Access Point Name

APP Application

APPs Mobile Edge Applications

AR Augmented Reality

ASICs Application Specific Integrated Circuits

AUTN Authentication Token

BBU Baseband Unit

BGP Border Gateway Protocol

BSS Business Support Systems

BTS Base Transceiver Stations

CAL Control Abstraction Layer

CC Cloud Computing

CDN Content Delivery Network

CDMA Code Division Multiple Access

CFS Customer Facing Service

CHT Chunghwa Telecom

C-ITS Cooperative Intelligent Transport Systems

CK Cipher Key

CLI Command Line Interface

CN Core Network

CONCERT Concept Converging Cloud and Cellular Systems

COTS Commercial off-the-shelf

CPU Central Processing Unit

C-RAN Centralized Radio Access Network

CWC Centre for Wireless Communications

DAL Device and resource Abstraction Layer

DBaaS Database as a Service

DC Data Centers

DHCP Dynamic Host Configuration Protocol

DNS Domain Name System

EAPoL Extensible Authentication Protocol over Local Area Network

EAP-AKA Extensible Authentication Protocol - Authentication and Key Agreement

EBS Elastic Block Store

ECOMP Enhanced Control, Orchestration, Management and Policy

EDGE Enhanced Data rates for Global Evolution

EEG Electroencephalography

eNB eNodeB

EoL End of Life

EPC Evolved Packet Core

EPS Evolved Packet System

ETSI European Telecommunications Standards Institute

FMC Follow me Cloud

FMCC Follow me Cloud Controller

ForCES Forwarding and Control Element Separation

FOV Field Of View

FPS Frames per Second

GPRS General Packet Radio Service

GRE Generic Routing Encapsulation

GSM Global System for Mobiles

GTP General Packet Radio Service Tunneling Protocol

GUI Graphic User Interface

GW Gateway

HSS Home Subscriber Server

HTTP HyperText Transfer Protocol

IaaS Infrastructure as a Service

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

IK Integrity Key
IMSI International mobile subscriber identity
ITM International Mobile Telecommunication
IOM Internet of Me
IoT Internet of Things
IP Internet Protocol
IRTF Internet Research Task Force
ISG Industry Specification Group
IS-IS Intermediate System to Intermediate System
IT Information Technology
ITS Intelligent Transport System
ITU International Telecommunication Union
JSON JavaScript Object Notation
KPI Key Performance Indicator
LAN Local Area Network
LCM Application Lifecyle Management
LIPA Local Internet Protocol Access
L-SCM Local Small Cell Manager
LTE Long Term Evolution
LTE-A Long Term Evolution Advanced
MAC Media Access Control
MAL Management Abstraction Layer
MANO Management and Orchestration
Mbps Megabits per second
MC MobiScud Controll
MCC Mobile Cloud Computing
ME Mobile Edge
MEC Multi-access Edge Computing
MIB Management Information Base
MIMO Multiple-input Multiple-output
MMC Mobile Micro Cloud
MME Mobility Management Entity
MNO Mobile Network Operator
NAT Network Address Translation
NETCONF Network Configuration Protocol
NF Network Function
NFaaS Network Function as a Service
NFV Network Functions Virtualization
NFVI Network Function Virtualization Infrastructure
NFV MANO Network Function Virtualization Management and Orchestration
NIST The National Institute of Standards and Technology
NSAL Network Services Abstraction Layer
NSO Network Service Orchestrator
NV Network Virtualization
ONAP Open Network Automation Platform
ONF Open Networking Foundation
ONOS Open Network Operating System
OPEN-O Open Orchestrator
OS Operating System
OSM Open Source Management and Orchestration
OSPF Open Shortest Path First
OSS Operations Support System
OvS Open vSwitch
PaaS Platform as a Service
PGW Packet Data Network Gateway
PIMRC International Symposium on Personal, Indoor and Mobile Radio Communications
PoCs Proof of Concepts
PSK Pre-Shared Key
QoE Quality of Experience
QoS Quality of Service
RAM Random Access Memory
RAN Radio Access Network
RAND Random Number
RAT Radio Access Technology
RCA Root Cause Analysis
RES Response
REST Representational State Transfer
RGW Residential Gateway
RIEs Radio Interface Equipments
RNC Radio Network Controller
RPC Remote Procedure Call
RRC Radio Resource Control
R-SCM Remote Small Cell Manager
RTA Real-Time Applications
RTMP Real Time Messaging Protocol
RTT Round-Trip Time
SaaS Software as a Service
SCC Small Cell Cloud
SCeNB Small Cell eNodeB
SCM Small Cell Manager
SDN Software Defined Networking
SDNRG Software Defined Networking Research Group
SGW Serving Gateway
SGW-LBO Serving Gateway with Local Breakout
SLA Service Level Agreement
SNMP Simple Network Management Protocol
SP Service Provider
S/P-GW Serving/Packet Data Network Gateway

SQN	Sequence Number
TCP	Transmission Control Protocol
TD-LTE	Time Division Long Term Evolution
TI	Tactile Internet
TLS	Transport Layer Security
TOF	Traffic Offload Function
TOSCA	Topology and Orchestration Specification for Cloud Applications
TV	Television
UAV	Unmanned Aerial vehicles
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
UPF	User Plane Function
URI	Uniform Resource Identifier
USIM	Universal Subscriber Identity Module
UTM	UAV traffic management
vEPC	virtual Evolved Packet Core
VI	Virtual Infrastructure
VIM	Virtualized Infrastructure Manager
VLAN	Virtual Local Network
VL-SCM	Virtual Local Small Cell Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFM	Virtual Network Function Manager
VPN	Virtual Private Network
VR	Virtual Reality
VRO	Virtual Resource Orchestrator
VTT	VTT Technical Research Center of Finland
V2I	Vehicle-to-infrastructure
V2V	Vehicle-to-vehicle
WAN	Wireless Access Network
WPA	Wi-Fi Protected Access
WSAN	Wireless Sensor and Actuator Networks
XML	Extensible Markup Language
XRES	Expected Response
YANG	Yet Another Next Generation
1G	First Generation
2G	Second Generation
3G	Third Generation
3GPP	The 3rd Generation Partnership Project
4G	Forth Generation
5G	Fifth Generation
5GPPP	5G Public-Private Partnership Association
5GTN	5th Generation Test Network

CHAPTER 1

Introduction

1.1 Motivation

Due to the fast growth of smart devices, higher demand for heavy media content and crescent social media diffusion, new interactive, compute-intensive, mobile applications and the upcoming Internet of Things (IoT) networks, a transformation in network infrastructure is presently taking place. Telecommunications industry is facing new challenges and currently seeking for new methods and system models to cope with the upcoming 5G Networks requirements.

Cloud computing provides enterprises and end users with a way to deploy applications or services without the need to have hardware at their premises. Instead, they can run their applications in the cloud, where mechanisms such as load balancing are already in place. This flexibility allows enterprises and end users to lower the costs when it comes to run a certain application. As an effect of using cloud computing, computational and network overhead at central cloud increases, but this creates problems with real-time applications where latency is a crucial factor.

Despite the development in smartphones, they still have limited processing power capability and limited battery life, especially with the growing demand for energy-hungry applications, such as video streaming, Augmented Reality (AR) applications and 3D gaming. The preparation for deployment of future 5G networks sparked conversations about issues that need to be solved to increase the Quality of Experience (QoE) of applications based on these new networks. These applications require low latency and real-time data to effectively utilize its functionalities. Multi-access Edge Computing (MEC) is a promising technology introduced to reduce network stress by shifting resources to the network edge in the vicinity of end-users, reducing latency and improving bandwidth. Moreover, MEC is considered as one of the key enablers of IoT and big data applications as it offers low latency, local awareness due to proximity of the IoT devices from the edge of the network, wide-spread geographical distribution and interconnection with a large amount of nodes. MEC is a natural development in the evolution of mobile base stations and the convergence of Information Technology (IT)

and telecommunications networking. MEC will enable new vertical business segments and services for consumers and enterprise customers. The trend of pushing cloud computing to the edge of mobile networks is expected to continue to accelerate in years to come.

1.2 Objectives

The aim of this thesis is to provide a MEC platform, within a virtualization environment, capable of providing seamless integration of multiple applications and services, towards mobile subscribers, enterprises, and other vertical segments. This platform must support several types of access networks, such as wireless, mobile and cable connections, exploiting new emerging technologies such as SDN and NFV while enabling the deployment of applications with latency constrains and alleviating the load on core networks.

Also, to test the proposed environment, some scenarios will be evaluated on the implemented MEC platform.

1.3 Contributions

The architecture implemented in this work will be used as a base platform for future work in the ongoing "Mobilizador 5G" project. The work made on this thesis contributed to two papers:

- "Using SDN and Slicing for Data Offloading over Heterogeneous Networks Supporting non-3GPP Access", submitted to the Institute of Electrical and Electronics Engineers (IEEE) International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) 2018 with the authors Flávio Meneses, Rui Silva, David Santos, Daniel Corujo and Rui L. Aguiar (accepted).
- "An Integration of Slicing, NFV and SDN for Mobility Management in Corporate Environments" with the authors Flávio Meneses, Rui Silva, David Santos, Daniel Corujo and Rui L. Aguiar, submitted to the Transactions on Emerging Telecommunications Technologies journal.

This thesis was presented at the 25th Seminar of Rede Temática de Comunicações Móveis (RTCM) 2018.

1.4 Thesis Structure

In Chapter 2, an overview of the key enablers and background knowledge required to implement the proposed MEC architecture and use cases is provided as well as MEC state of the art, solutions and implemented testbeds. Chapter 3 presents the design of the overall test-bed, and the ensuing implementation of MEC architecture as well as the studied use cases. Chapter 4 focuses on the performance of the system, highlighting it's strengths and limitations. Moreover, benchmark of the MEC performance against the traditional centralized clouds is also discussed in this chapter. Chapter 5 presents the conclusions, contributions and future work.

CHAPTER 2

Key Enablers and State of the Art

This chapter introduces the state of the art and key concepts relevant to this thesis.

2.1 Virtualization

Virtualization is the process to create and run a simulated/virtual, rather than actual, version of something, such as an instance of a computer system, a storage device or a network device in a layer abstracted from the actual physical hardware [1]. Usually, it refers to running multiple Operating Systems (OSs) simultaneously in the same physical computer system.

To the virtualized OSs and applications, it appears that they are on a dedicated machine as there is an abstraction from the true underlying hardware or software.

A software called hypervisor is responsible for creating the abstraction layer between the separate, distinct and secure virtualized environments and the hardware or software which sits below.

Hypervisors can be divided into two classes; bare metal hypervisors that run virtual machines directly on a system's hardware behaving like a traditional OS (Type 1); and hosted hypervisors that run virtual machines on top of software, such as an OS, (these hypervisors behave as traditional applications, they can be started and stopped as regular programs [2], and are called Type 2 hypervisors).

There are many benefits to virtualization: to desktop users, the most common use is to be able to run applications meant for a different operating system without having to switch computers or reboot into a different system. For servers administrators, virtualization offers a way to slice a large system into many smaller entities, allowing different users and applications, with different resources requirements, to exploit the server more efficiently. It also allows isolation between applications running in different virtual machines, running in the same host, allowing running unsafe processes in one virtual machine without compromising the other virtual systems [2].

There are several virtualization types:

- *Data virtualization:* data virtualization allows companies to dynamically handle their data, supplying processing capabilities that can aggregate data from multiple sources, integrate new data sources, and provide data according to user or application requirements [3];
- *Operating system virtualization:* Operating system virtualization its a simple way to use different OSs for different applications side-by-side on the same machine. Enterprises can also push virtual operating systems to computers, which increases security, reduces hardware costs and decreases time spent on IT services [3];
- *Desktop virtualization:* Easily confused with operating system virtualization. Desktop virtualization allows a central administrator to deploy simulated desktop environments to several physical machines at once. Unlike traditional desktop environments which are physically installed, configured, and updated on each machine, desktop virtualization allows administrators to perform mass configurations, updates, and security checks on all virtual desktops [3];
- *Server Virtualization:* It is a way to have isolated servers running specific applications or tasks without security risks. It allows also, in theory, to create enough virtual servers to use all of a physical machine's processing power, thus less hardware is used [3];
- *Network Functions Virtualization (NFV):* NFV separates the network key functions, such as Network Address Translation (NAT), firewalling, Domain Name System (DNS), and Internet Protocol (IP) configuration, to name a few, from proprietary hardware appliances so they can run in software. Once software functions are independent of the underlying physical machines, specific functions can be packaged together into a new network and assigned to an environment [3]. Virtualizing networks reduces the number of physical components. NFV is detailed in section 2.3.

2.1.1 Network Virtualization

As stated on [4], *A networking environment supports network virtualization if it allows coexistence of multiple virtual networks on the same physical substrate.*

Network Virtualization (NV) is defined by the capability to create logical, isolated, virtual networks that are separated from the underlying network hardware, as surveyed on [4].

Two popular technologies for creating logical networks on the same physical substrate are Virtual Local Network (VLAN) and Virtual Private Network (VPN).

PlanetLab [5], 4WARD [6], GENI [7] and VINI [8] are some important projects on the NV area, before the arrival of SDN and NFV. The main benefits of NV include reduction of equipment, Quality of Service (QoS) improvement and optimization of resource usage [9].

2.2 Software Defined Networks

Conventional computer networks can be classified in three different planes: data, control and management. Typically the control plane (which decides how to handle network traffic) and the data plane (which forwards traffic according to the decisions made by the control plane) have always been bundled together on a large number of network devices such as

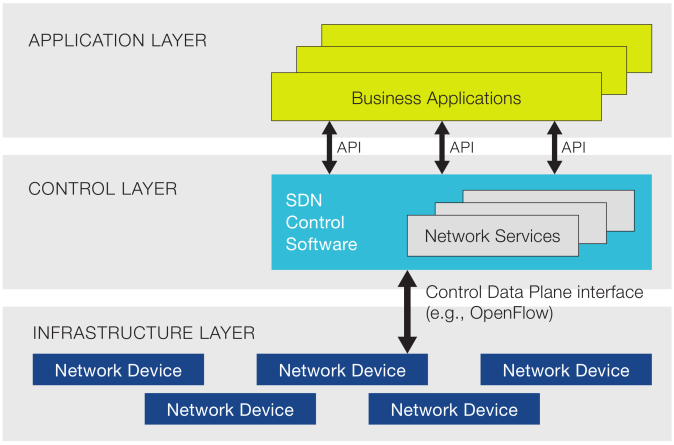


Figure 2.1: Overall view of SDN architecture

Source: [14]

routers, switches, and middle-boxes (i.e., devices that manipulate traffic for purposes other than packet forwarding, such as a firewall). These devices use special algorithms to control and monitor the data traffic in the network. In general, these routing algorithms and sets of rules are implemented in dedicated hardware components such as Application Specific Integrated Circuits (ASICs).

The network operators are responsible for configuring each individual network device separately using low-level and often vendor-specific commands. In addition to the configuration complexity, current networks do not adapt to the dynamics of faults and load, furthermore automatic reconfiguration and response mechanisms are virtually non-existent in today's IP networks. As a result, network management is quite challenging and thus error-prone [10] [11].

The idea of "programmable networks" was introduced to meet these challenges, and, as a result, a new network paradigm was presented: SDN, which promises to dramatically simplify and revolutionize traditional network architectures [11].

SDN essentially decouples the control plane from the data plane and move it to a centralized entity named controller. Therefore the complexity of network management is moved into this software-based controller which is directly programmable and manageable in a centralized manner [12]. This setup enables the underlying infrastructure to be abstracted for applications and network services, enabling the network to be treated as a logical entity [13]. Figure 2.1 depicts a logical view of the SDN architecture.

Despite the popularity of SDN in academia and industry in the early stages, there was a bit of confusion regarding the layers and interfaces of an SDN architecture.

With this in mind, the Open Networking Foundation (ONF) worked to define the SDN concept and terminology and the result was published in 2012 on a white paper entitled "Software-Define Networking: the new norm for networks" [14]. The Internet Research Task Force (IRTF) and Software Defined Networking Research Group (SDNRG) also worked intensively on clarifying these concepts and terminology. The result of this effort is the

RFC7426 [15], which addresses the questions about what exactly SDN is, what the layer structure is within the SDN architecture, and how layers interface with each other [16]. The ONF also worked to define the SDN concept and terminology and the result was published in 2012 on a white paper entitled "Software-Define Networking: the new norm for networks" [14].

2.2.1 Software Defined Networks Architecture

The SDN architecture spans multiple planes as illustrated in Figure 2.2. Starting from the bottom part of the figure and moving towards the upper part, RFC7426 [15] distinguishes the following five SDN planes:

- *Forwarding Plane:* Responsible for handling data packets in the data path based on the instructions received from the control plane. Actions of the forwarding plane include, but are not limited to, forwarding, dropping, and modifying packets. Examples of forwarding resources are classifiers, meters, etc. The forwarding plane is also widely referred to as the "data plane" or the "data path".
- *Operational Plane:* Responsible for managing the operational state of the network device, e.g., whether the device is active or inactive, the number of ports available, the status of each port, and so on. Examples of operational plane resources are ports, memory, and so on.
- *Control Plane:* Responsible for making decisions on how packets should be forwarded by one or more network devices and pushing such decisions down to the network devices for execution. The control plane's main job is to fine-tune the forwarding tables that reside in the forwarding plane, based on the network topology or external service requests.
- *Management Plane:* Responsible for monitoring, configuring, and maintaining network devices, e.g., making decisions regarding the state of a network device. The management plane usually focuses mostly on the operational plane of the device and less on the forwarding plane. The management plane may be used to configure the forwarding plane, but it does so infrequently and through a more wholesale approach than the control plane.
- *Application Plane:* The plane where applications and services that define network behavior reside. Applications that directly (or primarily) support the operation of the forwarding plane (such as routing processes within the control plane) are not considered part of the application plane.

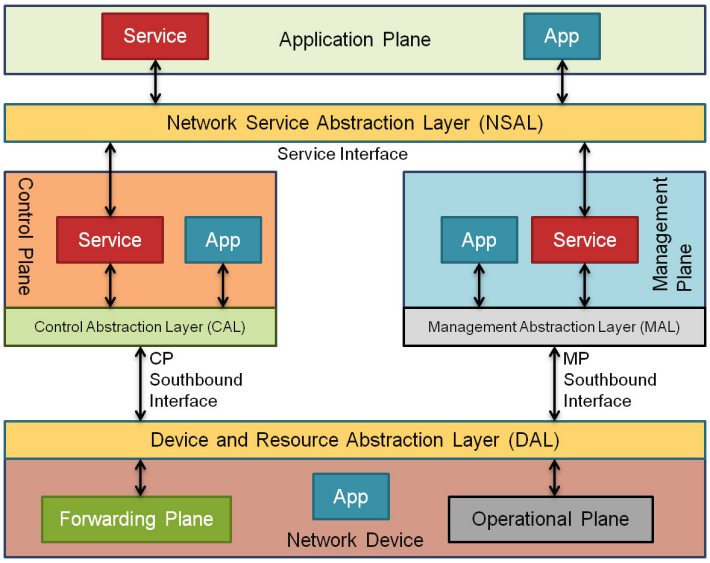


Figure 2.2: SDN Layer Architecture

Source: [15]

2.2.2 Software Defined Networks Interfaces

All planes mentioned above are connected via interfaces. An interface may take multiple forms depending also on whether the connected planes reside on the same device or on different devices. For the latter, then the interface can only take the form of a protocol. In case of the former, the interface could be implemented via an open/proprietary protocol, an open/proprietary software inter-process communication Application Programming Interface (API), or operating system kernel system calls. RFC7426 [15] focuses on the north/south communication between entities in different planes but does not exclude entity communication within any one plane (eastbound-westbound communication).

It is important to distinguish between control and management interfaces as they have their own distinct characteristics depending on the respective planes. Initially the management plane was considered out of scope for SDN, but now documentation has been published by both International Telecommunication Union (ITU) [17] and ONF [18] that includes the management plane and are well aligned with RFC7426 [15].

2.2.3 Software Defined Networks Abstraction Layers

RFC7426 [15] defines the following abstraction layers as is illustrated on Figure 2.2:

- *Device and resource Abstraction Layer (DAL):* abstracts the resources of the device's forwarding and operational planes to the control and management planes. The DAL is one of the most important abstraction layers, as the services that the rest of the planes provide depend on the DAL's richness and flexibility to describe resources. Some examples of forwarding-plane abstraction models are Forwarding and Control Element Separation (ForCES) [19], OpenFlow [20], Yet Another Next Generation (YANG) model [21], and Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) [22].

- *Control Abstraction Layer (CAL):* abstracts the Control Plane Southbound Interface and DAL from the applications and services of the control plane. Control applications can use CAL to control a network device without providing any service to upper layers. Examples include applications that perform control functions, such as Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Border Gateway Protocol (BGP).
- *Management Abstraction Layer (MAL):* abstracts the Management Plane Southbound Interface and DAL from the applications and services of the management plane. Management applications can use MAL to manage the network device without providing any service to upper layers. Examples of management applications include network monitoring, fault detection, and recovery applications.
- *Network Services Abstraction Layer (NSAL):* provides access to services of the control, management, and application planes to other services and applications. Service interfaces can take many forms pertaining to their specific requirements. The two leading approaches for service interfaces are RESTful interfaces and Remote Procedure Call (RPC) interfaces. Both follow a client-server architecture and use Extensible Markup Language (XML) or JavaScript Object Notation (JSON) to pass messages, but each has some slightly different characteristics.

2.2.4 OpenFlow

A number of protocol standards exist on the use of SDN in real applications, with the most popular protocol standard called OpenFlow.

It was proposed in the paper "OpenFlow: enabling innovation in campus networks" [23], aiming to provide a high performance traffic control across multiple vendors' network devices.

OpenFlow is one of the first protocol designed specifically for SDN, and is already standardised by the ONF [24].

OpenFlow is a open protocol used to establish communication between the control application layer (e.g., software applications) and forwarding plane (e.g., routers, switches) within the SDN infrastructure, thus enabling the handling of the network as a whole rather than individual devices, granting a more efficient use of network resources.

2.2.4.1 OpenFlow Architecture

An OpenFlow based architecture essentially consists of three main components: an OpenFlow-compliant switch, an OpenFlow controller, and a secure communication channel which provides a link for the transmission of commands and packets between the controller and the switch, as shown in figure 2.3 [25].

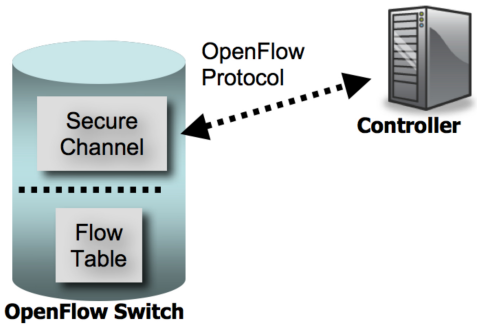


Figure 2.3: OpenFlow based Architecture from OFv1.0 specifications

Source: [25]

2.2.4.2 OpenFlow Switch

An OpenFlow Logical Switch consists of one or more *flow tables*, a *group table* which performs packet lookups and forwarding, and one or more OpenFlow channels to an external controller (Figure 2.4). The switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol through a secure channel [20] [26]. The OpenFlow switches establish a communication channel with the controller via an IP address using a specified port, then initiates a standard Transport Layer Security (TLS) or Transmission Control Protocol (TCP) connection to the controller. The traffic between the controller and switch does not go through the OpenFlow pipeline, so it must identify when incoming traffic is from the controller before matching it with the *flow tables*. Each OpenFlow switch may establish communication with a single or multiple controllers [20].

OvS [20] is one of the most popular software-driven OpenFlow switches.

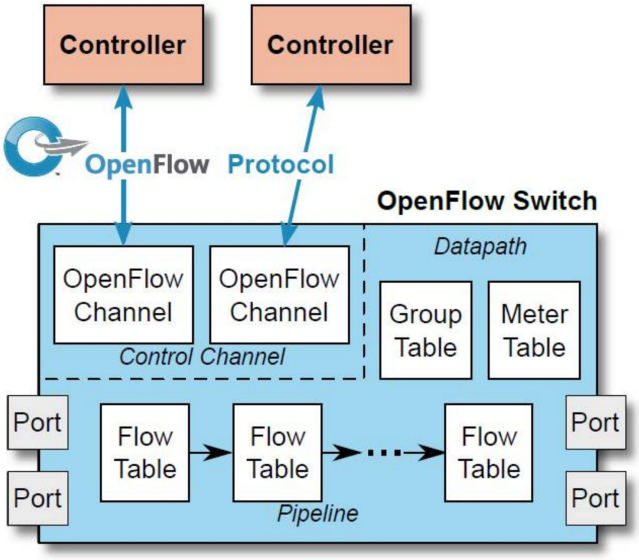


Figure 2.4: OpenFlow based Switch

Source: [20]

2.2.4.3 OpenFlow Flow Tables, Match Fields and Actions

Switches use flow tables (i.e. a list of flow entries) to forward packets. Each entry consists of: match fields, counters and instructions. Match fields are compared against the fields of the packet being processed; Counters are used to keep statistics about packets; and instructions to apply to the processed packet, such as modifying the packet's fields, drop the packet, forward the packet or encapsulate the packet and send it to the controller.

Incoming packets are compared with the match fields of each entry and if there is a match, the packet is processed according to the action contained by that entry.

The OpenFlow specification defines three types of tables in the logical switch architecture. A flow table matches incoming packets to a particular flow and specifies the actions that are to be performed on the packets. There may be multiple flow tables that operate in a pipeline fashion, as show in figure 2.5 A flow table may direct a flow to a group table, which may trigger a variety of actions that affect one or more flows. A meter table can trigger a variety of performance-related actions on a flow.

The typical packet processing flow in a OpenFlow switch is the following: When a packet arrives at the switch, the packet is matched against the flow entries of the flow table, if a match is found, the instruction set included in that flow entry is executed. These instructions may explicitly direct the packet to another flow table (using the GotoTable Instruction), where the same process is repeated again.

A flow entry can only direct a packet to a flow table number which is greater than its own flow table number, in other words, pipeline processing can only go forward and not backward. If the matching flow entry does not direct packets to another flow table, the current stage of pipeline processing stops at this table and the packet is processed with its associated action set. If a packet does not match a flow entry in a flow table, this is a table miss. The behavior on a table miss depends on the table configuration. The instructions included in the

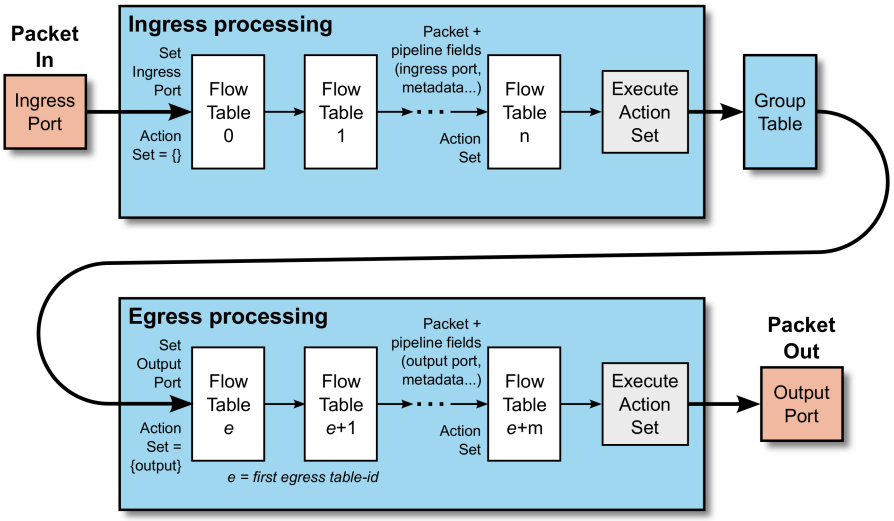


Figure 2.5: Packet flow through the processing pipeline

Source: [20]

table-miss flow entry in the flow table can flexibly specify how to process unmatched packets, such as dropping them, passing them to another table or sending them to the controllers over the control channel via packet-in messages. There are few cases where a packet is not fully processed by a flow entry and pipeline processing stops without processing the packet's action set or directing it to another table. If no table-miss flow entry is present, the packet is dropped [20].

2.2.4.4 OpenFlow Controller

The OpenFlow controller is the entity responsible for manipulating the switch's flow tables, managing how to handle traffic without valid flow entries, as well as distributing appropriate instructions to the network devices, using the OpenFlow protocol via a secure channel. The controller has a global, logical view of the network.

Several controllers have been proposed since OpenFlow was brought to the community in 2008 [27]. The first open source controller was NOX [28], followed by others, such as, Maestro [29], Beacon [30] or Trema [31]. Examples of controllers that address scalability [32] are DevoFlow [33], ONIX [34] or FlowN [35]. Examples of open source OpenFlow controllers, still maintained, being among the most widely used [36]: OpenDaylight [37], POX [38], Floodlight [39], ONOS [40] and Ryu [41].

2.3 Network Function Virtualization

The main idea behind NFV is to virtualize network functions, and migrate these functions from stand-alone boxes on dedicated hardware, to appliances running on cloud systems [42].

A network service can be broken down into a set of network functions, which are then virtualized and executed on general purpose servers. This approach allows Virtual Network Functions (VNFs) to be dynamically instantiated, relocated or destroyed, without necessarily requiring the purchase and installation of new hardware, giving flexibility and reducing costs to the network operator [43].

NFV paves the way to a number of differences in the way network service provisioning is realized in comparison to current practice. In summary, these differences are as follows [44]:

- *Decoupling software from hardware:* As the network elements are no longer a composition of integrated hardware and software entities, the evolution of both is independent of each other. This enables the software to progress separately from the hardware, and vice versa.
- *Flexible network function deployment:* The detachment of software from hardware helps reassign and share the infrastructure resources, thus together, hardware and software, can perform different functions at various times. The actual network function software instantiation can become more automated, leveraging the different cloud and network technologies currently available. Also, this helps network operators to deploy new network services faster over the same physical platform.
- *Dynamic scaling:* The decoupling of the functionality of the network function into instantiable software components provides greater flexibility to scale the actual VNF

performance in a more dynamic way and with finer granularity, for instance, according to the actual traffic for which the network operator needs to provision capacity.

The general concept of decoupling network functions from dedicated hardware does not necessarily require virtualization of resources. This means that network operators could still purchase or develop network functions and run them on physical machines. The difference is that these network functions would have to be able to run on commodity servers. Nonetheless, the advantages of running these functions on virtualized resources (e.g. dynamic resource scaling, flexibility, energy efficiency) are very strong selling points of NFV. It is also possible to have hybrid scenarios where functions running on virtualized resources co-exist with those running on physical resources: those hybrid scenarios may be important in the transition towards NFV [45].

2.3.1 Network Function Virtualization Architecture

According to ETSI, the NFV architecture is composed by three key elements, as illustrated on figure 2.6: Network Function Virtualization Infrastructure (NFVI), VNFs and services, and Network Function Virtualization Management and Orchestration (NFV MANO).

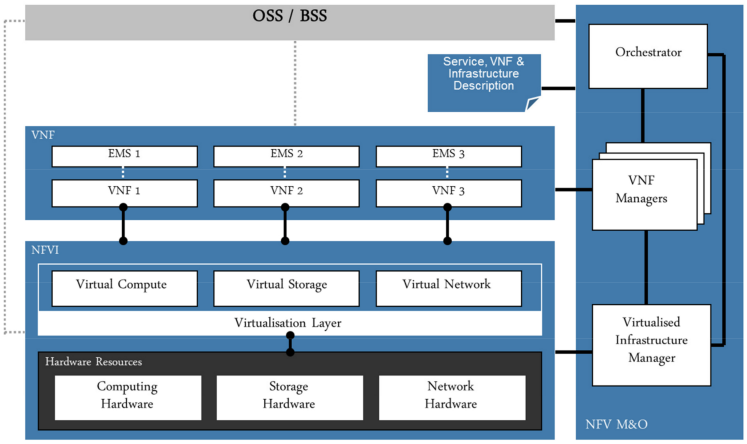


Figure 2.6: ETSI NFV Architecture

Source: [20]

2.3.1.1 Network Function Virtualization Infrastructure

The NFVI is the combination of both hardware and software resources which make up the underlying environment in which VNFs are deployed. The physical resources consist of Commercial off-the-shelf (COTS) computing hardware, storage and network (made up of nodes and links) that provide processing, storage and connectivity to VNFs. Virtual resources are an abstraction of both computing, storage and network resources. This abstraction is accomplished using a hypervisor to virtualize network functions, decoupling the virtual network functions from the underlying software or hardware they run on.

2.3.1.2 Virtual Network Functions and Services

A Network Function (NF) is a functional block within a network infrastructure that has well defined external interfaces and well-defined functional behaviour [46], such as Residential Gateway (RGW), firewalls, DHCP servers etc.

Basically, a VNF is an implementation of an NF deployed in virtual resources. A single VNF may be composed by a set of functions, and hence it could be deployed over multiple VMs, in which case each VM hosts a single component of the VNF [44].

A service is an offering provided by a network operator that is composed of one or more NFs. In NFV, the NFs that compose the service are virtualized and deployed on virtual resources. Nonetheless, in the users' perspective, the services should have the same or better performance, whether running in traditional hardware boxes or in VMs [45].

2.3.1.3 Network Function Virtualization Management and Orchestration

NFV MANO provides the functionalities required for the provisioning and configuration of VNFs, along with the configuration of the infrastructure these functions run on [47].

It includes the orchestration and lifecycle management of VNFs and physical and/or software resources that support the infrastructure virtualization. It also includes databases that are used to store the information and data models which define both deployment as well as lifecycle properties of functions, services, and resources.

In addition, NFV MANO also defines interfaces that can be used for communications between the different components, as well as coordination with traditional network management systems such as Operations Support System (OSS) and Business Support Systems (BSS).

The ETSI NFV reference architecture specifies initial functional requirements and outlines the required interfaces [44]. However, it excludes aspects such as control and management of legacy equipment, thus it is complex to specify the operation and Management and Orchestration (MANO) of an end-to-end service involving both VNFs and legacy functions [45].

2.3.2 Combining Network Virtualization and Software Defined Networking

Both NFV and SDN have a lot in common since they both promote towards open software and standard network hardware, exploiting automation and virtualization to achieve their respective goals.

In fact, NFV and SDN may be highly complementary, as said on ETSI NFV white paper [42], and hence combining them in one networking solution may lead to greater value.

For example, SDN can accelerate NFV deployment by offering a flexible and automated way of chaining functions, provisioning and configuration of network connectivity and bandwidth, automation of operations, security and policy control [48].

However, SDN and NFV are mutually beneficial but are not dependent on each other [42], they promote different concepts, aimed at addressing different aspects of a software-driven networking solution.

Table 2.1: Comparison of Software Defined Network and Network Function Virtualization Concepts

Issue	Network Function Virtualization	Software Defined Networking
Approach	Service/Function Abstraction	Networking Abstraction
Standards Organization	ETSI	ONF
Advantage	Flexibility and cost reduction	Unified programmable control and open interfaces
Protocol	Multiple control protocols /e.g. SNMP,NETCONF)	OpenFlow
Applications deployment	Commodity servers and switches	Commodity servers for control plane and possibility for specialized hardware for data plane
Leaders	Mainly Telecom	Mainly networking software and hardware vendors
Business Initiator	Telecom service providers	Born on the campus, matured in the data center

Source: [45]

NFV aims at decoupling NFs from specialized hardware elements while SDN focuses on splitting the handling and routing of packets and connections from overall network control [45].

As stated by the ONF in the description of the SDN architecture [18], *"the NFV concept differs from the virtualization concept as used in the SDN architecture. In the SDN architecture, virtualization is the allocation of abstract resources to particular clients or applications; in NFV, the goal is to abstract NFs away from dedicated hardware, for example to allow them to be hosted on server platforms in cloud data centers."*

Finally, an important distinction is that while NFV can work on existing networks, because it resides on servers and interacts with specific traffic sent to them, SDN requires a new network construct where the data and control planes are decoupled [45].

The relationship between SDN and NFV is described on Table 2.1

2.4 Cloud Computing

Cloud computing has become a tremendous paradigm for hosting and delivering services over the Internet. Diverse formal definitions have been proposed in both research studies and IT industry.

As defined by The National Institute of Standards and Technology (NIST): *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction* [49].

The goal of this computing model is to make a better use of various computer resources, in order to achieve higher throughput and to resolve problems that require high performance

computations. It allows users to develop and leverage available computer resources in a pay as you go manner [50].

The main essential characteristics of Cloud Computing (CC) include [50]:

- *On-demand self-service:* A user can use cloud resources, such as computation, server time or storage, automatically without requiring human interaction with each service provider.
- *Broad network access:* Cloud resources are accessible over the Internet and accessed through diverse terminals (e.g., computers, mobile phones, tablets etc.).
- *Resource pooling:* The computing resources are pooled together to serve multiple consumers in a multi-tenant manner. Both physical and virtual resources are assigned as consumers demand. The customer generally has no control or knowledge over the exact location of the provided computer resources, however users may be able to specify location requirements.
- *Rapid elasticity.* Computer resources provisioned to users can be elastically created and released and can be scaled rapidly outward and inward appropriately with consumers demand.
- *Measured service.* Computer resources can be automatically monitored, controlled and optimized using a metering capability appropriate to the type of service provisioned. Warnings can be reported to both provider and consumer.

2.4.1 OpenStack

Using the simplest definition, OpenStack is a framework for managing, defining and use cloud resources. The official Openstack website[1] defines: *OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface.*

OpenStack is a collection of open source software modules that provides a framework to create and manage both public cloud and private cloud infrastructure. To create a cloud computing environment, an organization typically builds off of its existing virtualized infrastructure, using a well-established hypervisor such as VMware vSphere[2], Microsoft Hyper-V[3] or KVM[4]. But cloud computing goes beyond just virtualization. A public or private cloud also provides a high level of provisioning and lifecycle automation, user self-service, cost reporting and billing, orchestration and other features. Openstack is a "cloud operating system" that can organize, provision and manage large pools of heterogeneous compute, storage and network resources. While an IT administrator is typically called on to provision and manage resources in a more traditional virtualized environment, OpenStack enables individual users to provision resources through management dashboards and the OpenStack API. It makes horizontal scaling easy, which means that tasks that benefit from running concurrently can easily serve more or fewer users on the fly by just spinning up more instances.

[1] www.openstack.org
[2] https://www.vmware.com/products/vsphere.html
[3] https://msdn.microsoft.com/pt-pt/library/mt169373(v=ws.11).aspx
[4] https://www.linux-kvm.org/

Openstack has a modular architecture with code names for its modules. These modules, shaped by open source contributions from the developer community, focus on specific functions within the cloud ecosystem. Key Openstack components, by category, include:

- Compute
  - Glance - a service that discovers, registers and retrieves virtual VM images;
  - Ironic - a bare-metal provisioning service;
  - Magnum - a container orchestration and provisioning engine;
  - Nova - a service that provides scalable, on-demand and self-service access to compute resources, such as VMs and containers;
  - Storlets - a computable object storage service;
  - Zun - a service that provides an API to launch and manage containers.
- Storage
  - Cinder - a block storage service;
  - Swift - an object storage service;
  - Freezer - a backup, restore and disaster recovery service;
  - Karbor - an application and data protection service;
  - Manila - a shared file system.
- Networking and content delivery
  - Designate - a DNS service for the network;
  - Neutron - a SDN service for virtual compute environments;
  - Dragonflow - a distributed control plane implementation of Neutron;
  - Kuryr - a service that connects containers and storage;
  - Octavia - a load balancer;
  - Tacker - an orchestration service for NFV;
  - Tricircle - a network automation service for multi-region cloud deployments.
- Data and analytics
  - Sahara - a provisioning service for big data projects;
  - Searchlight - a data indexing and search service;
  - Trove - a Database as a Service (DBaaS).
- Security and compliance
  - Barbican - a management service for passwords, encryption keys and X.509 Certificates;
  - Congress - an IT governance service;
  - Keystone - an authentication and multi-tenant authorization service;
  - Mistral - a workflow management and enforcement service.
- Deployment
  - Ansible OpenStack - a service that provides Ansible[5] playbooks for OpenStack;
  - Chef OpenStack - a service that provides Chef[6] cookbooks for OpenStack;

[5]https://www.ansible.com/
[6]https://www.chef.io

  - Kolla - a service for container deployment;
  - Charms - a service that offers Juju[7] charms for OpenStack;
  - Puppet OpenStack - a service that provides Puppet[8] modules for OpenStack;
  - TripleO - a service to deploy OpenStack in production.
- Management
  - Horizon - a management dashboard and web-based user interface for OpenStack services;
  - OpenStack Client - the OpenStack Command Line Interface (CLI);
  - Rally - an OpenStack benchmark service;
  - Senlin - a clustering service;
  - Vitrage - a Root Cause Analysis (RCA) service for troubleshooting;
  - Watcher - a performance optimization service.
- Applications
  - Heat - orchestration and auto-scaling services;
  - Murano - an application catalog;
  - Solum - a software development tool;
  - Zaqar - a messaging service.
- Monitoring
  - Aodh - an alarming service that takes actions based on rules;
  - Ceilometer - a metering and data collection service;
  - CloudKitty - a billing and chargeback service;
  - Monasca - a high-speed metrics monitoring and alerting service;
  - Panko - a service for metadata indexing and event storage to aid auditing and troubleshooting.

The major modules are listed in Figure 2.7. This is a simplified view of the architecture, assuming that all the services are used in the most standard configuration.

OpenStack has followed an alphabetical naming scheme for its version releases since its initial Austin release in October 2010. The original Austin, Baxar and Cactus releases have since been deprecated, and are no longer available. More recent releases, between 2012 and 2016, include Diablo, Essex, Folsom, Grizzly, Havana, Icehouse, Juno, Kilo, Liberty, Mitaka and Newton, which are all at End of Life (EoL).

These were followed by the Ocata release in February 2017, and the Pike release in August 2017. Pike added a variety of new features, including support for Python 3.5, a revert-to-snapshot feature in Cinder and support for globally distributed erasure codes in Swift. The current one, Queens, was released on February 2018.

[7] https://www.ubuntu.com/cloud/juju
[8] https://puppet.com/

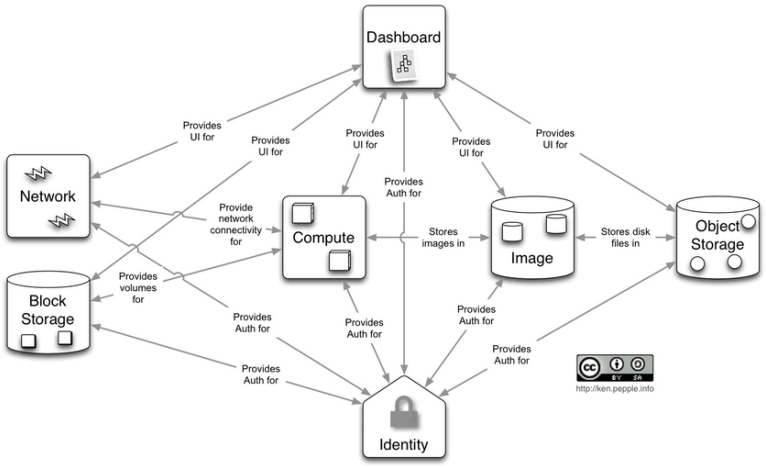


Figure 2.7: Openstack Diagram

Source: [9]

2.5 5G Network

In the simplest possible definition, 5G, which is short for “fifth generation", is the next generation of wireless networks and it promises to improve the delivery of mobile services. 5G is best understood in terms of its predecessors: The first generation of mobile networks (now referred as First Generation (1G)), was fully analog and came out in around 1982. The second generation of mobile networks, 2G, launched in 1991, made the jump to digital, expanded from a voice-based technology to one that supported text messaging and also added cellular data in the form of General Packet Radio Service (GPRS) and Enhanced Data rates for Global Evolution (EDGE) technologies. In 2001, the third generation, 3G, was launched offering even faster data rate than the previous generation. Roughly ten years after, the fourth generation was launched. 4G Long Term Evolution (LTE) enhanced those capabilities with higher speeds and increased reliability while having a low complexity network. 4G is a packet switched only network, dropping the circuit switch part of previous generations. Historically, that works out to a new generation of networking technology every decade or so.

5G promises not only to deliver higher data rates and reliability to mobile users (similarly to the evolution of previous cellular technologies) but also to provide a framework to support new verticals (i.e., new business) and new applications with specific requirements by using the concept of network slicing. Network slicing, one the the key enablers for 5G, enables an operator to run multiple independent logical networks sharing the same underlying hardware thus enabling the networks to be tailored as required by a certain service or application. Network slicing is possible by using the concepts of NFV and SDN, deploying the network functions in a cloud as VNFs. Some applications require specific Key Performance Indicators (KPIs) such as automotive, e-Health or Industry 4.0, that require low latency. In order to cope with this requirement, 5G proposes the use of MEC, deploying a cloud at the edge of the network, closer to the users, providing a way to lower the latency felt by the end user and reducing the load on core networks. This combination of network slicing, the use of NFV and SDN

and the ability to deploy network functions closer to the user enable a network operator to deploy multiple slices tailored according to the needs of a specific service. As an example, in order to support low latency applications such as AR, the required network functions would be deployed in the edge cloud, while for other applications without low latency requirements such as IoT, the network functions could be deployed at the core cloud. This way, two slices are created, sharing the same underlying infrastructure but with different KPIs. In order to automate network management, 5G aims to use a MANO mechanism, allowing network functions or services to be deployed or scaled on demand.

The organization that governs cellular standards, 3GPP, released its first 5G formal standard, Release 15 [51] [52] [53], in December 2017. As mentioned above, 5G brings about more improvements, but it's also comprised of a suite of new technologies. Not every vendor agrees on what should be included in the final specifications, but the most popular contenders are small cells; millimeter waves; massive Multiple-input Multiple-output (MIMO); beamforming; and full duplex, for the radio access part. For the core part of the network, the key aspects will reside on the introduction of SDN, NFV and cloud-based mechanisms for the operation of the network.

2.6 Multi-Access Edge Computing

2.6.1 Introduction

In recent years, there has been an increasing number of mobile subscriptions (63% subscription in 2016, whereas it was just 20% on the previous decade). Almost half a billion (429 million) mobile devices and connections were added in 2016 and by 2021 there will be 1.5 mobile devices per capita [54]. With the continuous technological evolution of the high end devices UEs (e.g., smartphones and tablets), conjoint with new and interactive mobile applications, new challenges to the mobile and wireless communications worldwide are created [55].

These networks have to cope with mobile devices with low storage capacity, lack of computational power, high energy consumption and with low bandwidth high latency interfaces [56]. In addition, exponential growth of IoT devices will overload an already congested network. Despite today's mobile devices being more powerful in terms of Central Processing Unit (CPU) and storage, they may not be able to handle applications requiring huge processing in a short period of time, and with high battery consumption performing these tasks still constitutes a major drawback, restraining the users QoE [57].

These aspects motivate the development of the Mobile Cloud Computing (MCC) concept, as an integration of cloud computing and mobile computing, providing mobile devices' users with storage services and data processing in the centralized cloud [58].

Despite MCC being able to address some of the issues, it still experiences several challenges, such as high latency, low coverage and lagged data transmission [59], that will become even more expressive with the next mobile generation 5G.

Furthermore, because MCC is located in a centralized CC far away from the mobile user (in terms of network topology) it is not suited for real-time scenarios (e.g. AR and Real-Time

Applications (RTA)) and since all data is sent to the centralized servers it overloads the backhaul of mobile networks, increasing the latency.

To address these problems, the cloud services should be moved towards the edges of the network, and a new paradigm was created: MEC [60].

By bringing the MCC closer to the UEs it is widely agreed that MEC is a key technology to deploy next-generation Internet scenarios [61] such as IoT [62], Tactile Internet (TI) [63] and Internet of Me (IOM) [64].

ETSI Industry Specification Group (ISG) MEC, in its first introductory white paper [60], defined that: *"Mobile-Edge Computing provides IT service environment and cloud-computing capabilities within the Radio Access Network (RAN) in close proximity to mobile subscribers."*

2.6.2 Evolution towards Multi-Acess Edge Computing

In this section some solutions and concepts of cloud computing moving towards MEC are described.

2.6.2.1 Mobile Cloud Computing

The Mobile Cloud Computing (MCC) is described as an integration of cloud computing technology with mobile devices [65].

Mobile devices face some constraints [66], such as finite energy source, variable wireless behavior and different bandwidth connectivity, memory size, storage capacity and computing power. A solution to solve some of these constrains is to use cloud computing in the mobile environment [67], leading to the emergence of a new paradigm called MCC. MCC can address these problems by executing mobile applications on a resource provider in the Cloud platform. A generic architecture of MCC is presented in figure 2.8.

The MCC architecture is composed by:

- *Mobile Network:* The mobile network controls the connection of the functional interface between mobile device and network operator. The mobile devices are able to be connected

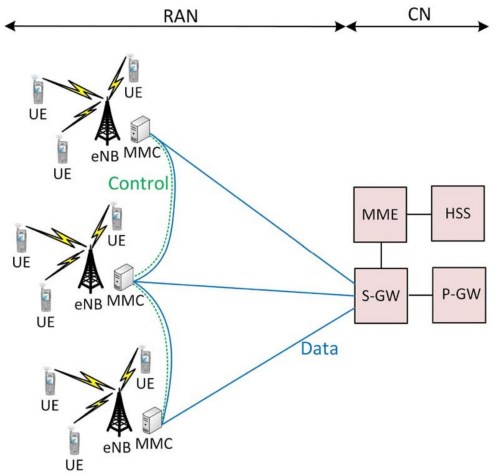


Figure 2.8: MCC Architecture

Source: [68]

to the mobile network through satellite, Base Transceiver Stations (BTS) or access points.

- *Internet Service:* The Internet connects the mobile network with the cloud. The mobile users can access the cloud services via LTE, Global System for Mobiles (GSM), Code Division Multiple Access (CDMA) or wireless connections.
- *Cloud Services:* The service-oriented cloud computing architecture can be divided in three layers, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS); IaaS offers virtualization platforms such as VMs. Users can get these infrastructure services dynamically according to their requirements. PaaS provides a platform where users can build, deploy and test their applications. Users can get software, such as databases, in SaaS [68].

2.6.2.2 Fog Computing

Fog computing, alternatively known as fog networking or "fogging", is a term originally introduced by Cisco, proposed to enable cloud computing architecture away from centralized cloud datacenters, considering a large number of geographically wide spread edge nodes as a part of a distributed and collaborating cloud. Fog computing foresees cloud nodes deployed directly at the edge of the network and being capable to deliver new applications and services, especially for the future IoT services [69] [70].

The notion of fog computing nodes is wide. Any equipment with processing power and storage, e.g., ranging from wireless access points, switches and routers to base stations and resource-rich datacenters or cloud platforms, can be qualified as a fog node [71]. Cisco introduced the first commercial fog device, IOx[72], capable of hosting applications in a guest operating system running on a hypervisor directly on routers. As a generic cloud platform to develop and execute software, fog computing is considered as an open ecosytem for wearables/IoT, big-data analytics and for emerging services such as automotive, hostile and tactile applications [73].

2.6.2.3 Cloudlets

A cloudlet is a mobility-enhanced small-scale cloud datacenter that is located at the edge of the Internet. The main purpose of the cloudlet is supporting resource-intensive and interactive mobile applications by providing powerful computing resources to mobile devices with lower latency. It is a new architectural element that extends today's cloud computing infrastructure. It represents the middle tier of a 3-tier hierarchy: mobile device - cloudlet - cloud. A cloudlet can be viewed as a data center in a box whose goal is to bring the cloud closer. The cloudlet term was introduced by Satyanarayanan *et al.* in [67] [66]. It was purposed as an edge cloud node, which can reside in community places, e.g, coffee shops or shopping malls, and highly populated areas, e.g., train stations and exhibition halls [73]. Cloudlets are instantiated based on a soft state implementation that relies on VMs and merely acts as micro datacenters in a box, offering access to end users over Wi-Fi for deploying and managing their own VM [74]. Using a VM-based approach, cloudlets offer a transient guest environment for individual users, providing isolation from other host software.

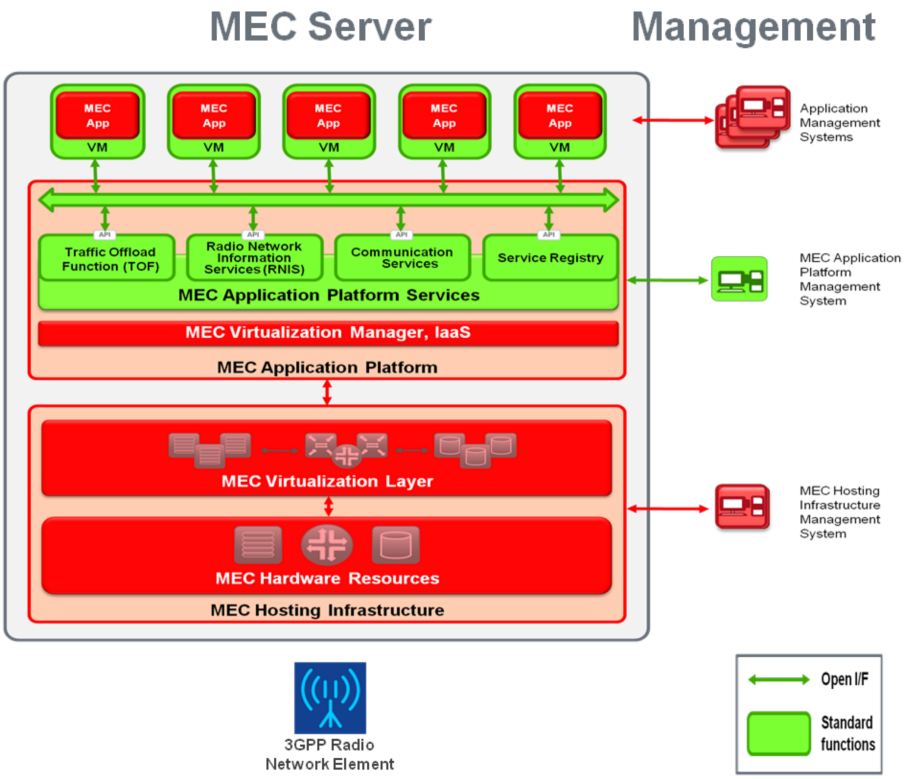


Figure 2.9: MEC server platform overview as displayed in ETSI first technical white paper
Source: [60]

2.6.3 Multi-Access Edge Computing Architecture and Standardization

This section introduces several concepts for the computation on the edge and standardization.

2.6.3.1 Multi-Access Edge Computing ETSI

Standardization is an indispensable step for promoting and evolving a new technology, which documents the consensus among multiple players and defines characteristics and rules in a specific industry.

The standardization effort by ETSI has started in 2014, and a new ISG was established set up by Huawey, Vodafone, Nokia Networks, IBM and NTT docomo. The purpose of the ISG is to create a standardized, open environment which will allow the efficient and seamless integration of applications from vendors, service providers, and third-parties across multi-vendor MEC platforms [75].

In 2014, ETSI, published is first technical white paper on MEC [60] where the MEC concept was introduced, consumer and technical benefits were discussed and a high-level architectural blue print of MEC was introduced, as showed in figure 2.9. Additionally, typical use case scenarios, technical requirements and challenges for MEC were pointed out. Later, these aspects have been documented in more detail, in the ETSI specifications [76] [77] [78]. At the MEC World Congress 2016, ETSI has announced six Proof of Concepts (PoCs) [79] which will assist the strategic planning and decision-making of organizations, helping them identify which MEC solutions may be viable in the network. In this congress ETSI MEC ISG

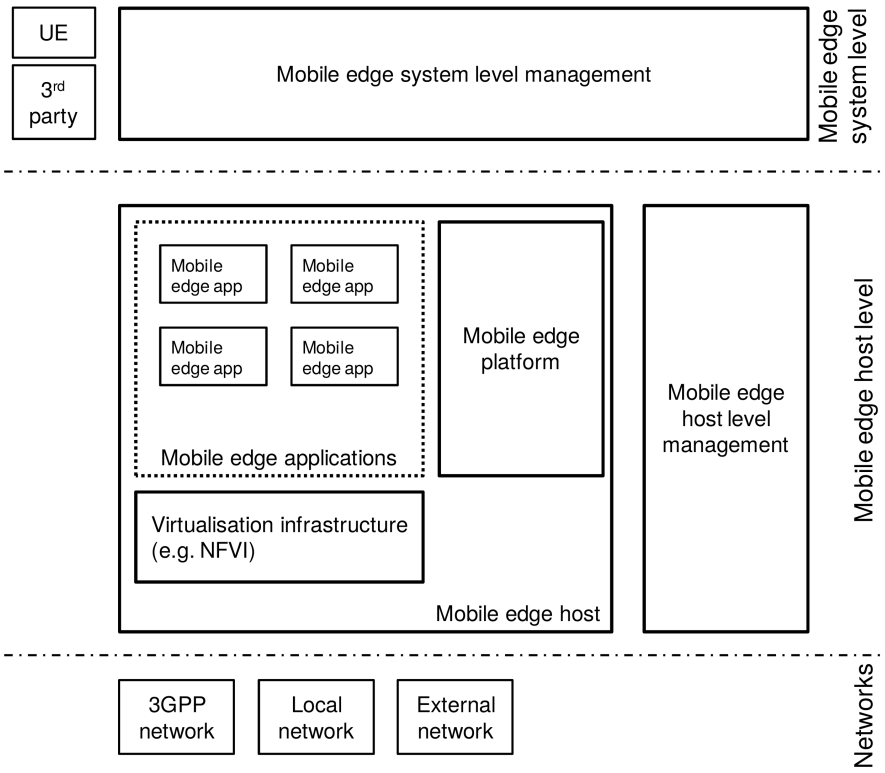


Figure 2.10: MEC framework overview

Source: [76]

has renamed *Mobile Edge Computing* to *Multi-access Edge Computing* in order to express the growing interest in MEC from non-cellular operators [80]. Recently 3GPP has included MEC architecture into its 5G standards and in recent technical specification document [51].

The MEC framework, described in [76], shows the abstract entities and functions involved in the MEC structure, categorized into network level, Mobile Edge (ME) system level, ME host level and networks level as illustrated in figure 2.10.

At the top, the ME system level management provides abstraction of the bottom MEC system simplifying connections from UEs and third parties.

The ME host levels consist of two main parts; ME host and ME host level management. The ME host provides the virtualization infrastructure and ME platform, providing resources for running mobile edge applications.

The ME host management handles the management of functionalities of a particular ME host and the applications running on it.

The underlying network levels offers represent the MEC connectivity of the access networks; such as 3GPP mobile, Wireless Access Network (WAN), Local Area Network (LAN) and external ones such as Internet.

Figure 2.11, illustrates the MEC reference architecture, including its functional entities and respective reference points.

The ME host is an entity that contains the ME platform and a virtualization infrastructure

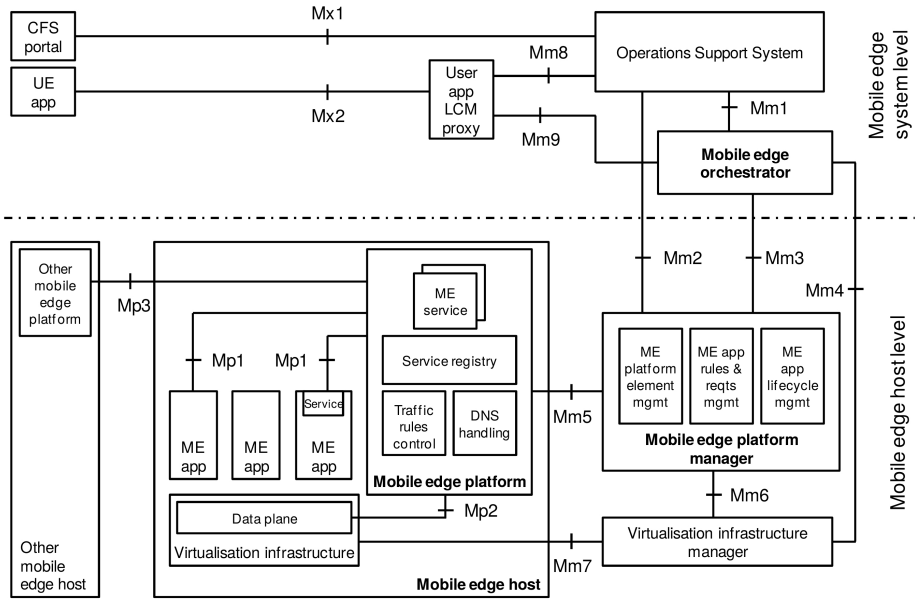


Figure 2.11: MEC reference architecture

Source: [76]

which provides compute, storage and network resources for the Mobile Edge Applications (APPs).

It also contains the ME platform which enables applications to discover, advertise, consume and offer mobile edge services, and provides the virtualization infrastructure with a set of data plane traffic rules. Those rules are based on policies received either from the ME platform management via Mm5 interface or via applications and/or services. The ME platform is also responsible for configuring a DNS proxy/server accordingly to DNS records received from the ME platform management, facilitate the user traffic to reach the desired ME application and/or communicate with other peer platforms via the Mp3 interface, which allows peer platform clustering. It is also in charge of providing access to persistent storage and time of day information.

The ME platform manager is responsible for managing the life cycle of applications, providing element management functions to the ME platform, via the Mm5 interface, and managing the APPs rules and requirements including service authorizations, traffic rules and DNS configuration. The ME platform manager also receives reports and performance measurements from the Virtualized Infrastructure Manager (VIM), via Mm6 interface, for further processing.

APPs are executed as VMs on top of the virtualization infrastructure and interact with the ME platform via the Mp1 interface in order to discover offered mobile edge services, indicate their service requirements and to perform APP migration when the UE moves around within the network.

The VIM is responsible for managing, allocating and releasing virtualized resources of the Virtual Infrastructure (VI) and to prepare the VI to run a software image. It connects with

the VI via interface Mn7. Openstack is considered the most widely adopted VI.

The ME orchestrator is the core functionality in mobile edge system level management; it has the visibility over the resources and capabilities of the entire ME network including a catalog of available APPs. The ME orchestrator is responsible for electing appropriate ME host(s) for application instantiation based on constraints, such as latency, available resources and available services. It interacts with the VIM, via Mm4 interface, for managing the APPs images, maintain records of available resources and status information and preparing the VIM to handle the applications.

OSS is an entity that receives requests via the Customer Facing Service (CFS) portal and from UEs applications for instantiation or termination of applications, and decides on the granting of these requests. Granted requests are forwarded to the ME orchestrator via Mn1 interface, for further processing. The OSS interacts with ME platform manager through Mn2 interface, for the ME platform configuration, fault and performance management.

The CFS portal allows access from third parties, providing mobile edge applications that meet their particular needs, and to receive back Service Level Agreement (SLA) or billing related information from the provisioned applications.

The user Application Lifecyle Management (LCM) proxy is a function that enables UEs applications to request Application (APP) related services such as instantiation and termination of user applications and when supported, relocation of user applications in and out of the ME system. For processing these applications requests, the user LCM proxy interacts with the OSS via Mm8 interface, and with the ME orchestrator via Mm9 interface.

2.6.4 Use cases and applications

The presence of the MEC on the edge of operator's network brings many advantages to all stakeholders, such as service providers, mobile operators and end users [57].

The next subsection discusses individual use case categories, scenarios, services and applications.

For a better analysis the individual use cases where divided in three main categories (see Figure 2.12), as was presented by ETSI [81].

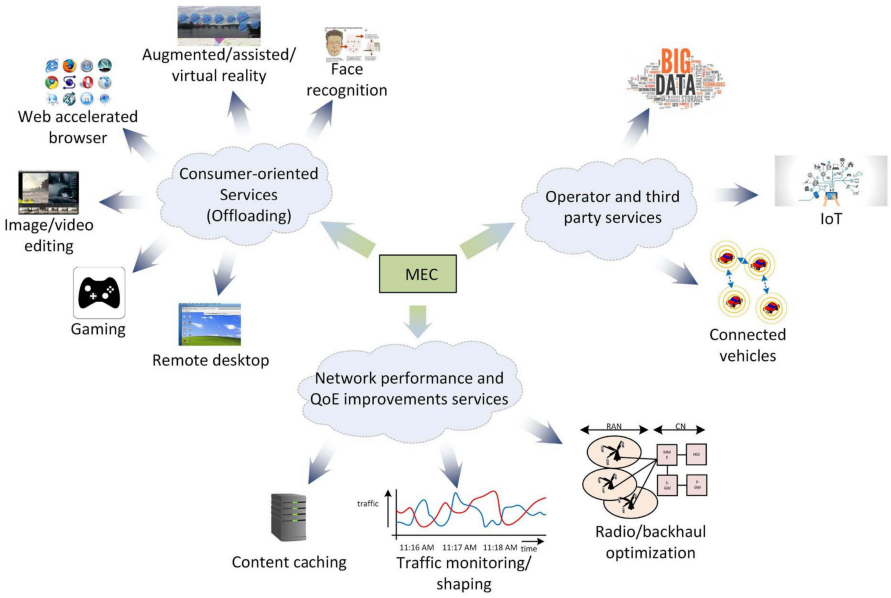


Figure 2.12: Example of MEC use-cases and Scenarios

Source: [57]

2.6.4.1 Consumer-Oriented Services

The first use case category generally benefits directly the end-user.

Mainly, users benefit from MEC by offloading intensive computation from mobile devices to nearby MEC servers, enhancing mobile devices computation capabilities and prolonging their batteries' lifetime, which enables running new and compute-intensive emerging applications at the UEs.

Compute offloading is a technique by which a mobile device, fully ou partially, offloads a computation-intensive task to a resource-sufficient cloud environment [73]. Computation offloading is performed mainly to save energy or due to the inability of the end device to perform heavy-compute tasks, or just to simplify the end devices.

One of the applications that benefits from computation offloading is Edge Accelerated Web Browsing [82], a web browser where most of the browsing functions are offloaded to MEC. Likewise, face/speech recognition or image/video editing also benefit from computation offloading as these require large amounts of storage and computation [83].

Because of the low latency and responsiveness, MEC infrastructures have also been recognized to be a niche for latency-sensitive applications in AR domain [84].

One AR application using MEC is shown by Doleza et al. [85]. The authors use an AR application that discovers places of interest visible in the view of the device's camera and overlay additional text information in the screen. With this implementation the authors demonstrate the reduction of latency up to 88% and energy consumption of the UE up to 93%.

Chen et al. [86] propose a cognitive assistance application using Google Glass with cloudlet. In their application Google Glass is used as a input device of information from the user (gps,

acceleration and video). The data is sent to a cloudlet server that has different subsystems running, such as object recognition, face recognition and motion classifier. Each subsystem is running on a separate VM on an enhanced Openstack platform.

Verbelen et al.[87] implemented an AR application featuring markless tracking and object recognition. The application tracks feature points in the video frames and shows 3D objects as an overlay. The authors split the application logic into several components and deploy each component based on real time requirements, whether to nearby edge server or central cloud.

Another example of AR application is Augmented Brain Computer Interaction Game presented by Zao et al. [88] based on Fog Computing and Linked Data. When a person plays the game, raw streams of data collected by Electroencephalography (EEG) sensors are generated and classified to detect the brain state of the player. Brain state classification is among the most computationally heavy signal processing tasks, and needs to be carried out in real-time. The system employs both MEC and cloud servers, a combination that enables the system to perform continuous real-time brain state classification.

Alongside with AR, Mangiante et al. [89] proposes using MEC to deploy a Virtual Reality (VR) application. In this implementation a Field Of View (FOV) rendering solution is presented, running on edge servers, designed to optimize the bandwidth and latency required by VR 360°video streaming. Although preliminary, test results show the immediate benefits in bandwidth saving using this approach.

Additionally, applications like gaming and remote desktop may benefit from MEC due to the low latency requirement [57].

Healthcare can also be aided by edge computing. According to Heindenrench et al. [90] one third of the strokes could possibly be averted by early mitigating the fall incidents. In order to prevent and detect a fall, Cao et al. [91] proposes a smart healthcare infrastructure called U-fall, that exploits smartphones by engaging edge computing technology. The application is based on a detection algorithm that is designed using acceleration magnitude values and nonlinear time series analysis. The propose infrastructure maintains integrity between the smartphone and the server to ensure real time detection [91] [92].

Furthermore, MEC can help health advisors to better assist their patients: MEC enables smartphones and sensors to collect data from patients (eg. pulse rate, body temperature, blood pressure, etc) with health advisers having access to the cloud server, being able to diagnose and assist them accordingly [93].

2.6.4.2 Operator and Third Party Services

The second category of use cases is represented by the services and scenarios from which operators and third parties can benefit.

An example of this use case exploits MEC for IoT purposes [94] [95]. Since IoT devices are connected through several radio technologies (e.g. Wi-Fi, LTE, Third Generation (3G), LoRa, etc) and using different protocols, there is a need for a low latency aggregation point where the messages can be processed, distributed and/or aggregated. A MEC platform can encompass a local IoT gateway functionality, capable of aggregating the messages from nearby

IoT devices and perform big data analytics for event reporting. Moreover, the MEC server can filter the communications performing data trimming before the corresponding information reaches the associated cloud server [96], sending only meaningful messages (for example when a sensor value changes more than a specified threshold). This will significantly reduce traffic on the network backhaul as well as communications and processing overhead on central clouds [97].

Sun et al. [94] propose an edge IoT architecture that considers a hierarchy of edge cloud platforms in order to provide flexible IoT services while maintaining user privacy. In the proposal, each user's IoT devices are associated with a proxy VM (located in an edge server), that collects, classifies and analyzes the device's raw data and transmits the metadata to the correspondent application VM (which is owned by the IoT service providers and used by user's in a shared manner). Since the metadata is generated from the raw streams it is not violating the user privacy.

Another application, known as IoTCloud [98], is a cloud-compatible open source controller and an extensible API, which enables developers to create scalable high performance IoT and sensor-centric applications.

Osmotic computing [99] proposes a new paradigm for the efficient execution of IoT services and applications at the network edge. Its design concept splits the application processing in three tiers, namely, IoT devices, edge servers and central cloud. Applications are decomposed and tailored into lightweight microservices deployed on an edge server or complex microservices deployed on a central cloud. Like *osmosis* in the chemistry context, the dynamic management of resources in the central cloud and edge is performed to achieve the balanced deployment of microservices while ensuring resource constraints and applications requirements. For microservices deployment, Osmotic Computing uses lightweight container-based virtualization technologies such as Docker [100] and Kubernetes [101], as a result it is easy for the operator to dynamically decide which microservices should migrate from edge to central cloud and vice versa.

MEC can also support a number of smart city services, including video analytics, location services, intelligent public spaces, public safety and emergency services, to mention a few. These services can easily produce a large amount of data, with location specific and latency requirements.

Sapienza et al. [102] presents a scenario that exploits MEC in order to detect abnormal or critical events such as terrorist threats, natural and man-made disasters. The MEC performs two services; Mash-Up Service that monitors and analyzes from information sources (personal devices like smartphones, video surveillance system deployed in the city and wireless air quality sensor system) and Alert Notification Manager that makes and sends notification messages to BTS and eNodeBs to rapidly notify the users which are close to the critical area.

Video stream analysis is another service scenario that benefits from MEC [97]. For example, suppose a vehicle license plate recognition service that monitors vehicles passing a certain area, checks if the vehicles are legal, and then sends the recognized plate number to the cloud and signals the authorities. The mechanism for the video analysis remains the same to the

one used on the cloud [103], but the amount of data that is sent to the cloud is negligible compared to the original video stream.

Moreover, MEC enables surveillance cameras to be spread across the city which can be beneficial for several applications such as traffic management, and/or detecting traffic accidents. The same cameras can be used with face recognition to trace missing persons or wanted criminals [104] [105]. Several vendors are working on this service scenario, too [106] [107].

The characteristics of MEC can also be exploited in the connected vehicles for Intelligent Transport System (ITS). An evolution in connected vehicles technologies is foreseen, vehicles will become more equipped with devices and applications that connect the vehicle to its surroundings, for example Interactive Advanced Driver-Assistance systems (ADASs) and Cooperative Intelligent Transport Systems (C-ITS). Connected-vehicle safety applications are designed to increase situation awareness and mitigate traffic accidents through Vehicle-to-vehicle (V2V) and Vehicle-to-infrastructure (V2I) communications [108]. One vehicle can exchange communications with other vehicles and inform them about road hazards, traffic jams, vehicles' next behavior, expected risk or even the presence of pedestrian and bikers. This information is frequently locally usable, so deploying these services on a local MEC has significant advantage, especially for low-latency critical communications, by adding computation and geo-distributed services to roadside BTS. Regarding this scenario, a lot of business opportunities and new value-added services are expected, such as unoccupied parking location and car finder [109], thus the industries also notice MEC-enabled connected vehicles services and some consortium had been instituted [110] [111] [112] [113]. Some comprehensive surveys which are specific to connected vehicles' MEC-enabled deployment are available [114] [115]. In addition to the common connected vehicles systems (automobiles, buses, trains, etc), MEC will also be useful to enable connected Unmanned Aerial vehicles (UAV)s, which play an increasingly important role in several scenarios such as critical missions, emergency response, inspection and monitoring.

In 2016, Nokia proposed the UAV traffic management (UTM) [116] based on MEC architecture, where the UTM provides automated UAV missions, fleet management, 3D navigation and collision avoidance. Despite the effort, the existing mobile networks are mainly designed to the ground user, thus UAVs will have limited connectivity and bandwidth, hence it is necessary to recompose the mobile networks and deploy MEC servers to guarantee the connectivity and low latency required by the UAVs. Other less common scenarios can benefit from MEC deployment such as smart grids, Wireless Sensor and Actuator Networks (WSAN)s, ocean monitoring and smart building control.

2.6.4.3 Network performance and QoE Improvement Services

The third category of use cases and scenarios are those relating with optimizing network performance and/or improving QoE. One such case is to alleviate congested backhaul links by local content caching at the edge. Providing a distributed caching by extending Content Delivery Network (CDN) services toward the mobile edge can heighten user QoE, while

alleviating congested backhaul and core network usage, as confirmed in several research works [117] [118]. Several architectural solutions to integrate distributed parallel edges servers, capable to perform video streaming and cache, are presented in [119], [120] and [74].

Media Cloud [121] proposes a solution to deliver on-demand adaptive video streaming services, where those resources can be dynamically scheduled in an on-demand fashion. Liu et al. [122] present a context network-aware with edge caching capabilities based on the prediction of content popularity, user preferences and the characteristics of the local wireless environment.

MEC can also increase QoS, especially for heavy objects like 3D images, by caching content locally (e.g. museums, shopping centers, stadiums) without requiring centralized servers and core network resources [123].

Besides alleviation and optimization of the backhaul network, MEC can also enhance the radio network, for example, by gathering related information from the UEs and processing these at the edge, resulting in a more efficient scheduling.

2.6.5 Deploying Multi-Access Edge Computing in 3GPP networks

The 5G standards are currently under development by 3GPP, however 3GPP has already included MEC architecture into its 5G standards in a technical specification document [51]. Additionally MEC is recognized by the European 5G Public-Private Partnership Association (5GPPP) research body as one of the key emerging technologies for 5G networks (together with NFV and SDN) being identified as a natural development in the evolution of mobile BTS and the convergence of information technology and telecommunications networking [124].

The basic performance criteria for 5G systems has been set by the ITU in their International Mobile Telecommunication (ITM)-2020 Recommendation [125].

ITU-R M.2083 describes three overall usage scenarios for 5G systems:

- Enhanced Mobile Broadband to deal with hugely increased data volumes, overall data capacity and user density;
- Massive Machine-type Communications for the IoT, requiring low power consumption and low data rates for very large numbers of connected devices;
- Ultra-reliable and Low Latency Communications to cater for safety-critical and mission critical applications.

To achieve these visions, 5G systems will exploit the MEC as one of the innovative technologies and as being an essential tool to facilitate the smooth transition from fourth generation 4G networks to the new generation [126].

Opposed to the ongoing mobile network architecture, the 5G platform was conceived to allow a more flexible deployment of the data plane, intending to natively support MEC.

Since MEC underlying connectivity of the access networks, along with the hardware of the MEC platform, remains open, new levels of flexibility to choose deployment scenarios are enabled. Therefore, Service Providers (SPs) can deploy MEC to work as early applications test beds, enabling SPs and third parties means to trial their applications, without waiting for full approval of the 5G standards, with the full capital investment. For example, SPs

can host applications on a MEC test bed, test the revenue return, and scale up or remove as appropriate [127].

Another aspect for early deploying MEC platforms is the possibility for re-using existing deployed systems. Due to the virtualized and flexible characteristics of MEC, it is effortless to monitor performance and resources needed, which enables more rigorous pricing control for SPs regarding application providers for hosting the applications as well as dimensioning the edge equipment accordingly with the application set proposed [127].

One solution to early deployment MEC platforms presented in an ETSI white paper [127], released in 2018, contemplates a distributed SGW-LBO deployed on the edge site, whereas the control plane functions of the 3GPP Evolved Packet Core (EPC) components (as specified in the Forth Generation (4G) system architecture in ETSI TS-123.401 [128]), such as Mobility Management Entity (MME) (the key control-node for the LTE access-network which provides the control plane function for mobility between LTE and 2G/3G access networks) and Home Subscriber Server (HSS) (a central database that contains user-related and subscription-related information) are located at the operators core site. Local breakout at the S/P-GW is a new architecture for MEC that originates from the operators' desire to have a greater control on the granularity of the traffic that needs to be steered. The S/P-GW is an entity that combines both Serving Gateway (SGW) (responsible for forwarding data packets, and serve as the mobility anchor between LTE and other 3GPP technologies) and Packet Data Network Gateway (PGW) (responsible for providing connectivity from the UE to external packet data networks by being the point of exit and entry of traffic to the UE). This principle is dictated by the need to have the users able to reach both the MEC applications and the operator's core site application in a selective manner over the same Access Point Name (APN) [127]. For traffic steering the SGi - Local Break Out interface is used, which supports traffic separation and allows the same level of security as the operator expects from a 3GPP-compliant solution.

The diagram illustrated on figure 2.13 describes the co-locating MEC hosts with the SGW in a mobile network where the MEC system and the distributed SGW are co-located at the edge.

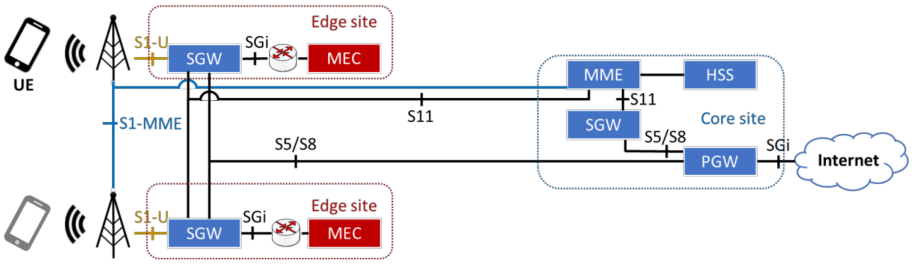


Figure 2.13: SGW-LBO MEC deployment

Source: [127]

So, starting out with a 4G plus an edge test bed, with limited deployments at first, MEC empowers a smooth transition into the 5G network roll out, without the need for major upgrades when the time for transition arrives.

2.6.6 Network Slicing

Network slicing has emerged as a key concept for providing an agile networking platform to support emerging businesses with different service requirements in an efficient way [129] [130]. It consists in slicing one network into multiple instances, each architected and optimized for a specific requirement and/or application/service. Network slicing introduces a multi-tenant environment capable of supporting flexible provisioning of network resources, as well as dynamic assignment of network functions, Radio Access Technologys (RATs) and applications. Network slicing enables resource sharing among virtual MNO, services and applications as developed in [131] considering 3GPP mobile networks, by introducing the notion of network slice broker that complements the network sharing management. From the infrastructure perspective, network slicing allocates a set of dedicated or shared resources, either virtual or physical, to particular tenants by introducing a network hypervisor. To accommodate the service requirements of incoming requests, network slices need to combine a set of network and cloud resources, such as network functions, processing power, storage and access to big data or RAN analytics, etc., which are typical MEC utilities. Figure 2.14 illustrates an example of network slicing on a common network infrastructure considering the potential role of MEC in mobile broadband, automotive and massive IoT services [73].

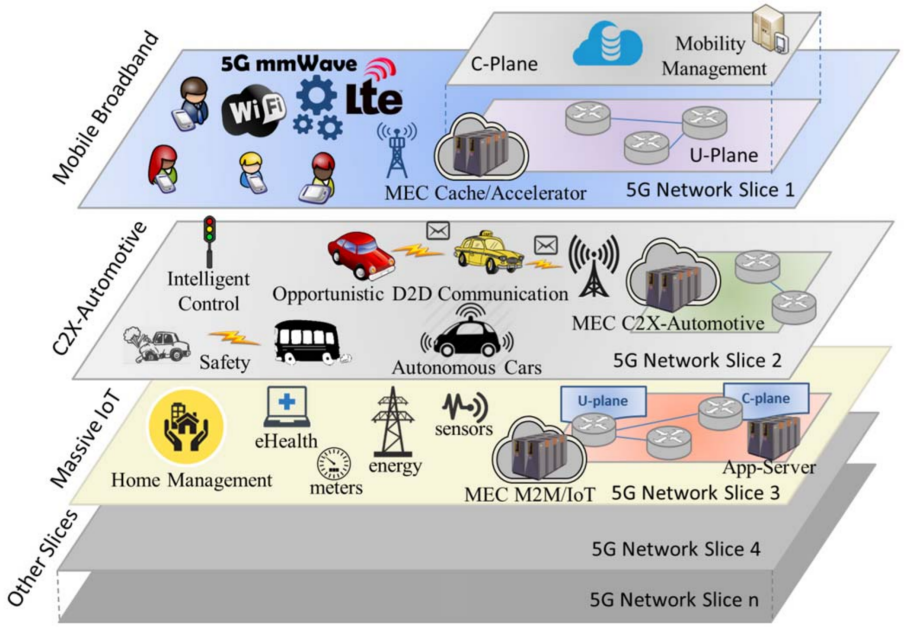


Figure 2.14: Example of network slicing and the role of MEC

Source: [73]

Since mobile broadband requires high capacity, a MEC platform can cache content at the edge decreasing the load on the mobile backhaul and reducing core network traffic by offloading the traffic to the local edge. MEC can also provide a number of services to enhance mobile broadband capabilities such as video acceleration or application aware performance optimization. For the automotive slice, MEC is a catalyst element that shapes

the capabilities needed to accommodate strict latency and scalable network functions and applications instantiated at the edge. Regarding IoT communications, MEC can provide processing and storage resources to efficiently handle huge amounts of small data.

2.6.7 MEC Deployment

MEC provide resources, usually located on a centralized remote cloud, to be scattered among a set of multi-cloud platforms. A straightforward deployment of MEC is as a separate and individual platform, allowing a Mobile Network Operator (MNO) to integrate it into the RAN to provide local services without considering service continuity and user mobility. A more breakthrough deployment scenario is a set of MEC heterogeneous environment platforms, providing different services, taking into account network and traffic conditions, and supporting user and service mobility [73]. The first release of the ISG MEC will support deployment scenarios where the MEC server is deployed either at the LTE macro base station (eNodeB (eNB)) site, or at the 3G Radio Network Controller (RNC) site, or at a multi-technology (3G/LTE) cell aggregation site [60]. An overview of the different MEC deployment scenarios is illustrated at figure 2.15.

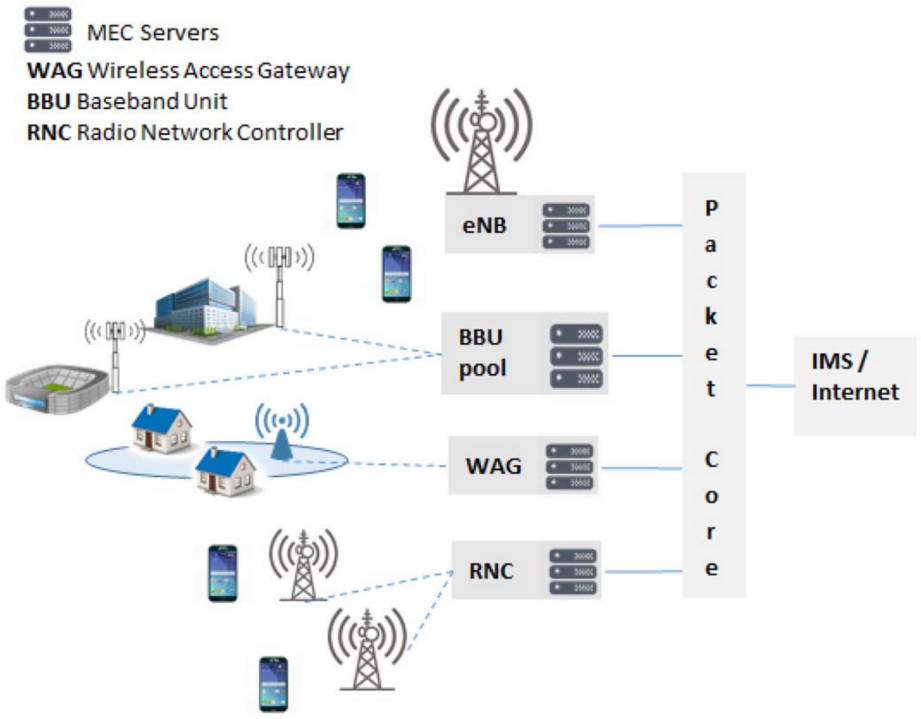


Figure 2.15: Example of MEC deployment Scenarios

Source: [73]

For outdoor regions, MEC can be deployed directly at the RAN or in close proximity, enabling a close coordination of applications with the RAN, understanding traffic and radio conditions, providing a flexible service while enabling computing and storage at the RAN edge [73].

The MEC platform can reside on macro-based station sites, such as eNB in LTE networks, or at RNC of a 3G mobile network. MEC can also be deployed directly at the mobile backhaul, such as a small cell gateway, or deployed at an aggregation point such as Baseband Unit (BBU).

For indoor environments, MEC can act as a powerful gateway, enabling multiple services within a particular location, such as security in public spaces, augmented reality on museums, video stream on sports and social events or empower social network applications [73].

Considering IoT applications, the research work in [69] explored the deployment of edge cloud platforms at a cell aggregation point and at a WiFi access point. A seamless integration of edge-cloud platforms in a small cell deployment without any impact on the operations of 3GPP LTE networks is detailed in [132] and [133], forming a cluster of interconnected computing resources. In these solutions, mobile users transmit cloud and conventional data over the local gateway Local Internet Protocol Access (LIPA), which is responsible for encapsulating the packets and then send them to the edge cloud platform.

Selection of MEC server deployment depends on many factors, such as, scalability, physical deployment constrains, performance required, low latency, economics, etc. hence, each deployment must be planned to fulfill its purpose.

2.6.7.1 Multi-Access Edge Computing Deployment Solutions

2.6.7.1.1 Small Cell Cloud (SCC).

The SCC was proposed in 2014 [133], and the main idea was to enhance the network with small cells (Small Cell eNodeB (SCeNB)s), like microcells, picocells and femtocells, with computation and storage capabilities. A similar idea was address later on SESAME [134] [135] project.

Because of the requirements of the next generation mobile networks, a higher number of SCeNB is supposed to be deployed, as a result SCC should be capable of providing enough computation power for the UEs, especially for applications and services that require low-latency (see examples on section 2.6.4).

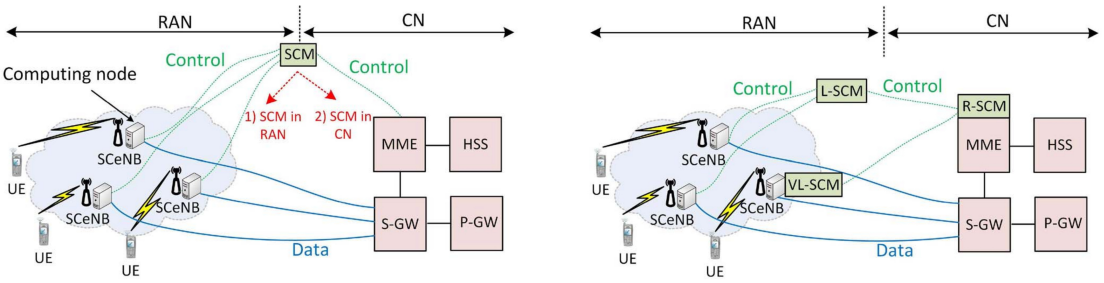
To fully integrate SCC in the mobile network architecture a new entity must be created to fully control and manage the SCeNBs, the Small Cell Manager (SCM).

The SCM is aware of the overall cluster context (both radio and cloud wise), thus it is capable of perform dynamic and elastic management of the computation resources within the SCC, deciding where to deploy or migrate computational power, optimizing the service delivery for the end-user. The computation resources are virtualized in a VM located on the SCeNBs and they can pool their resources exploiting NFV.

In the SCC architecture the SCM can be deployed either in a centralized manner, located on the Core Network (CN), or deployed within the RAN, close to a cluster of the SCeNBs or as an extension of MME.

Furthermore, the SCM can be deployed in a distributed hierarchical manner, where a Local Small Cell Manager (L-SCM) or a Virtual Local Small Cell Manager (VL-SCM), located on the RAN, manages the resources on the correspondent SCeNBs, while a Remote Small

Cell Manager (R-SCM), located on the CN, manage all the resources of all SCeNB connected to the CN as displayed on figure 2.16.



(a) Centralized SCM architecture **(b)** Distributed hierarchical SCM architecture

Figure 2.16: SCC architecture (Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (SGW), Packet Data Network Gateway (PGW)

Source: [133]

2.6.7.1.2 Mobile Micro Cloud (MMC).

The MMC concept was introduced in [136]. The main difference between this concept and the SCC is that the MMC does not introduce any control entity in the network, as it is assumed to be fully distributed in a similar way as the VL-SCM in SCC.

While in the SCC the resources are provided by interworking clusters of SCeNBs, in MMC the UEs exploit resources from a single MMC.

In order to maintain service continuity if the UEs move within the network, the MMCs are interconnected between them and connected through the backhaul, to guarantee fast and smooth service migration between MMCs.

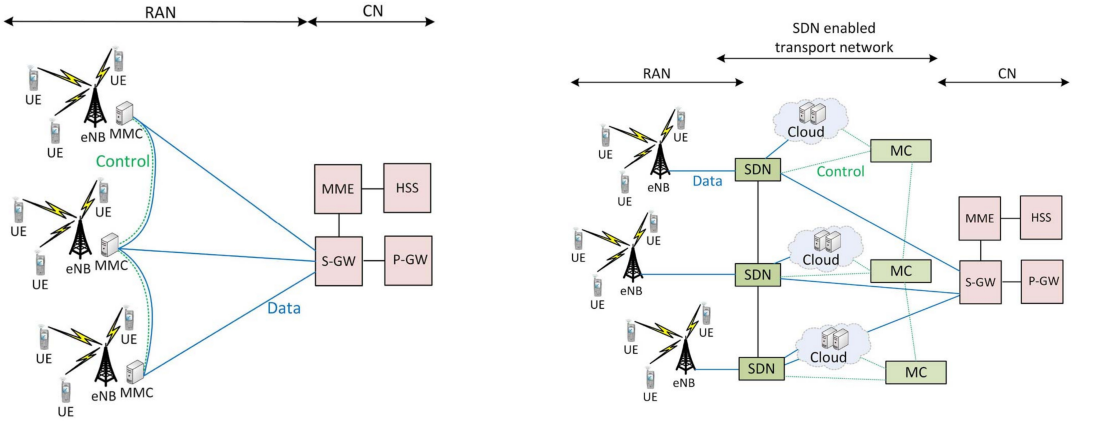
2.6.7.1.3 Fast Moving Personal Cloud (MobiScud).

The MobiScud architecture [137] integrates cloud services in the mobile networks by exploiting SDN and NFV technologies while maintaining backwards compability with existing mobile networks. MobiScud enables personalized virtual machines to seamlessly "follow" UEs as they move throughout the network.

When compared with the MMC and the SCC architectures, the cloud resources in MobiScud are not deployed directly on the access nodes like SCeNBss and eNBs, but in centralized clouds within the RAN or near the RAN. However, these clouds are assumed to be highly distributed, as in the MMC and SCC concepts, providing cloud services to all network UEs.

The Mobiscud concept also introduces a new control entity, a MobiScud Controll (MC), which basically has two functionalities: a) monitoring signaling messages between mobile networks elements to be conscious of the UEs activity and b) orchestration and routing data traffic within the SDN-enabled transport network to aid compute offloading and storage applications and to perform VM migration when the UE moves around within the network.

2.6.7.1.4 Follow me Cloud (FMC).



(a) MCC architecture

Source: [136]

(b) MobiScud architecture

Source: [137]

Figure 2.17: MCC and MobiScud architecture

Similar to the MobiScud concept the main idea of FMC [138] [139] is to have cloud services running VMs deployed on distributed Data Centers (DC) that follows the mobile users as they move throughout the network, but in contrast to the other concepts, the cloud services are deployed farther from the users, on the CN of the operator, the FMC architecture is displayed on figure 2.18a.

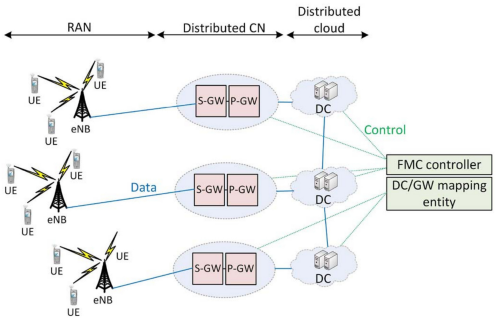
The FMC concept implements new entities in the network: a DC/GW mapping entity and an FMC controller (FMCC). The DC/GW mapping entity maps the DCs to the distributed S/P-Gateways (GWs) according to distinct metrics, such as, location or hop count between the DC and distributed CN. The FMCC manages DCs resources, such as computation and storage and cloud services running on them. The Follow me Cloud Controller (FMCC) is also responsible for selecting which DC should be associated to the UE using the cloud services. These two entities can be deployed either centrally or locally for better scalability as displayed on figure 2.18a.

While in the previous concepts the authors assume the existing centralized CN deployment, FMC leverages from the fact that mobile operators need to decentralize their networks to cope with the future mobile networks requirements, thus it is assumed that the centralized CN used on network deployment will be replaced by a distributed one, and the DC may be located in the same place as the centralized PGW and SGW.

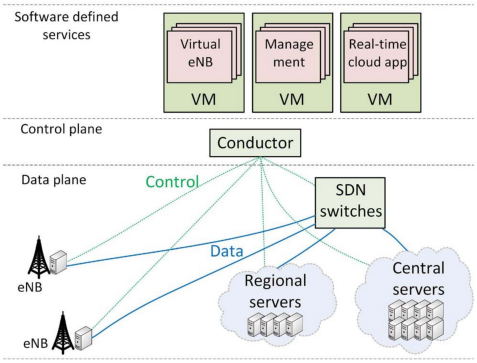
2.6.7.1.5 CONCERT.

CONCERT [140], like the above-mentioned solutions is assumed to exploit NFV and SDN technology.

The control plane essentially consists of a new entity, conductor, deployed centrally or in a hierarchical manner for better scalability, which manages communication, computing and storage resources in CONCERT architecture. This entity can be deployed centrally or in a hierarchical manner for better scalability like SCC or FMC.



(a) FMC architecture
Source: [138] [139]



(b) CONCERT Architecture
Source: [140]

Figure 2.18: FMC and Concert architecture

The data plane consists of Radio Interface Equipments (RIEs) physically representing the eNB, SDN switches and computing and storage resources (see Figure 2.18b).

The computer resources are both for baseband processing (similarly as Centralized Radio Access Network (C-RAN)) and for handling application level processing (e.g. application offloading, AR). In his concept, local servers with low resources are assumed to be located near the users (directly at the physical BTS) and if compute power and/or storage proved to be insufficient, regional and even central servers are exploited.

2.6.8 Orchestration Options

As new networks progressively incorporate different technologies and cloud infrastructures, becoming more heterogeneous in nature, the resource allocation and management processes are becoming more complex. On top of such a heterogeneous environment, new requirements on distributed service provisioning, programmability and multi-tenancy support leads the network and cloud control approaches towards a more unified orchestration. Such orchestration should take into account networking, cloud and service requirements. Currently, a number of different orchestration deployment options have emerged from the industry and standardization, with the most significant ones detailed below.

- OpenBaton [141]: OpenBaton, developed by Fraunhofer FOKUS and TU Berlin, ensures the development of virtual network infrastructure by adapting network functions to the specific cloud environment, providing a comprehensive implementation of the ETSI NFV MANO specification. The framework considers a generic Virtual Network Function Manager (VNFM) for the life cycle of the VNFs, based on the corresponding descriptors, and a Juju[10] (a service orchestration tool for the cloud) VNFM adapter in order to deploy Juju charms. Openbaton integrates two different engines: i) event management engine for dispatching and ii) auto scaling engine for managing scaling operations. A fault management system is also included for automatic run-time management where

[10] https://www.ubuntu.com/cloud/juju

monitoring information is gathered using Zabbix[11](open source monitoring solution for network and application monitoring). Finally, it provides plugins for addition and deletion inside the orchestration logic.

- OSM [142]: Open Source Management and Orchestration (OSM) is an ETSI-hosted project to develop an Open Source NFV MANO software stack aligned with ETSI NFV. The framework offers SDN underlay control (integrating multiple SDN controllers), multi-site capability and multi-VIM capability with enhanced performance awareness. The architectural components contain: resource orchestrator (from the Telefonica discontinued OpenMano), VNF configuration component (Canonical Juju), network service orchestrator and Graphic User Interface (GUI) (RIFT.io[12]), virtualized infrastructure based on intel architecture, virtual infrastructure manager (OpenVIM[13] and Openstack[14]) and finally service VNFs (Metaswitch[15] and 6wind[16]).
- Cloudify [143]: Cloudify is an open source framework based on Topology and Orchestration Specification for Cloud Applications (TOSCA), which acts as a cloud platform orchestrator. It provides a complete solution for automating and managing application development and DevOps processes on top of a multi-cloud environment. Cloudify eliminates the boundaries between orchestration and monitoring, assuring automatic reaction to pre-defined events with the appropriated corrective measures. It organizes workflow for environment setup, application installation, infrastructure management, scaling and fault recovery. Cloudify offers interoperability among diverse cloud platforms (e.g, VMware, Cloudstack, Amazon and Azure) and reduces multi-vendor lock in. A CLI based client is used to perform the different operations.
- M-CORD [144]: M-CORD is a cloud-native solution built on SDN, NFV and cloud technologies. It includes both virtualization of RAN functions and a virtual Evolved Packet Core (vEPC) to enable mobile edge applications and innovative services using a micro-services architecture. M-CORD enables virtualization of the RAN and core network functions, while separating the control functions from the data plane enabling a unified network orchestration and managment. Moreover, it allows third parties to build mobile edge services facilitating localized applications. M-CORD offers a single SDN control plane following Open Network Operating System (ONOS) [145] to control the virtual network infrastructure, SDN and NFV resources based on openstack and TOSCA facilitating the deployment of VNFs and network slices, providing mobile services with the desired performance, orchestrated by XOS [146].
- T-NOVA [147]: T-NOVA is a management and orchestration platform for automated provisioning of Network Function as a Service (NFaaS) on top of virtualized infrastructures. It leverages the benefits of SDN and cloud management architectures to enable automated provisioning, configuration, monitoring and efficient operations of VNFs.

[11]https://www.zabbix.com/
[12]https://riftio.com/
[13]https://github.com/nfvlabs/openvim
[14]https://www.openstack.org/
[15]https://www.metaswitch.com/
[16]http://www.6wind.com/

T-NOVA differs from the other frameworks in terms of an additional marketplace layer, which allows operators to offer their infrastructures as a value added service. This layer is placed on top of the orchestrator and contains a customer facing module for implementing business related functionalities in a multi-user setting, employing the paradigm of "APP-Store". T-NOVA follows the ETSI NFV architecture separating the VIM from the NFVI, which are based on Openstack and OpenDaylight. The orchestrator divides its functionalities into two modules, namely Network Service Orchestrator (NSO) and Virtual Resource Orchestrator (VRO). NSO maintains the lifecycle of the network services focusing on connectivity, and VRO manages compute, storage and network resources.

- ONAP [148]: Open Network Automation Platform (ONAP) is an open source orchestrator project, carried out within the Linux Foundation with the support of AT&T, China Mobile and other leading industry partners. It is an initiative created by the combination of the Enhanced Control, Orchestration, Management and Policy (ECOMP) and Open Orchestrator (OPEN-O) projects, into ONAP, to bring the capabilities for designing, creating, orchestrating and handling of the full lifecycle management of VNFs, SDNs, and the services that all of these things entail. ONAP allows the end users to connect products and services through the infrastructure, and allows deployments of VNFs and scaling of the network, in a fully automated manner. ONAP expands the scope of ETSI MANO, introducing the notion of the resource controller and policy component as well as the concept of resource description, i.e., meta-data, for lifecycle management of the virtual environment enabling network agility and elasticity, while improving the time-to-market. It follows a hierarchy of three orchestration modules consisting of: i) the Global Service-Orchestrator that enables end-to-end service composition and delivery, ii) the NFV-Orchestrator responsible for NFV orchestration, considering diverse VNFs across a wide range of VNFMs and VIMs and iii) the SDN-Orchestrator that provides network connectivity and traffic steering via the means of different SDNs controllers (e.g., OpenDaylight and ONOS), and/or the conventional element management systems. ONAP adopts TOSCA, YANG and Network Configuration Protocol (NETCONF) data models, Representational State Transfer (REST) APIs, Openstack and supports resource abstraction over diverse SDN, NFV and legacy networks, allowing a set of common services including policy management, security and other management capabilities.

2.6.9 Testbeds and Trials

This section lists some developed MEC-enabled testbeds.

- The 5th Generation Test Network (5GTN) was developed and tested in Finland, and it is based on LTE and Long Term Evolution Advanced (LTE-A) technology [149]. This test bed aims to provide application developers to experiment their APPs with a carrier grade test network, supplied with a number of measurement and monitoring tools in different parts of the network. It is composed by two distinctive environments, located at VTT Technical Research Center of Finland (VTT) 5G Laboratory and at the University

of Oulu's Centre for Wireless Communications (CWC). While CWCs network is targeted to be an open test environment with public users, VTT's network provides a more private and configurable environment, which facilitates more confidential research and testing. Despite the fundamental difference in the usage, the networks are interconnected, which allows bringing some functionality from the open network to the private network, and vice versa. Both networks are based on carrier-grade technologies in order to create a realistic environment. The MEC functionality is based on a Nokia provided solution that is operating in an Airframe cloud environment. It allows third-party service providers to bring their services and service-specific functions close to users through standardized interfaces and an open architecture [149].

- China and Nokia Mobile successfully tested an advanced mobile solution, to demonstrate that the solution is capable to deal with the high-speed, high-bandwidth demands generated by big sporting and entertainment events [150]. The testbed was deployed in Beilun Stadium in Ningbo, China where 11707 active users were simultaneously connected with small cells [151] and 6195 users connected with macro cells. In total, 95 LTE small cells were installed. The platform was built for MEC with Airframe [152] radio cloud platform [59]. UEs were receiving data at speeds of 12 Megabits per second (Mbps) and upload speeds improved by 62% compared to existing Time Division Long Term Evolution (TD-LTE) networks in high-traffic locations, the power efficiency of their devices was also improved by up to 33% [150].
- Nokia and Chunghwa Telecom (CHT) (Taiwan) implemented a test bed at a baseball stadium streaming live to spectators, allowing them to experience Television (TV)-like live coverage with multiple streams while absorbing the stadium atmosphere [153]. the MEC enviroment was created with the help of Nokia Flexi Zones BTS that use 30MHz of LTE spectrum. The deplyed MEC architecture enables spectators to see four video feeds on a split screen or select one for a close-up view.

2.6.10 MEC Security and Privacy Issues

In opposition to traditional centralized cloud computing, MEC introduces significant security risks especially when it is deployed at BTS or at areas where it is relatively vulnerable to physical attacks. Thus it is imperative to consider special security measures against on site attacks. MEC also demands more strict policies as third party stakeholders can gain access to the platform and collect information regarding user proximity and radio analytics. Isolation between different parties is another critical issue: a security attack on a particular application should not affect other running applications. Fine-grained access control needs to be investigated with appropriate encryption to ensure a secure collaboration and interoperability between heterogeneous resources and different operational parties [73]. An analysis on security threats and challenges associated with MEC is performed in [154], a study on MEC security is presented on [155] and a state of the art study is performed on [156].

2.7 Summary

MEC is an emerging paradigm that brings forward the technical benefits of the edge-cloud computing combined with SDN and NFV technologies, allowing third parties to provision applications and services on-demand through standardized APIs.

MEC is recognized as one of the key emerging technologies for 5G systems, thanks to its significant contribution to low latency assurance and capacity enhancements in the backhaul and core networks. The success of MEC fundamentally hinges on the alignment of the technology with ETSI NFV ISG for the proper definition of management and orchestration system with respect to the service elasticity and life-cycle management, service mobility as well as regarding joint optimization with the network resources.

Currently, MEC brings forward a range of different challenges that are yet to be solved. However, considering its potential, it is obvious that MEC will significantly uplift the shape and experience of mobile communications.

The next chapter introduces a design overview of the proposed solution for the MEC server architecture followed by the implementation details of each entity, and network functions deployed. Lastly the details of implementation of different use case scenarios to test the infrastructure will be presented.

CHAPTER 3

Design and Implementation

This chapter presents the design and implementation of the proposed MEC solution, as well as the design and implementation of use case/scenarios deployed on the implemented architecture.

3.1 Design

Figure 3.1 displays an overall view of the proposed solution. The architecture can be split in two different parts: the MEC server and the access network. The MEC server is where resources are available, such as, compute, storage and network functions. The VMs that will provide services and applications are instantiated on the MEC servers, as well as the VMs containing the entities to manage the access network and the access to the services VMs. These entities are the DHCP server that is responsible for attributing the IP addresses to the UEs connected to the AP and the AAA server which is responsible to authenticate these same mobile devices. It is on the MEC servers that the S/P-GW and the entity which controls them (Edge SDN Controller) is deployed. The S/P-GW is responsible to handle the traffic from the APs and eNBs and redirect it to the services VMs, or to the core S/P-GWs. This entity is also in charge of the encapsulation and decapsulation of the traffic flowing from the eNBs or APs to the VMs and vice-versa. The edge SDN controller is the entity responsible for managing the decisions of the S/P-GW.

The access network is composed by the AP and the eNBs scattered in the network. The AP is composed by two different wireless networks, one to attend the UEs traffic, acting as an AP for SPs and the other one providing access as a residential gateway. All of these wireless networks are connected to the backhaul using SDN capabilities and the ensuing advantages that this new technology delivers. Adopting this kind of APs the SPs could have a more distributed wireless network, allowing them to offload traffic from the conventional LTE antennas, accompanying with higher bandwidth, latency and coverage.

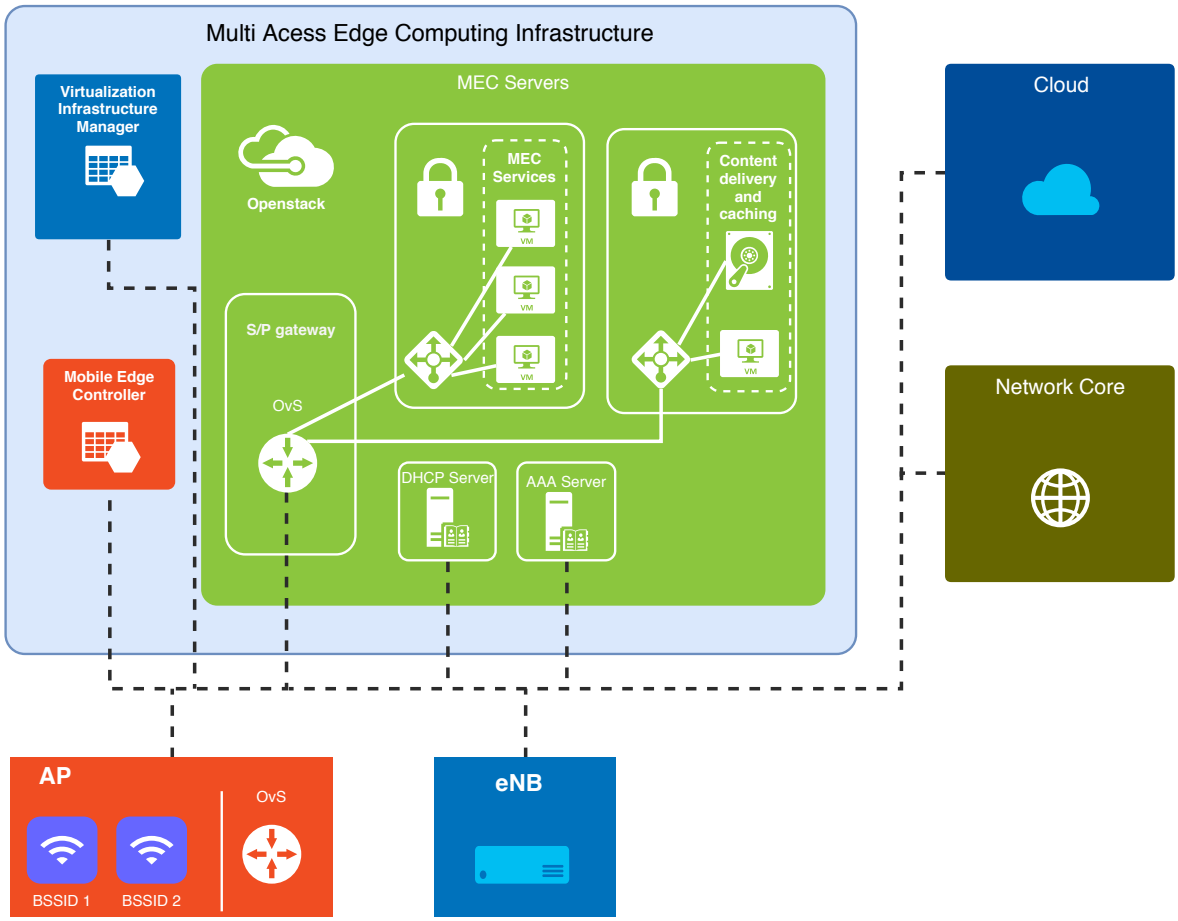


Figure 3.1: Multi Access Edge Computing Deployed Architecture

The eNB was already implemented in a parallel ongoing master thesis so it is not the subject of the work on this thesis. Nevertheless, the S/P-GW deployed on the MEC server has to be able to cope with the traffic from the eNB to the MEC services.

To test and evaluate the proposed architecture some applications were deployed to simulate use case scenarios usually associated with future MEC deployments. The use case scenarios to test on this infrastructure are:

- Mobile Code Offloading - the mobile device sends the code to compute on the edge server instead of computing it locally;
- Video Streaming/AR - A video is sent to the edge from the mobile device, is processed and sent back to the mobile device;
- Cache - The MEC server acts as cache were the contents from webpages are stored;
- Face Recognition - The video is sent from the mobile device to the edge for face recognition since the mobile device has limited processing power for this application.

3.2 MEC Architecture Implementation

The overall MEC architecture is illustrated in figure 3.2. The proposed MEC architecture was deployed in a Openstack environment (pikes release), using three physical nodes with nova,

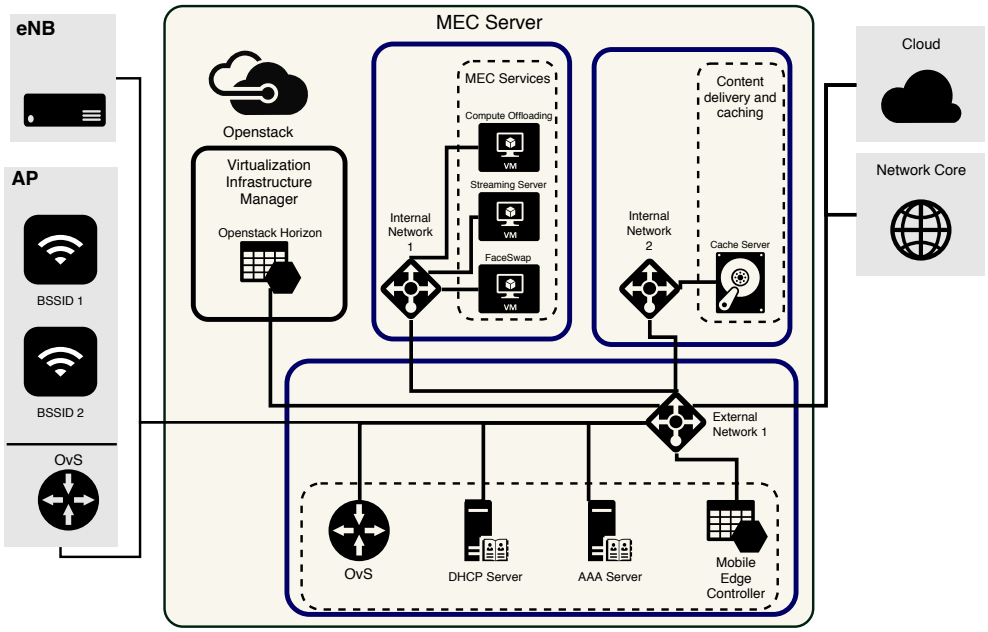


Figure 3.2: Deployed MEC architecture

neutron and cinder modules installed. The physical machines were equipped with Intel Xeon processors E5-2620 v4 and 1 Gbit/s network cards. On the first physical machine, equipped with 250Gb of Random Access Memory (RAM), the Cinder module was deployed, on the second machine with 500Gb of RAM the Nova module was deployed, and in the third and last machine equiped with 1Tb of RAM the Nova, Glance and Neutron modules were deployed.

Within the Openstack environment several virtual machines and networks were deployed according to figure 3.2. Although the figure only refers to four MEC applications VMs (FaceSwap, Mobile Code Offloading, Video streaming server and Web Caching), other applications can be deployed in parallel.

This architecture follows the distributed S/P-GW concept described in the ETSI white paper [157] for 4G access which, in this architecture, relates only to the data plane of the S/P-GW. For the future 5G, the S/P-GW becomes the User Plane Function (UPF). The S/P-GW provides a way to dynamically steer data plane traffic inside the edge to the different VMs where MEC applications are running. An SDN controller application was developed and it was called Edge Controller. This SDN controller is composed by two interfaces: the Northbound and the Southbound interfaces. The northbound interface provides APIs to applications while the southbound interface is used to manage the forwarding rules of the S/P-GW or UPF. In addition, two more VMs where instantiated: one for running an authentication server and the other one for the DHCP server deployment. The following sections present the implementation details of each of the network functions and applications indicated in figure 3.2.

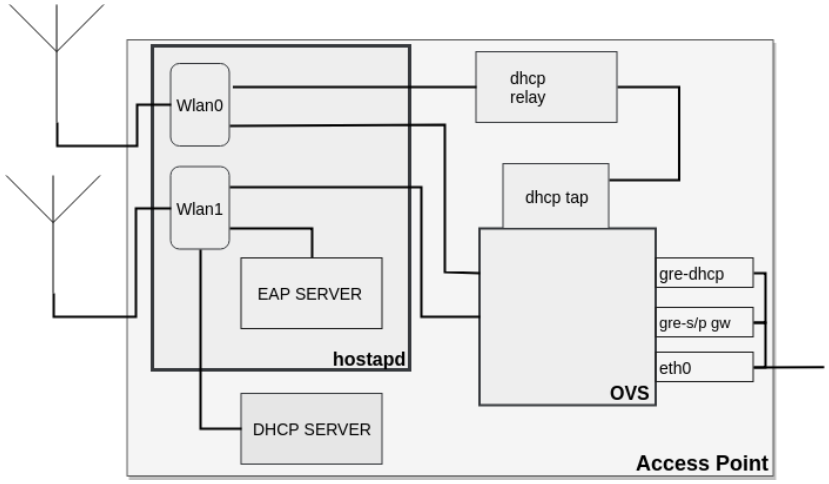


Figure 3.3: Deployed AP architecture

3.2.1 Wi-Fi AP

The base hardware used to deploy the Wi-Fi AP was an apu1d[1] with an AMD based CPU (with 2 CPUs) and 4 GB RAM running Ubuntu 14.04.4 LTS OS with a 4.2.0-34-generic kernel. The wireless network card connected to this device through miniPCI express was a wle200nx[2]. In order to setup this device as a Wi-Fi AP, the Host Access Point Daemon (hostapd)[3] user space software was used and configured to provide two different wireless networks: one using internal authentication through Wi-Fi Protected Access (WPA)-Pre-Shared Key (PSK) and an internal DHCP server installed on the AP, acting as a typical residential gateway, and the other one was configured to use EAP-AKA authentication through an external AAA server using the RADIUS protocol and an external DHCP server to attribute IP addresses to the UEs connected through Wi-Fi. Both the DHCP server and the AAA server, responsible for the EAP-AKA authentication, were deployed within the MEC servers. Regarding to the radio interface, hostapd was configured to use in both wireless networks the 802.11g protocol, which uses a radio frequency of 2.4 GHz with an expected throughput of 54 Mbps.

To handle user traffic, a virtual switch (OvS version 2.9) was installed in this device and an OvS bridge was created. The physical Ethernet port of the AP (eth0) was added to the OvS bridge, and the OvS bridge was configured to have the same Media Access Control (MAC) and IP address of the physical port. The created bridge was configured in the device as the default gateway, meaning that all traffic trying to leave the device will enter the bridge and can be processed as described in the switch's flow entries. Knowing that all the user traffic will enter the bridge, a GRE tunnel port towards the edge S/P-GW was created to encapsulate all the traffic authenticated by both EAP-AKA and WPA-PSK. This tunnel will carry user plane traffic from the Wi-Fi AP to the edge S/P-GW. The default flow entry for this device is to forward all outgoing UE traffic to the GRE tunnel port and to forward all incoming UE traffic, received in the GRE port, to the respective hostapd interfaces (named wlan0 and wlan1

[1]https://www.pcengines.ch/apu1d.htm
[2]https://www.pcengines.ch/wlle200nx.htm
[3]https://wiki.gentoo.org/wiki/Hostapd

in this implementation). In this implementation, the flow entries were statically configured at the time of deployment, but the AP OvS can be controlled by the deployed Edge SDN Controller entity or by an exterior SDN controller. In order to connect to the edge-deployed DHCP server, another GRE tunnel was created, that will be used to receive DHCP offers from the DHCP server. To cope with the possibility of other DHCP servers in the network, a DHCP relay had to be used. The DHCP relay used was the isc-dhcp-relay daemon[4]. When the relay gets a broadcast DHCP packet it redirects the packet to the DHCP server deployed on the MEC server.

3.2.1.1 Wi-Fi Attachment procedure

In order to authenticate an UE in the Wi-Fi network of the architecture, the AAA server and DHCP server are deployed in the edge cloud. In order to get the authentication information for the UE, the AAA must communicate with the HSS, located in the core cloud. Furthermore, the DHCP server's IP address range must be different from the IP address range of the DHCP server located in the core cloud. The attachment procedure is illustrated in figure 3.4 and described in detail below.

1. The Wi-Fi AP sends an Access Request message to the AAA server with the user's International mobile subscriber identity (IMSI) upon receiving an attachment request from the UE;
2. The AAA server, in order to retrieve the Universal Mobile Telecommunications System (UMTS) authentication vector to authenticate the UE, sends a Multimedia-Auth Request DIAMETER message to the HSS. The HSS calculates the authentication vector and answers to the AAA server through a Multimedia-Auth Answer DIAMETER message (refer to section 3.2.1.2 for a description of the EAP-AKA based authentication);
3. The AAA server sends an Access Challenge message to the Wi-Fi AP with the Authentication Token (AUTN) and Random Number (RAND) parameters and saves the remaining parameters of the authentication vector;

[4]https://packages.ubuntu.com/source/xenial/isc-dhcp

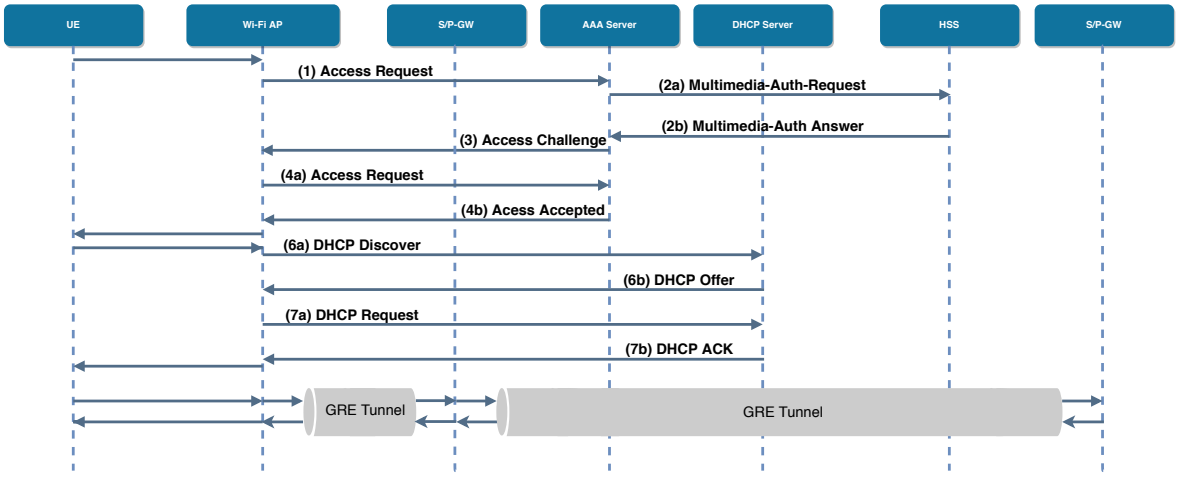


Figure 3.4: Wi-Fi attachment procedure

4. The UE calculates the Response (RES) parameter and responds to the AAA server through an Access Request message that contains the calculated RES;
5. The AAA server checks if the received RES is equal to the expected RES. If this condition is true, the AAA server sends an Access-Accept to the Wi-Fi AP;
6. Knowing that the user is authenticated, the UE starts the process of obtaining an IP address. In order to do so, it sends a DHCP Discover message to the DHCP server which in turn answers with an IP address offer for the UE through a DHCP Offer message;
7. The UE sends a DHCP Request message containing the offered IP address to the DHCP server which then answers with a DHCP ACK.

The GRE tunnels between the Wi-Fi AP and the edge S/P-GW and the edge S/P-GW and the core S/P-GW are pre-established and do not contribute for the attachment time in this architecture.

3.2.1.2 EAP-AKA Authentication

The EAP-AKA authentication is a key exchange authentication procedure. The key exchange is performed between the UE and the AAA server. In this type of authentication, a user is identified by the Universal Subscriber Identity Module (USIM)'s IMSI. The AAA requests the keys to exchange with the UE to the HSS which returns them in the form of an authentication vector. This authentication vector contains the following parameters:

1. AUTN;
2. Expected Response (XRES);
3. A RAND generated by the HSS;
4. Integrity Key (IK);
5. Cipher Key (CK).

Although all these parameters are sent by the HSS to the AAA server, only the AUTN and RAND are sent to the UE which answers with the RES parameter. This RES parameter is compared at the AAA server with the XRES, being that the authentication is successful if the two parameters match.

3.2.2 S/P-GW or UPF

The edge S/P-GW provides a way to dynamically steer data plane traffic inside the edge to the various MEC applications. The edge S/P-GW only deploys the data plane of the S/P-GW and the control is made using the edge SDN controller. The S/P-GW is realized using an OvS switch connected to the Wi-Fi AP and to the core network's S/P-GW, through a GRE port, and connected to the edge-internal network, which allows the switch to reach the MEC applications. The edge S/P-GW can also be connected to the eNB, allowing mobile devices connected through LTE to reach applications deployed on MEC servers (this procedure is later explained in section 3.2.6).

The OvS 2.9 was installed in the S/P-GW VM with 1 CPU and 2 GB RAM running Ubuntu 16.04.4 LTS with the 4.4.0-127-lowlatency kernel. A bridge was created and configured to receive instructions from the edge-controller. Then, the GRE tunnel ports were added,

to communicate with the AP and core S/P-GW, as well as the port for the edge-internal network. When the packets encapsulated in GRE tunnels reach the GRE tunnel port they are decapsulated and the inner headers will be matched against the switch's flow entries. The OvS switch was configured to use the Openflow 1.3 protocol in the southbound interface.

The default behavior of this switch is to forward every IP packet received in the Wi-Fi AP GRE port to the S/P-GW GRE port and vice versa. Furthermore, the switch also forwards all IP packets received in the edge-internal network to the Wi-Fi AP GRE tunnel port. The forwarding rules for this default behavior are sent by the edge SDN controller when the switch connects to the controller.

3.2.3 Edge SDN Controller

The edge SDN controller was implemented as an SDN controller application using RYU[5]. The VM used for the SDN controller had 1 CPU and 2 GB RAM and ran the Ubuntu 16.04.4 LTS with the 4.4.0-127-generic kernel. For this controller a specific controller application was developed. This application provides Northbound APIs to applications (cache, video streaming, mobile code offloading, etc), interprets the received information and converts it into Openflow 1.3 flow modification messages. The information that the controller application receives in the Northbound API is the following:

- Destination IP Address;
- IP Protocol (Only User Datagram Protocol (UDP) and TCP are supported for now);
- Destination UDP or TCP port.

The Uniform Resource Identifier (URI) of the REST messages where these parameters are sent to the controller application is http://<sdn controller ip>:<sdn controller port>/edge/app/add. When a new MEC application is deployed in the edge cloud, the application initialization script contains a method to send a REST message with the application's information to the SDN controller, allowing it to install the necessary flows in the S/P-GW switch in order to forward the necessary traffic to the MEC application that was just deployed. The behavior of the developed controller application is now described.

When the S/P-GW switch connects to the controller, it saves its datapath id and installs the default flows in it using flow modification messages. These default flows, as already stated, allow the user's data plane traffic to traverse the edge S/P-GW as if it was not there. When the controller receives a REST message, the data treatment procedure is presented in figure 3.5. In the figure we can see that the controller verifies the contents of the received REST message and builds the OpenFlow match accordingly. Some situations, such as the absence of an IP address, are not allowed and, in that case, the controller returns an error message to the application that sent the REST message in the first place. With this implementation, new MEC applications can be dynamically deployed since the edge SDN controller has the ability to dynamically reconfigure the flow entries in the S/P-GW virtual switch.

[5]https://osrg.github.io/ryu/

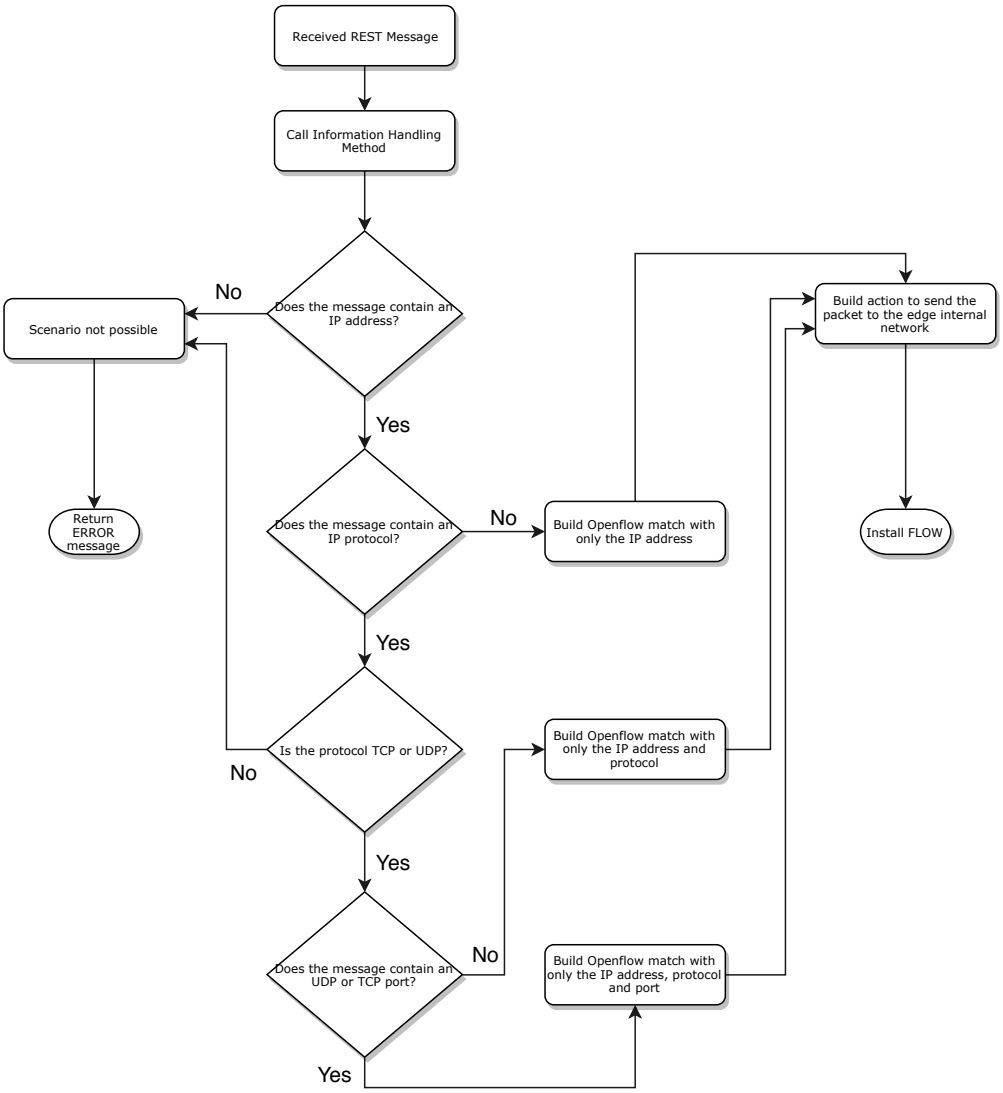


Figure 3.5: Edge Controller flowchart

3.2.4 AAA Server

For the deployment of the AAA server, the freeRADIUS[6] server was used. It was installed in a VM with 1 CPU and 2 GB RAM running Ubuntu 16.04.4 LTS with a 4.4.0-127-generic kernel. The base freeRADIUS server supports EAP-AKA authentication however, the authentication vectors must be manually defined in the configuration file and, since the Sequence Number (SQN) authentication parameter changes each time the user authenticates, a modification of the source code had to be performed. A module was developed to replace the static authentication vector definition and get the authentication vector from the HSS instead, using DIAMETER messages. For the DIAMETER implementation, the freeDIAMETER[7] project was used. This VM was connected to the external network since it needs to communicate with entities outside the edge cloud (the Wi-Fi AP and the HSS).

[6]https://freeradius.org/
[7]http://www.freediameter.net/

3.2.5 DHCP Server

In order to deploy a DHCP server at the edge cloud, the isc-dhcp-server[8] was used and configured with an IP address range that differs from the one used by the DHCP server at the core network. The DHCP server was deployed in a VM with 1 CPU and 2 GB RAM running Ubuntu 16.04.4 LTS with a 4.4.0-127-generic kernel. Once again, this VM was connected to the external network since it needs to communicate with the Wi-Fi AP.

3.2.6 MEC Traffic Offloading Function

The edge S/P-GW provides a way to dynamically steer data plane traffic to the various MEC applications through a set of rules provided by the edge SDN controller. These rules are implemented according to several factors such as access network type, service or application to use, QoS, etc. A mobile device connected to the Wi-Fi AP can connect to an application running on the MEC server in two ways:

- the front-end application running on the mobile device knows the IP address of the back-end server running on the MEC and the S/P-GW routes the traffic as a traditional router;
- an API connected to the edge controller enforces a set of rules on the S/P-GW to redirect specific traffic to the edge application, instead of the normal routing.

If the mobile device is connected through LTE a set of procedures have to be made in order to coexist with current deployed architectures. This is due to the core network idle timeout which closes the session if the user has a period of inactivity (defined by the value of the “User Inactivity timer").

On expiry of User Inactivity timer, the network releases the default Evolved Packet System (EPS) bearer to save resources and hence the UE is forced to enter Idle mode. Once the user is put in idle mode, the default radio bearer is torn down, i.e., there is no Radio Resource Control (RRC) connection once the user enters idle mode. When the user comes out of idle mode (due to traffic, paging, expiry of timers, etc.), the UE has to reestablish the RRC connection before the bearers can get reactivated. To prevent this from happening, a VM had to be deployed within the MEC server to send "keep-alive" traffic towards the core network. In this thesis, Internet Control Message Protocol (ICMP) packets were used (ping tool). The packets are sent to the edge S/P-GW, which encapsulates the traffic within the General Packet Radio Service Tunneling Protocol (GTP) protocol, setting the tunnel endpoint id to the original one present in the GTP packets sent by the eNB. Finally, the packet is sent to the core S/P-GW. This procedure is illustrated on figure 3.6.

[8] https://help.ubuntu.com/community/isc-dhcp-server

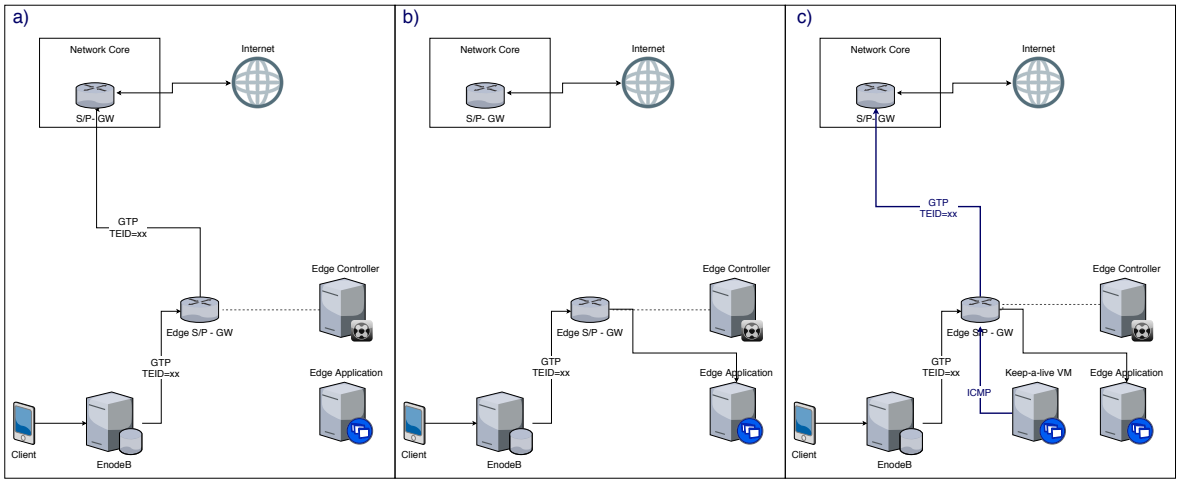


Figure 3.6: TOF procedure (a)Initial working state b)UE connects to the MEC application c)VM is created and the keep-a-live packets are sent to the core)

3.3 Use case Scenarios Implementation

In this section the implementation of use case scenarios deployed on the implemented architecture are described.

3.3.1 Edge Cache

To implement the edge cache the squid[9] tool was used. This tool allows to create a proxy with cache capabilities. For this use case a VM was deployed and connected to the external network, since it needs to be reachable from all machines within the network. A simple architecture is presented on figure 3.7.

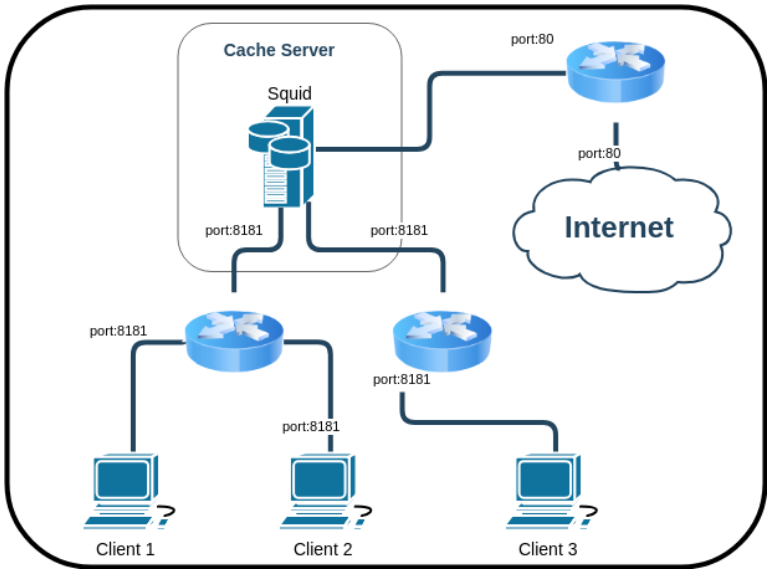


Figure 3.7: Simple web cache architecture

[9]http://www.squid-cache.org/

The proxy was deployed, defining the port and the address of the proxy, as well as the allowed IP addresses to access the cache.

The presence of cookies headers in requests does not affect whether or not an HyperText Transfer Protocol (HTTP) reply can be cached. Similarly, the presence of Set-Cookie headers in replies does not affect whether the reply can be cached. The proper way to deal with Set-Cookie reply headers, according to RFC 2109 [158] is to cache the whole object, except the Set-Cookie header lines. The installed proxy can filter specific HTTP headers but instead of filtering them on the receiving-side, it filters them on the sending-side. Thus, the proxy does cache replies with Set-Cookie headers, but it filters out the Set-Cookie header itself for cache hits.

3.3.2 Remote Code Offloading

The remote code offloading was installed using a framework from an ongoing project[10]. The source code was compiled and installed on the android devices. For code computing locally on the mobile device the compiled android application was used. For the remote code computing, a VM was deployed. Since the server side for remote code offloading from the android devices was deployed on the android platform, a virtual machine was instantiated on VirtualBox[11] running Android-X86[12], and the server application was installed.

For remote code offloading in linux machines, java applications were used: one for the client side running on a laptop, and another for the server, installed on a VM testbed. A simple diagram is show in figure 3.8.

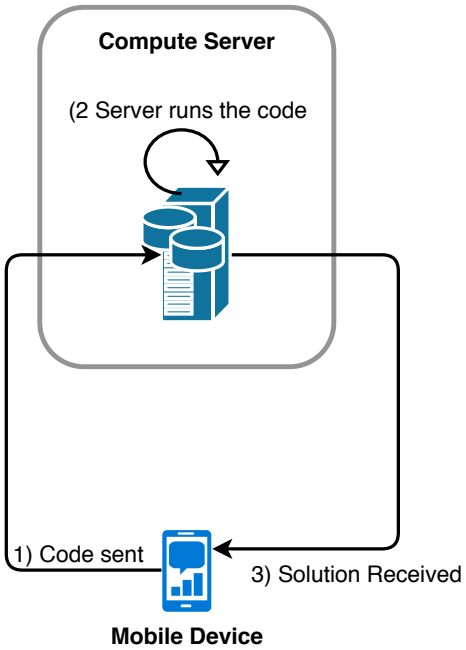


Figure 3.8: Simple Mobile Code Offloading architecture

[10] http://www.rapid-project.eu/
[11] https://www.virtualbox.org/
[12] http://www.android-x86.org/

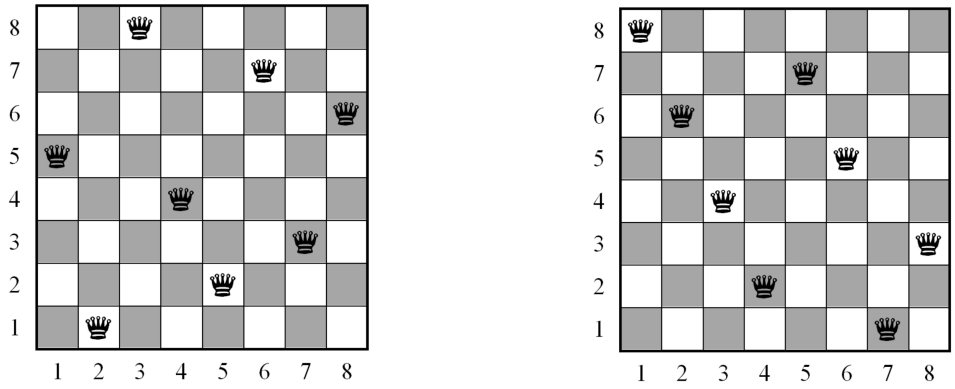


Figure 3.9: Two possible solutions to 8-Queens puzzle

Source: [159]

The remote code offloading application enables automatic remote computation of heavy tasks on Android and Linux Java applications.

The task in the implemented application is the N-Queens puzzle: a task of arranging N chess queens in the chess board so that no two queens can attack each other. The solution to this problem is obtained by a brute force algorithm. The number of queens varies from 4 to 8, varying this way the difficulty of the problem and consequently the time to solve it. Two possible solutions for a 8-Queens puzzle (8x8 board with 8 queens) are show on figure 3.9.

In the android application it is possible to choose if the user wants to run the code locally directly on the mobile device, or, offload the code, which sends the code for remote computation, and upon receiving the solution, displays it on the screen along with the total time of the procedure. In the linux application the computation is performed remotely on the VM and cumulative statistics of the time to solve the task are displayed. The expected result is that while increasing the number of queens, the gap between the local and remote execution time should increase, with the remote execution being faster than the local execution when increasing the difficulty.

3.3.3 Video Streaming

For the video streaming use case, the nginx[13] software was used. Nginx is a HTTP and reverse proxy server, a mail proxy server, and a generic TCP/UDP proxy server. Is this scenario, this software was used as a streaming video server. The protocol used was Real Time Messaging Protocol (RTMP), a protocol for streaming audio, video and data over the Internet, which works on top of TCP, usually on port 1935 by default. RTMP can be encapsulated within HTTP requests to traverse firewalls and it is frequently found utilizing clear text requests on TCP ports 80 and 443 to bypass most corporate traffic filtering. The encapsulated session may carry plain RTMP packets within. In the software configuration, a RTMP server was created to receive the video source, and an application was created for live streaming the received video. The clients that wish to received the video stream, connect to the created application (see figure 3.10).

[13] http://nginx.org/en/

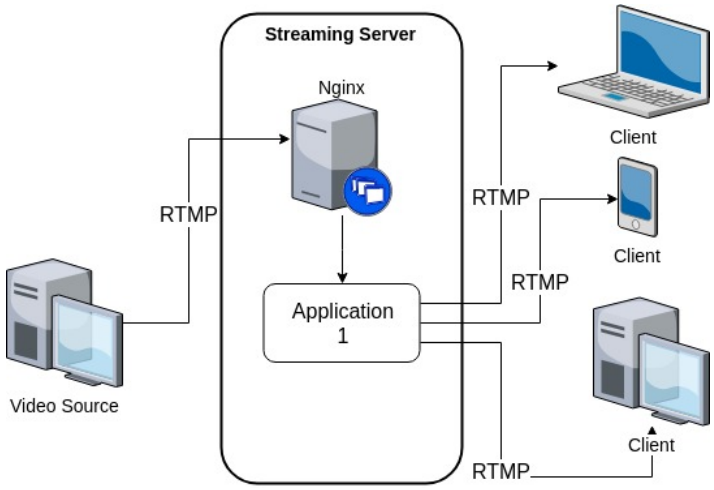


Figure 3.10: Simple re-stream architecture

Another application, that works along with the first application, was deployed to stream the same received video but with lower definition, and consequently lower bitrate. This way, the client can choose between high quality video, or if the network condition doesn't allow, lower quality, which requires lower bandwidth. For this second application an encoder to resize the video was needed, and the software chosen was FFmpeg[14]. FFmpeg is a free software project, designed for command-line-based processing of video and audio files. The software was running in the background, resizing the original received video and sending it to the nginx listening server, in the second application. An overall architecture of the scenario is shown in figure 3.11.

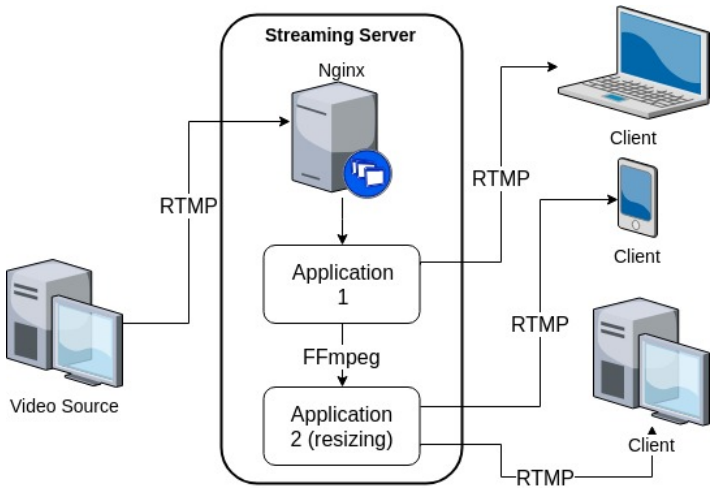


Figure 3.11: Resizing streaming architecture

Lastly, another application was deployed within nginx. This application, like the second application, uses FFmpeg to modify the source video and re-stream it. This application in addition to resizing the video, adds an overlay to it. This application was deployed to simulate an AR application, where the user sends the video to the MEC servers, the video is

[14]https://www.ffmpeg.org/

transformed and sent back to the user. Like the second application, FFmpeg resizes the original video, adds an overlay to it, and sends the re-encoded video to a listening server, deployed on the third application within nginx. Figure 3.12 shows the implemented architecture.

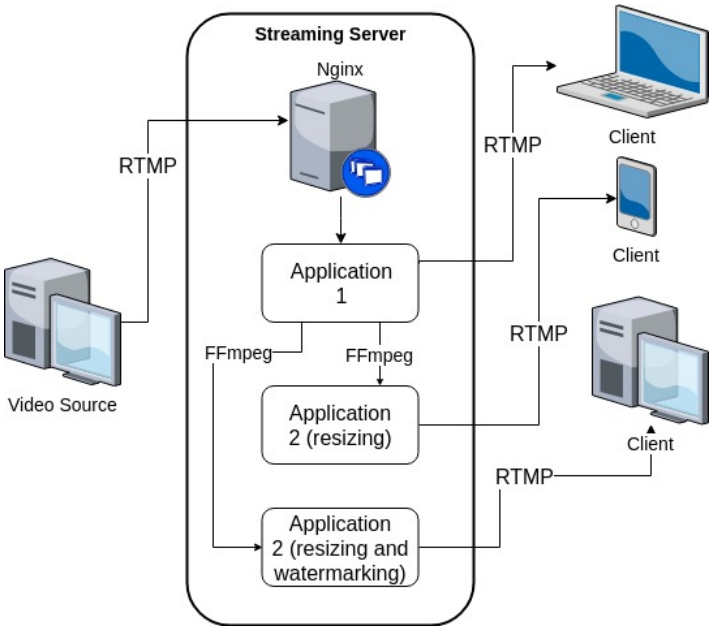


Figure 3.12: Resizing Streaming Architecture with overlay

3.3.4 Face Recognition

In this use case, the FaceSwap application[15] was deployed to demonstrate the critical role of edge servers in shortening end-to-end latency for computation offloading mobile applications. This application consists of a front-end Android client and a back-end server deployed on both edge server and centralized clouds VMs. The android client continuously streams 640x480 images captured from the mobile device camera to the back-end server. In the backed server, a hierarchy of face detection, face tracking and OpenFace-based[16] face recognition is employed. For each frame, if the face tracking is available, bounding boxes and compressed pixels of all faces are then returned. Face detection and face recognition run outside the critical path and opportunistically update trackers once their results become available. On figure 3.13, the FaceSwap architecture is displayed.

[15]https://cmusatyalab.github.io/faceswap/#faceswap-documentation
[16]https://github.com/cmusatyalab/openface

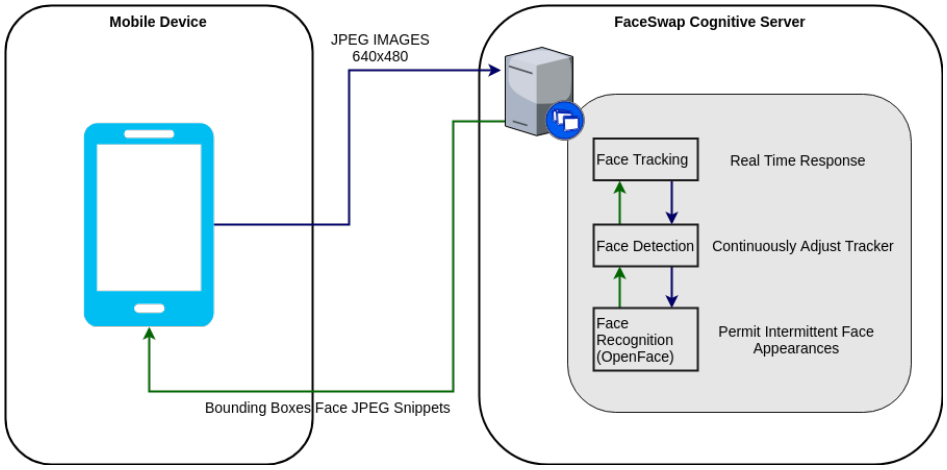


Figure 3.13: FaceSwap application Architecture

3.4 Summary

This chapter introduced a design overview of the proposed solution for MEC server deployment followed by the implementation details of each entity and network functions deployed. Lastly, the details of the implementation of several different use case scenarios to test the infrastructure were presented. In the next chapter the proposed architecture and the deployed applications are evaluated in order to assess the benefits of the MEC deployment.

CHAPTER 4

Architecture Validation

This chapter presents the tests conducted and results to corroborate the advantage of using the edge framework architecture presented in the previous chapter.

The architecture was evaluated by measuring performance indicators such as latency, throughput and attachment times. Afterwards, use cases were implemented and tests conducted to evaluate and compare the implemented framework versus centralized clouds.

Lastly, the measurements and performance results of the proposed testbed were analyzed. All tests were performed 10 times, with the results presenting their average with a confidence interval of 95 percent, unless stated otherwise.

4.1 Architecture signaling and performance indicators

This section aims to measure and analyze the performance indicators, such as latency and throughput, as well as signaling exchange in the proposed architecture.

4.1.1 Signaling

This section aims to evaluate the size of the control messages in the implemented architecture. Two different scenarios were evaluated: one where the entities, AAA server and DHCP server were deployed on the edge, and another experiment where the authentication was made internally on the AP using WPA2-PSK and the DHCP server was deployed directly on the AP. The mobile device used was a Samsung Galaxy J5 2016 running Android 7.1. The tool used to collect the messages exchanged between architecture entities was tcpdump[1].

For the first scenario the packets were captured in both the authentication server and the DHCP server entities. The results obtained are presented in table 4.1.

In the second scenario, the size of the exchanged messages was measured by capturing the packets directly at the AP. The results are displayed at the following table 4.2.

[1] https://www.tcpdump.org/

Entity	Message	Size (bytes)
AAA server	Access Request	315
	Access Challenge	176
	Access Accept	255
DHCP	DHCP Discover	348
	DHCP Offer	385
	DHCP Request	360
	DHCP ACK	385

Table 4.1: Size of messages exchanged - Authentication and DHCP deployed on MEC server

Entity	Message	Size (bytes)
EAPoL Authenticator	EAPOL-Key (message 1)	113
	EAPOL-Key (message 2)	135
	EAPOL-Key (message 3)	187
	EAPOL-Key (message 4)	113
DHCP	DHCP Discover	346
	DHCP Offer	342
	DHCP Request	358
	DHCP ACK	342

Table 4.2: Size of messages exchanged - Authentication and DHCP deployed on AP

By looking at the size of the signaling messages, it is possible to observe that the DHCP messages are the ones with highest impact. The impact of the signaling related to the DHCP in both experiments is roughly the same.

4.1.2 Attachment Time

To obtain the total attachment time, tcpdump was used to capture the packets, and by calculating the time between packets the total attachment time can be measured. To have a better estimation of the total attachment time, the WiFi was turned on and off in order to perform the 10 tests. Both scenarios defined in the previous section were evaluated, and the same mobile device was used.

For the first scenario, the packets were captured in both the authentication server and the DHCP server, and in the AP. The times were measured considering that the GRE tunnel was already assembled. In the second scenario the packets were captured only at the AP. The results obtained for both scenarios are presented in table 4.3.

In the first scenario using external entities for authentication and DHCP, the attachment time was approximately 290ms slower, this is mainly because of the physical location of the entities. On the second scenario all the entities are within the AP, and on the first scenario the packets have to travel through the network to connect to the external entities instantiated on both the MEC server (AAA server and DHCP server) and the network core (HSS). Despite the longer attachment times with the authentication and the DHCP entities on the MEC server, it is a necessary procedure since it enables the SDN controller in the MEC server to have a general view of the network, following the SDN's centralized control architecture. If these entities were deployed on the network core the attachment times would be even slower.

Scenario	Attachment Time (ms)
Authentication and DHCP on MEC server	512.31(±32.17)
Authentication and DHCP on AP	223.04(±23.83)

Table 4.3: Attachment Times Comparison

4.1.3 Latency

For measuring the latency the ping tool was used. The ping uses the ICMP echo function which is detailed in RFC 792 [160]. The source computer sends a small packet through the network to a particular IP address, and waits for a return packet. The time between sending the packet and receiving the response packet is the Round-Trip Time (RTT) delay. For measuring latency, adjacent to the previous scenarios, another two scenarios were considered: Using the same AP, a third scenario was deployed using OvS but without encapsulating/decapsulating the packets, and a forth scenario without using OvS neither in the AP or edge S/P-GW. This was to evaluate the effects of using GRE tunneling and SDN, in latency times. For measuring latency in all scenarios a mobile device, Samsung Galaxy J5 2016 running Android 7.1, connected through Wi-Fi was used for sending the ICMP packets. To receive the packets, a virtual machine was deployed within the MEC servers. The packets were captured using tcpdump. The results are displayed on table 4.4.

Scenario	Latency (ms)
AAA and DHCP on MEC server (with OvS and GRE)	13.34(±2.86)
AAA and DHCP on AP (with OvS and GRE)	12.61(±3.02)
AAA and DHCP on AP (with OvS and without GRE)	13.66(±2.89)
AAA and DHCP on AP (without OvS and GRE)	7.67(±1.49)

Table 4.4: Latency comparison

With the values displayed on table 4.4 and figure 4.1, it is visible that using external authentication and DHCP server does not have significant impact on latency times. The

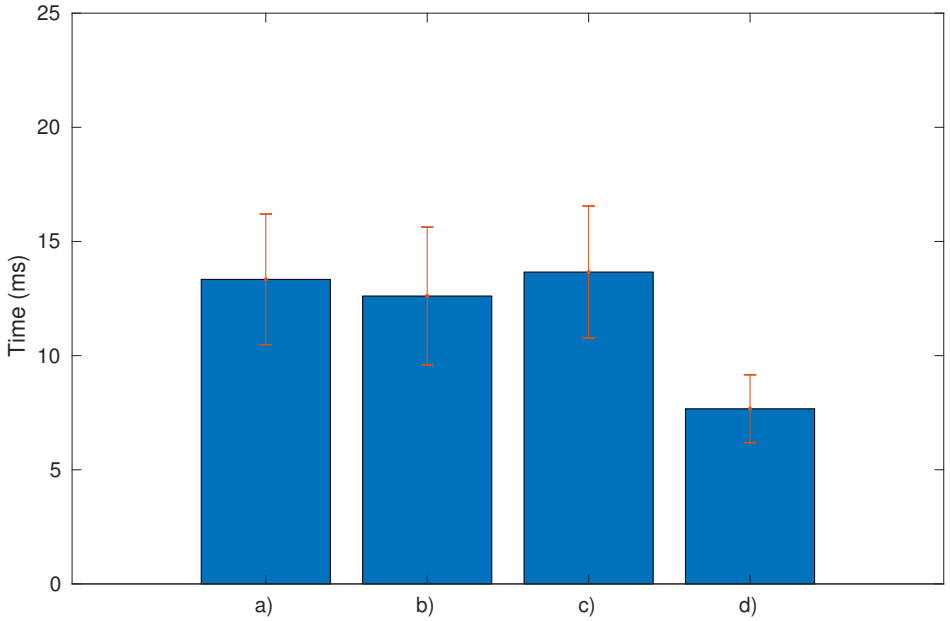


Figure 4.1: Delay Times Comparison - a)Authentication and DHCP on MEC server (with OvS and GRE) b)Authentication and DHCP on AP (with OvS and GRE) c)Authentication and DHCP on AP (with OvS and without GRE) d)Authentication and DHCP on AP (without OvS and GRE)

location of the AAA server and DHCP server does not influence the latency values, since they are only used upon new connections. Using GRE tunneling between the entities does not have substantial impact either on latency, and since it is an improvement on security the platform will continue to use GRE tunnels. On the other hand, the latency times improve when not using SDN entities. This is because two latency points were added using OvS both on the AP and the S/P-GW, since the switches used are virtual switches (OvS) deployed on VMs. Despite MEC servers aiming for latency critical applications, SDN technologies are crucial to a functional MEC deployment. In production environments using hardware-based SDN switches, the observed latency times will certainly improve.

4.1.4 Throughput

To measure the throughput on the implemented architecture the iperf3 tool was used. The mobile device (Samsung Galaxy J5 2016 running Android 7.1) with the Android's Magic Iperf[2] application installed, was used for sending UDP packets to a VM deployed on the MEC server, with iperf3 installed. Several test runs were made, increasing the bitrate of the UDP packets until saturation was reached. After that, using the bitrate saturation values, ten tests were evaluated, for uplink and downlink and values were recorded. The same four scenarios used on the previous section were used to evaluate the throughput. Table 4.5 displays the results obtained in this section.

	Throughput (Mbps)	
	Downlink	Uplink
AAA and DHCP on MEC server (with OvS and GRE)	23.31(±0.73)	25.26(±1.12)
AAA and DHCP on AP (with OvS and GRE)	23.37(±0.68)	23.76(±.74)
AAA and DHCP on AP (with OvS and without GRE)	24.13(±1.04)	24.13(±1.02)
AAA and DHCP on AP (without OvS and GRE)	23.96(±.94)	23.67(±.84)

Table 4.5: Throughput Comparison

The throughput values obtained in the performed tests (figure 4.2) indicates that there is no considerable difference in throughput values on different evaluated scenarios. Similar to latency times, the location of the AAA server and DHCP server does not affect the throughput values, since these entities are only used upon new connections. The absense of OvS and tunneling does not influence the throughput values mainly because the tests were conducted using wireless 802.11g protocol, with a theoretical throughput of 54 Mbps, and at this throughput ceiling the effects of using OvS switches and encapsulate/decapsulate packets are not noticeable, since the biggest bottleneck is on the radio.

[2]https://play.google.com/store/apps/details?id=com.nextdoordeveloper.miperf.miperf

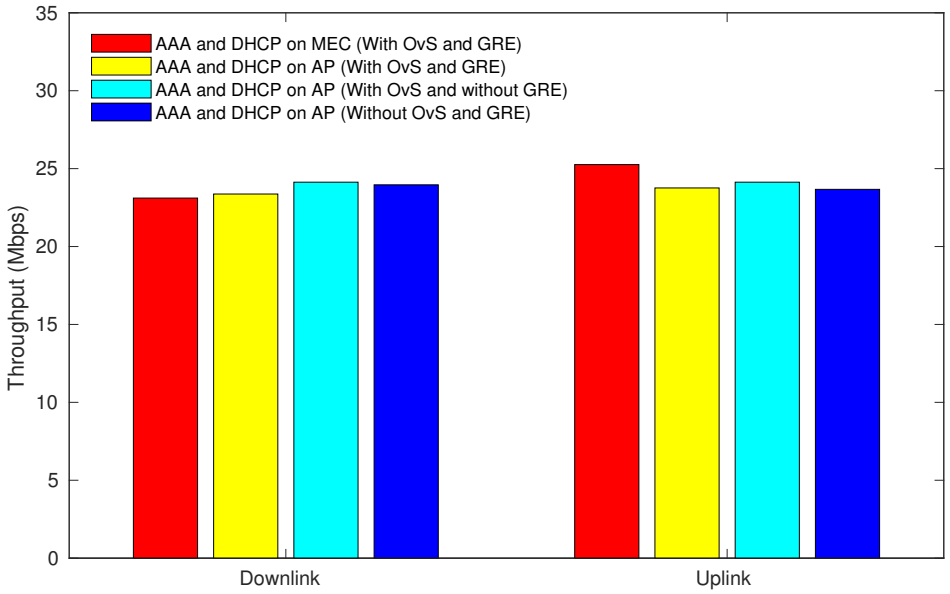


Figure 4.2: Throughput Comparison

4.2 MEC Scenarios

For experimenting the implemented architecture, different use cases/applications were deployed to validate the infrastructure and to corroborate the advantages of employing MEC servers when compared with centralized cloud infrastructures. In the next subsections, the implementation details and results of deployed applications are presented.

4.2.1 Remote Code Offloading

For evaluating the remote code offloading use case scenario, a computation task (with different levels of complexity) was performed, both locally, on android devices or remotely, on VMs physically deployed on different physical locations. The specifications of the mobile devices and VM are presented in section 4.2.1.1. Previously, the performance indicators of the UE and the VMs were measured, and are presented in section 4.2.1.2.

To perform the computation task on the UEs, an android application was compiled and installed. The android application, after request, runs the code and displays the total time of execution. The results of these compute times are shown on section 4.2.1.3. The same android application can offload the code to a remote android machine, and display the total time spent on: sending the code, executing the code remotely and receiving the solution. To measure these times a virtual machine was created, to be used as a server to compute the offloaded code, using VirtualBox[3] runing AndroidX86 6.0. The VirtualBox was deployed on a machine equipped with a Intel Core i7-3632QM Processor, 8Gb RAM running Ubuntu 16.04.4 LTS.

To perform tests on centralized clouds, a Java[4] client application was deployed on a computer (Intel Core i7-3632QM Processor, 8Gb RAM running Ubuntu 16.04.4 LTS), and a server application was deployed on several VMs (either on centralized clouds and on MEC

[3]https://www.virtualbox.org/
[4]https://www.java.com/en/

servers). Several tests were then conducted using the same client but with different servers. Results are summarized on section 4.2.1.3. In these tests, both client applications were connected to the network using the AP wireless interface with internal authentication and DHCP server. This interface was used because the mobile devices where the application was installed did not support the EAP-AKA authentication method.

4.2.1.1 Devices and Virtual Machines

In this section the general specifications of both mobile devices and VMs used on evaluating this scenario are summarized in table 4.6 and table 4.7.

Brand and Model	CPU	CPU frequency	RAM	OS
Samsung J5	Qualcomm Snapdragon Quadcore	1.2GHz	1.5Gb	Android 5.1.1
Oukitel U15Pro	MediaTek MT6753 Octacore	1.3GHz	3Gb	Android 6.0
Asus Transformer	Intel AtomZ3745 QuadCore	1.86GHz	1Gb	Android 4.4.2

Table 4.6: Mobile Devices Specifications

In table 4.7 the VMs characteristics are summarized. The VMs were deployed using popular centralized cloud services, which grant the possibility to chose the physical location of the datacenter where the VMs are instantiated, between many location options. After some deployment tests, the final locations were chosen taking into consideration throughput and latency performance (figure 4.8). The VMs were instantiated using similar resources, either in all cloud service providers or the MEC server.

Virtual Machine	CPU	RAM	Disk	OS
Edge Android	1vCPU	1Gb	4Gb SSD	AndroidX86 6.0
Edge Linux	1vCPU	1Gb	4Gb SSD	Ubuntu 17.10
Microsoft Azure	1vCPU	1Gb	4Gb SSD	Ubuntu 17.10
Google Cloud Platform	1vCPU	1.7Gb	10 Gb SSD	Ubuntu 17.10
Amazon AWS	1vCPU	1Gb	Elastic Block Store (EBS)	Ubuntu 16.04

Table 4.7: Vitual Machines Specifications

Virtual Machine	Continent	Region	Country	City
Microsoft Azure	Europe	West Europe	Netherlands	Amsterdam
Google Cloud Platform	Europe	europe-west1	Belgium	St. Ghislain
Amazon AWS	Europe	eu-central-1	Germany	Frankfurt

Table 4.8: Vitual Machines Locations

4.2.1.2 Virtual Machines Latency and Throughput

To test the latency, the ping tool was used. In the VMs instanciated on service cloud providers, firewall rules have to be modified to open ports and to allow ICMP packets. On the VM deployed on Microsoft Azure it was not possible to measure latency with the ping tool, because the ICMP protocol is not allowed through the Azure load balancer (inbound or outbound),

and it is not possible to change that particular rule. So in order to measure latency on Microsoft Azure VM, the tool PSPing was used.

The tool used to measure throughput was iperf3. On cloud service providers VMs, iperf3 was installed and firewall rules changed, allowing UDP packets on a designated port (ingress and egress). Afterwards, the measurements were made, increasing UDP bitrate until saturation was reached. After that, using the bitrate saturation values, ten tests were performed, for uplink and downlink. The results are displayed on the following table 4.9.

When measuring latency for the Microsoft Azure instantiated VM, a laptop (Intel Core i7-3632QM Processor, 8Gb RAM) running Windows 10 was used, connected through Wi-Fi to the network using the AP wireless interface with internal authentication and DHCP server. In all the other tests the same machine was used, but running Ubuntu 16.04.4 LTS, connected to the network in the same way.

	Latency (ms)	Throughput (Mbps)	
Virtual Machine		Downlink	Uplink
Edge Android	18.4(±0.18)	19.68(±1.86)	19.52(±1.97)
Edge Linux	9.70(±0.32)	24.47(±0.91)	23.55(±0.96)
Microsoft Azure	61.3(±2.31)	20.13(±1.12)	19.19(±1.58)
Google Cloud Platform	55.1(±1.89)	18.67(±1.14)	20.97(±0.97)
Amazon AWS	58.4(±0.87)	19.71(±0.86)	20.77(±1.22)

Table 4.9: Virtual Machines Latency and Throughput

4.2.1.3 Compute Times

The results of the compute time when running the code directly on the mobile devices are displayed on the following table 4.10. All tests were made 10 times, with the exception of the more demanding task "8 queens", that was only made 5 times because of the exaggerated time it took to execute (around four and half days). The Asus Transformer Pad couldn't solve the solution for "8 queens" because after around 15 minutes the application froze.

Device	4 Queens(ms)	5 Queens(ms)	6 Queens(ms)	7 Queens(ms)	8 Queens(ms)
Samsung	11.82(±2.11)	117.476(±5.36)	1932.61(±49.27)	37131.67(±411.45)	389428.23(±9728.35)
Oukitel	12.12(±4.45)	86.22(±5.97)	1182.42(±11.11)	27297.70(±372.85)	242397.70(±6541.75)
Asus	206.36(±3.86)	250.37(±34.91)	531.90(±7.83)	6228.77(±71.37)	–

Table 4.10: Mobile Devices Computing Times (ms)

The results of compute time when running the code remotely on virtual machines are displayed on the following table 4.11.

The following graphics allows to better visualize the time discrepancy between compute directly on the UEs, or compute remotely (send, compute on the VM, and collect the results), when performing the same task.

When computing 4 queens (figure 4.3), the task is effortless to compute, so it is faster to compute in mobile devices. The Asus Transformer took a disproportionate amount of time,

VM	4 Queens(ms)	5 Queens(ms)	6 Queens(ms)	7 Queens(ms)	8 Queens(ms)
Edge Android	62.23(±14.02)	66.24(±11.80)	102.53(±15.65)	931.62(±40.20)	20068.56(±460.00)
Edge Linux	28.79(±1.03)	30.91(±1.22)	56.01(±1.08)	168.81(±1.51)	2161.79(±13.62)
Amazon AWS	76.45(±1.84)	76.81(±1.72)	98.01(±1.65)	203.37(±1.71)	2173.63(±23.23)
Microsoft Azure	81.41(±2.37)	79.02(±1.44)	105.76(±3.89)	201.07(±2.76)	2136.58(±20.78)
Google Cloud	80.49(±0.76)	80.26(±1.61)	111.05(±7.17)	205.79(±2.40)	2100.15(±21.70)

Table 4.11: Virtual Machines Computing Times (ms)

.

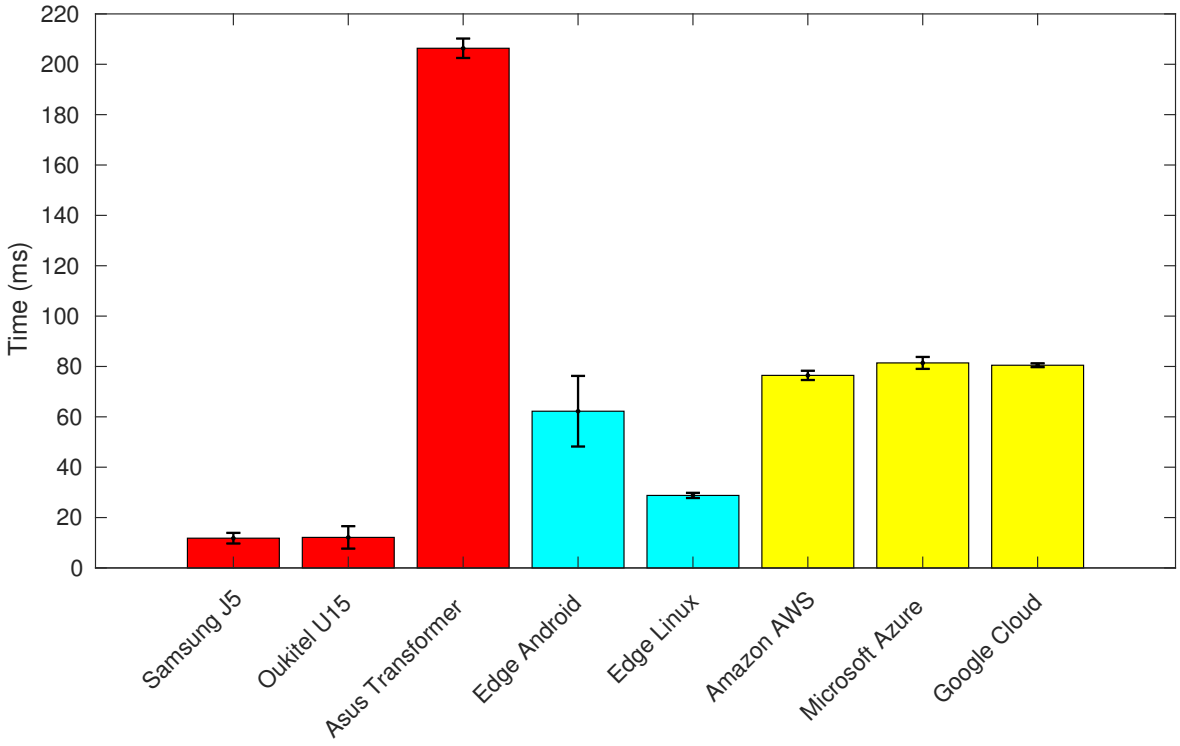


Figure 4.3: Computing Time Comparison - 4 Queens

probably due to the slow processor and low RAM, since everything was made to ensure the device's best conditions.

With 5 queens (figure 4.4), offloading the code to compute remotely is slightly faster than computing the code locally. With increasing difficulty the mobile devices began to take too long to locally compute when compared with remote compute. The VM instantiated on the MEC server is faster (roughly half the time) when compared with the centralized cloud located VMs.

When computing 6 and 7 queens the time differences between solving the solutions directly on the mobile devices and computing remotely are substantial (figures 4.5 and 4.6). The VM deployed on the MEC server is faster than all the others deployed on centralized clouds due to its physical proximity.

When computing the most difficult task, 8 queens, the edge server compute time was similar to the compute time performed by centralized cloud VMs, even slower than Microsoft Azure and Google cloud (figure 4.7). This is mainly because the latency time is insignificant when compared to the amount of time that takes to compute this task, and probably the

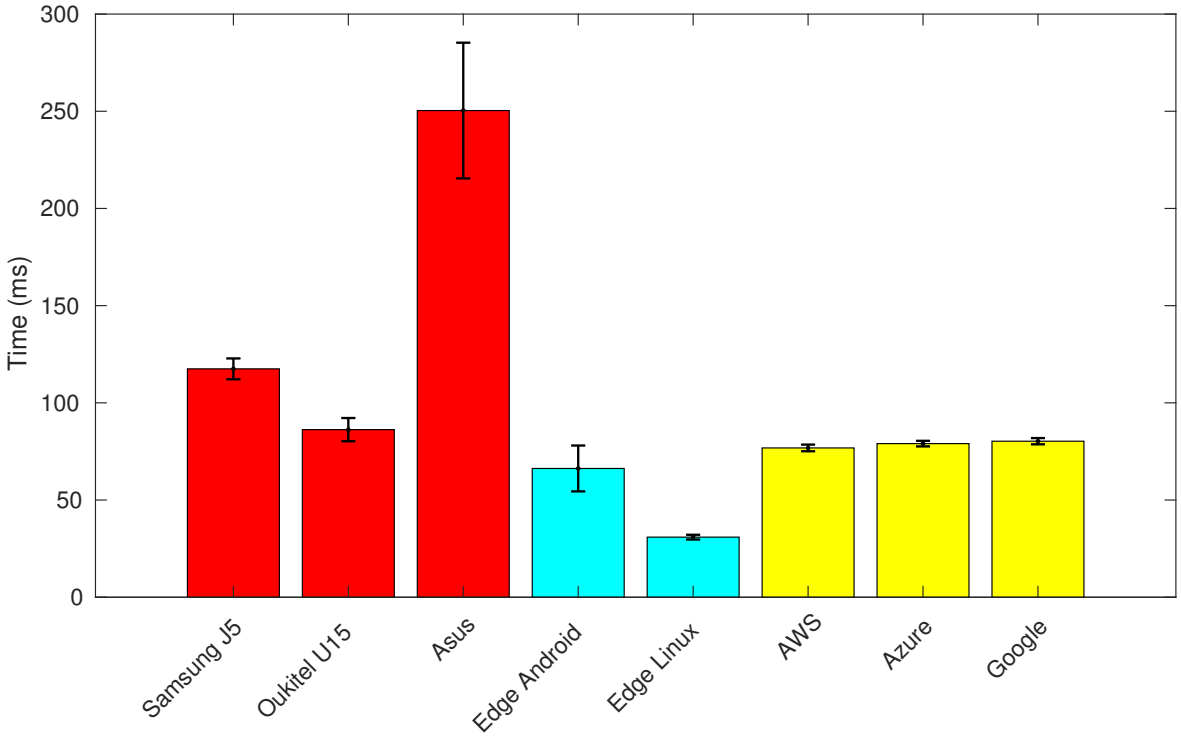


Figure 4.4: Computing Time Comparison - 5 Queens

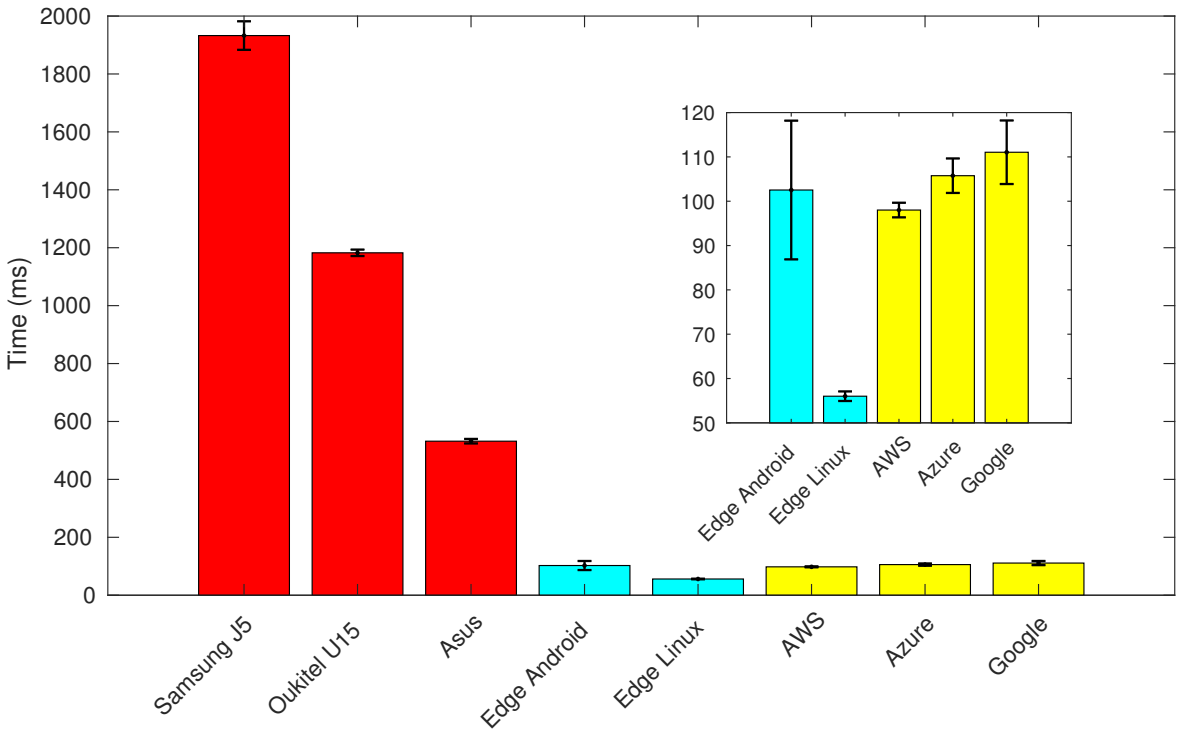


Figure 4.5: Computing Time Comparison - 6 Queens

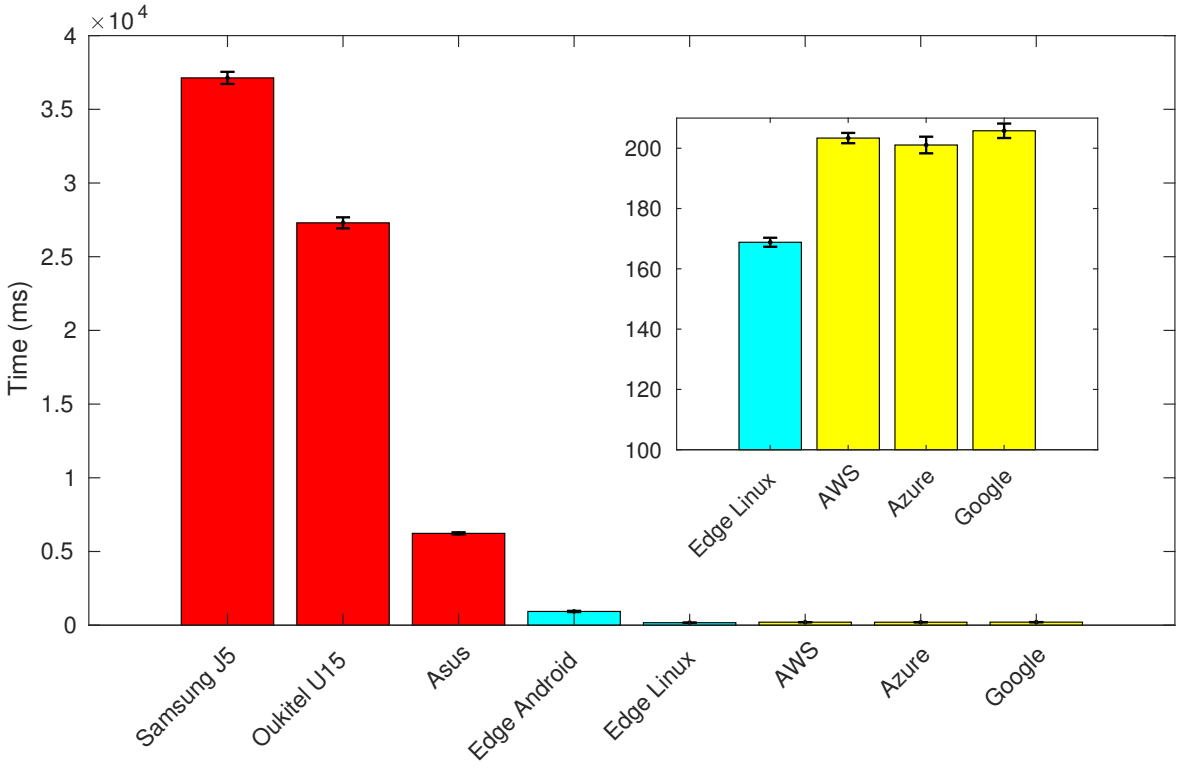


Figure 4.6: Computing Time Comparison - 7 Queens

processors used on the centralized clouds have better performance that the ones used on the MEC server. It is important to notice that with a congested network, the results for centralized cloud compute times may fluctuate substantially.

In figure 4.8 a graph is presented with the overal performance time between all the devices and VMs. It is observed that the mobile devices benefit from code offloading, especially when computing heavy tasks. By using computing offloading on the MEC the UEs save time, battery life and reduce backhaul congestion.

The traffic exchanged between the mobile device and the back-end server is presented on figure 4.9. For all the tested tasks, 4 queens to 8 queens, the amount of data sent and received is approximately the same, only varying the time spent by the back-end server solving the tasks.

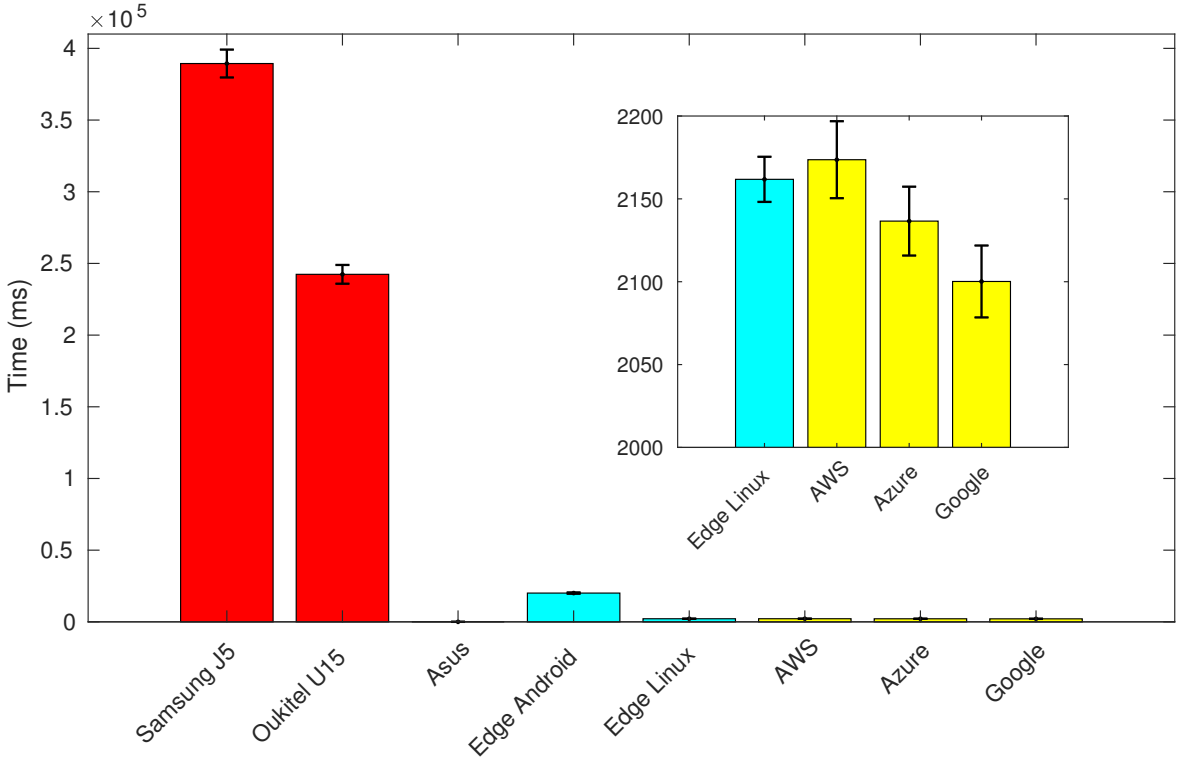


Figure 4.7: Computing Time Comparison - 8 Queens

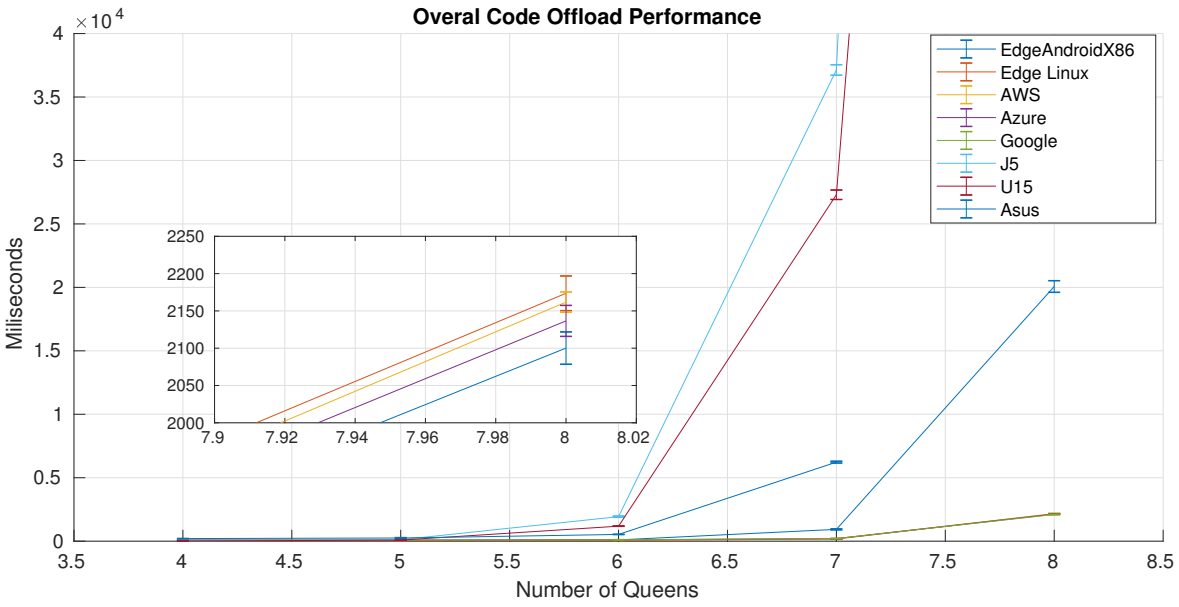


Figure 4.8: Overall Computing Time Comparison

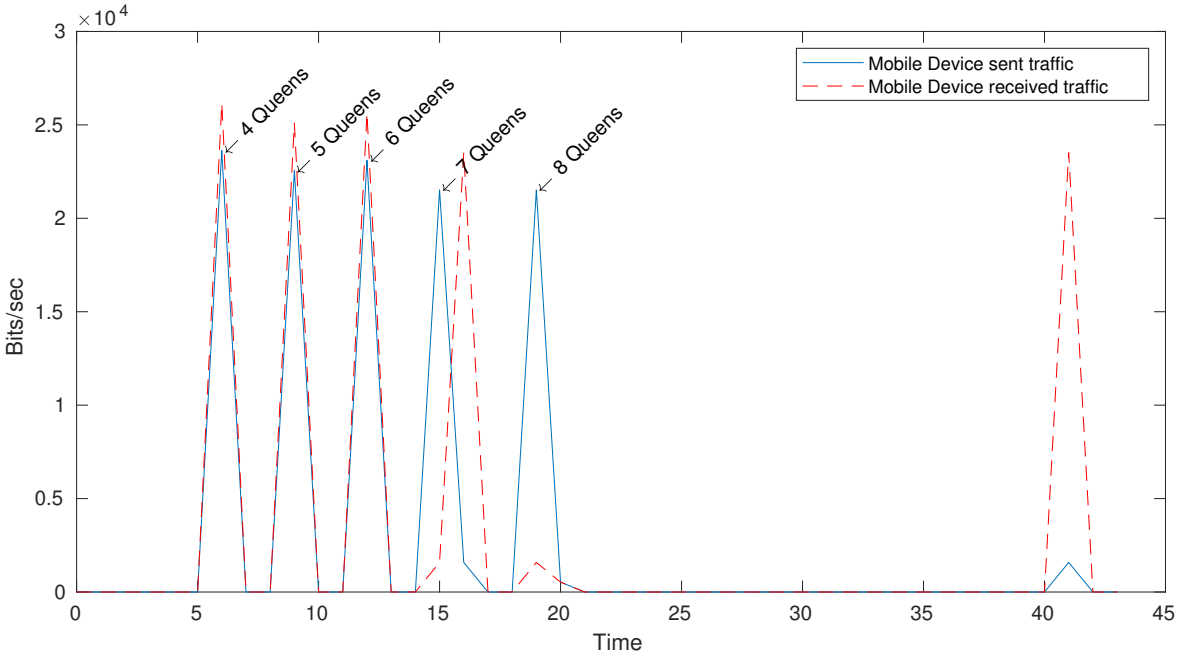


Figure 4.9: Traffic between mobile device and back-end server

4.2.2 Video Streaming

This section presents the results of video stream delay measurement, using a re-streaming server deployed on a VM. The re-streaming was deployed on a MEC VM and on centralized cloud services VMs, as in the previous section.

For this scenario evaluation, three experiments were implemented and tested: the first was a simple re-stream server, the second one a re-streaming server that resizes the video, and the final one a re-streaming server that resizes the video and adds an overlay image on top of the video. In the three different experiments, the same video stream source was used, a screen capture with ffmpeg software[5], a free software project designed for command-line-based processing of video and audio files. For the rendering settings the h264 encoder was used, with a 800x600 pixels resolution at 30 frames per second. To limit the output bitrate the encoder buffer size was set at 4000k bits per second with a maximum tolerance of 1500k bits per second. The audio codec used was Advanced Audio Coding (AAC) with a bitrate of 64kbit per second. This scenario was implemented to evaluate the edge advantage on streaming with low latency capabilities, and to simulate an AR scenario. To this extent, the streaming source and client were deployed on the same computer. The computer used was an Intel Core i7-3632QM Processor, with 8Gb RAM running Ubuntu 16.04.4 LTS. The table 4.12 displays the VMs used to evaluate this scenario, and table 4.13 displays the data center physical location where the VMs were instantiated on.

The performance indicators were measured again and summarized on table 4.14. Since this scenario requires low latency times, the laptop was connected through Ethernet directly on the AP using a Realtek RTL8101/Gigabit Ethernet card. Since the main goal in this scenario

[5]https://www.ffmpeg.org/

Virtual Machine	CPU	RAM	Disk	OS
Edge VM	1vCPU	1Gb	4Gb SSD	Ubuntu 17.10
Microsoft Azure	1vCPU	1Gb	4Gb SSD	Ubuntu 17.10
Google Cloud Platform	1vCPU	1.7Gb	10 Gb SSD	Ubuntu 17.10
Amazon AWS	1vCPU	1Gb	EBS	Ubuntu 16.04

Table 4.12: Vitual Machines Specifications

Virtual Machine	Continent	Region	Country	City
Microsoft Azure	Europe	UK South	England	London
Google Cloud Platform	Europe	europe-north1	Finland	Hamina
Amazon AWS	Europe	eu-west-2	England	London

Table 4.13: Vitual Machines Locations

is to replicate an AR application, the latency is a critical factor, so in order to have the best conditions for low latency, the OvS was not instantiated in the AP and the connection to the VM deployed on the MEC server was made directly to is external IP (without going through the S/P-GW). To measure the latency and throughput, the same metrics were used as described in section 4.2.1.2.

		Throughput (Mbps)	
Virtual Machine	Latency (ms)	Downlink	Uplink
Edge VM	1.10(±0.17)	94.96(±1.78)	94.34(±2.57)
Microsoft Azure (London)	59.86(±2.08)	34.25(±1.19)	32.81(±1.20)
Google Cloud Platform (Hamina)	52.21(±1.86)	41.84(±1.41)	40.14(±1.43)
Amazon AWS (London)	57.3(±2.21)	30.95(±1.53)	29.52(±1.08)

Table 4.14: Virtual Machines Latency and Throughput

4.2.2.1 Original Video Re-Streaming

For this scenario, the video source streams the video to the specified server, using the RTMP protocol. The server re-streams the original video through an application. The server was deployed with Nginx, an open source software for proxy and web serving, and an application was created within Nginx for the single purpose of re-streaming via RTMP to multiple clients. A client is then connected using FFplay which is a simple media player that uses the FFmpeg libraries. Figure 4.11 presents the scenario architecture.

To evaluate this scenario, the re-stream server was deployed on multiple virtual machines and the delays between the original video stream and the re-stream on the client were measured. To measure the delays, a running stopwatch was displayed on the source screen, which enabled to determine the time spent between sending the video and receiving it back.

In table 4.15 the time delays between the original source and the re-streamed video are presented as well as the average bitrate received in the client.

In figure 4.11 the time delays between the original source and the re-streamed video are

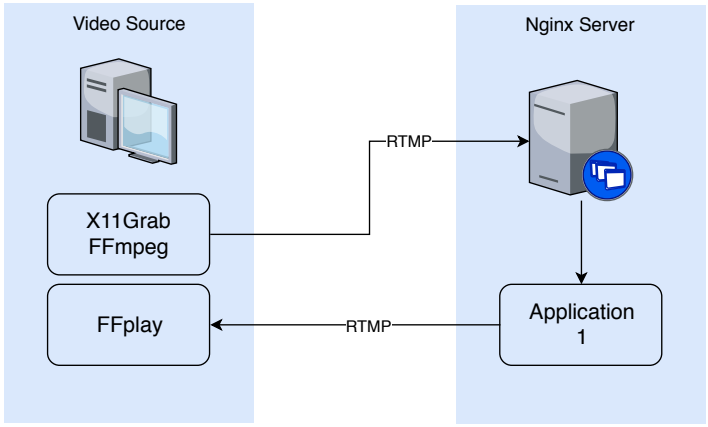


Figure 4.10: Simple re-stream architecture

Stream Server	Delay (ms)	Bitrate (kbps)
Edge Cloud	408.6(±14.11)	401.03(±0.44)
Microsoft Azure (London)	1014.70(±11.03)	397.05(±0.25)
Google Cloud Platform (Hamina)	986.20(±25.58)	397.86(±0.41)
AWS (London)	1105.10(±62.36)	397.63(±0.23)

Table 4.15: Original Video Re-Streaming

displayed. In this scenario the difference between latency times is substantial. Since the edge server has a delay of around 400ms whereas all the centralized clouds have roughly 1000ms of delay, due to the proximity of the edge server.

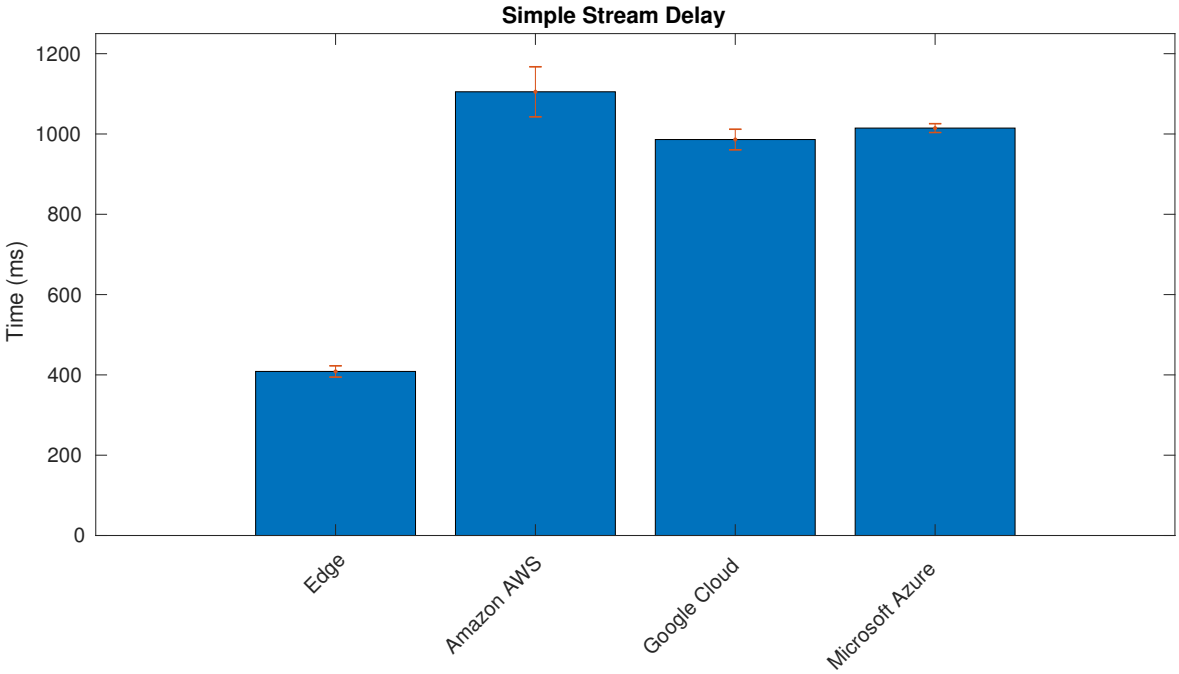


Figure 4.11: Original Video Re-Streaming Delay

4.2.2.2 Streaming and Resizing

For the second scenario, a second application was deployed that resizes the original video and then proceeds to stream the resized video, as illustrated on figure 4.12. For resizing the video FFmpeg was used, with the same codec as the original video, and with a resolution of 480x320 pixels. The delay between the original video and the resized one was measured, using the same technique as on the previous scenario. The average bitrate received in the client was also measured. The measured values from the second scenario are displayed on table 4.16.

For the second scenario, where the video needs to be encoded before re-stream, the edge server has considerable low delay when compared with centralized servers, as showed on figure 4.13. Again, the low latency is due to the proximity of the edge server.

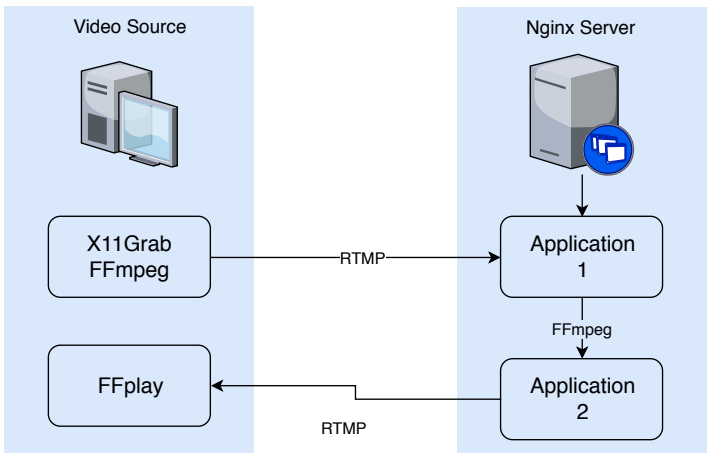


Figure 4.12: Resizing and re-streaming architecture

Stream Server	Delay (ms)	Bitrate (kbps)
Edge Cloud	1883.60(±16.79)	279.95(±0.65)
Microsoft Azure (London)	4209.30(±29.31)	236.69(±0.10)
Google Cloud Platform (Hamina)	3764.10(±36.89)	244.56(±0.18)
AWS (London)	3969.80(±16.01)	243.14(±0.18)

Table 4.16: Resizing Stream

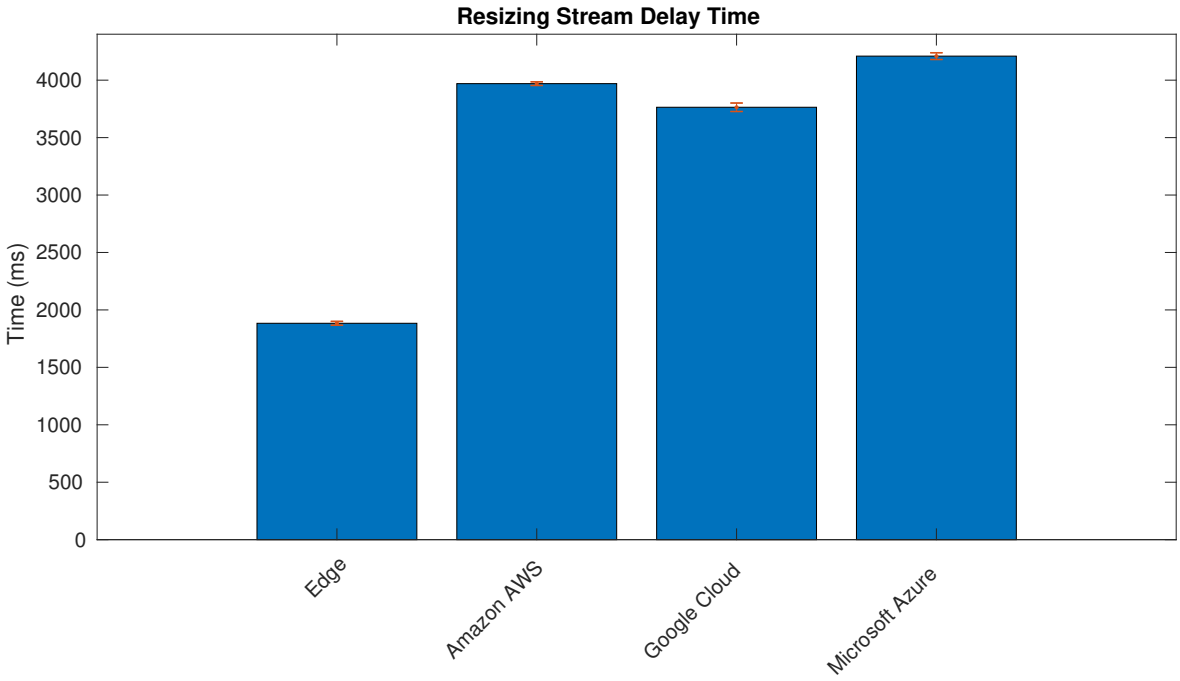


Figure 4.13: Resizing Streaming Delay

4.2.2.3 Streaming, Resizing and Overlay

For this scenario a third application was deployed which resizes the video and adds an overlay on top of the video. The overlay used was a static image, displayed as a watermark on top of the original video using the FFmpeg software. The scenario architecture is displayed on figure 4.14. The delay between the original video and the resized one was measured, using the same technique as in the first scenario and the results measured are shown on table 4.17.

Stream Server	Delay (ms)	Bitrate (kbps)
Edge Cloud	1946.80(±30.96)	295.68(±0.70)
Microsoft Azure (London)	4294.90(±38.61)	252.4(±0.22)
Google Cloud Platform (Hamina)	3826.60(±44.58)	260.17(±0.40)
AWS (London)	4029.80(±26.29)	258.8(±0.30)

Table 4.17: Resizing and Overlay Stream

Like the scenarios above, the edge server has less delay than the servers deployed on

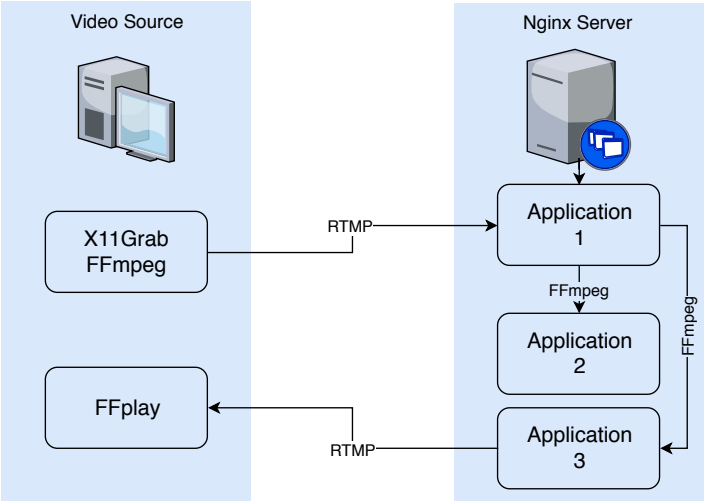


Figure 4.14: Resizing and re-streaming architecture

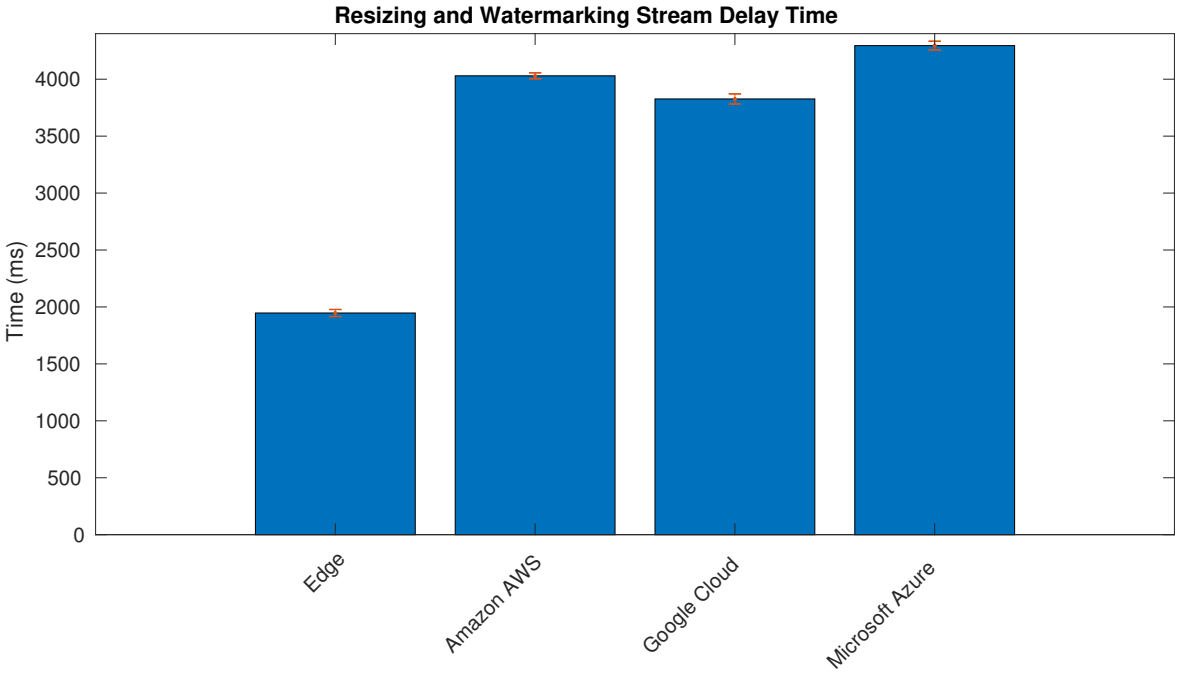


Figure 4.15: Resizing and watermarking Streaming Delay

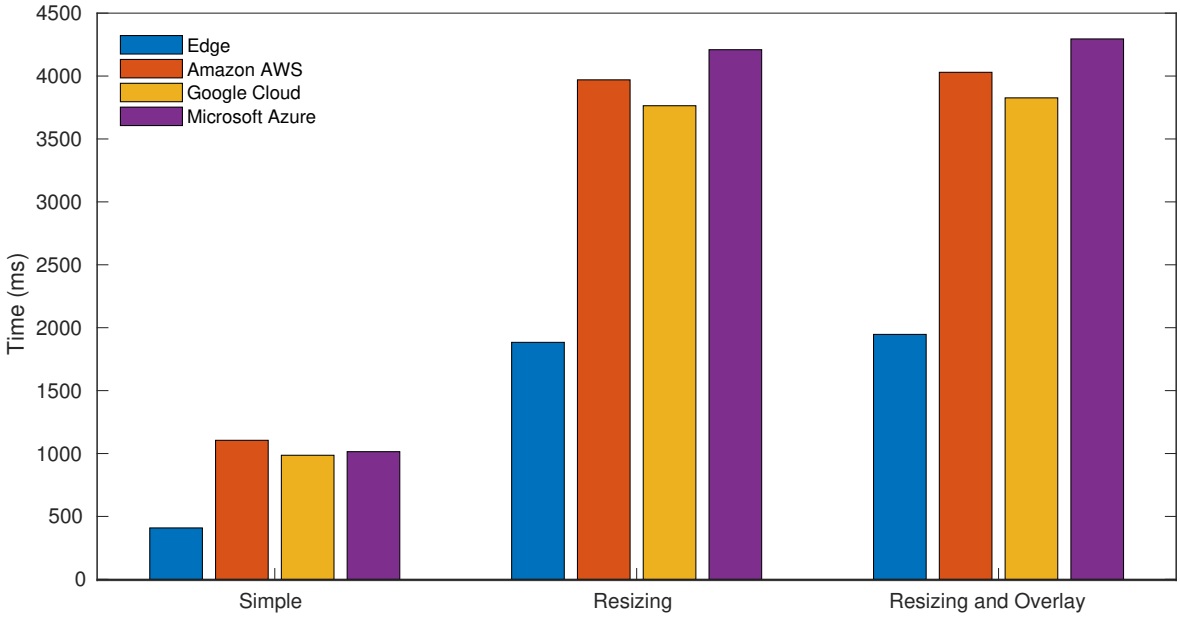


Figure 4.16: Resizing Streaming Delay

centralized clouds as displayed on figure 4.15. Again, due to the proximity of the users, the MEC servers have an important role on latency critical applications.

The figure 4.16 summarizes the results obtained on this section. It is clear to observe that when using MEC servers deployed close to the source of the video the latency times were significantly smaller (roughly half of the time). The last scenario was supposed to be a simulation of an AR application, making an overlay on top the received video, but with an average delay of four seconds, it is too much delay for an AR application. Nevertheless, it was established that the physical location of the re-streaming server has an important role on video streaming, mainly due to low latency, the trump card of the MEC deployments.

4.2.3 Caching

For this scenario a proxy server was deployed using squid, implemented on a virtual machine on the edge with 1vCPU and 1GB RAM running Ubuntu 16.04.4. To validate the cache scenario, two different experiments were performed.

In the first experiment, one laptop was used, equipped with an Intel Core i7-3632QM Processor, with 8Gb RAM running Ubuntu 16.04.4 LTS, connected to the network through an AP cable interface using a Realtek Semiconductor Co., Ltd. RTL8101/2/6E PCI Express Fast/Gigabit Ethernet card. The latency to the proxy server was 1.85($\pm$0.81)ms with a throughput of 98.78($\pm$2.66)Mbits per second of downlink and 96.12($\pm$2.66)Mbits per second of uplink, measured as in previous sections using ping tool and iperf3 respectively. A python script was created to open several webpages in a web browser, with six second delay between each new webpage. Before starting the monitoring, the proxy cache and the browser cache were cleaned. Tcpdump was set to capture packets on the proxy external interface and the script for opening the pre-defined webpages was initialized. After the script stops, the webpages on the browser were closed, the browser cache was cleaned, and the script is initialized again.

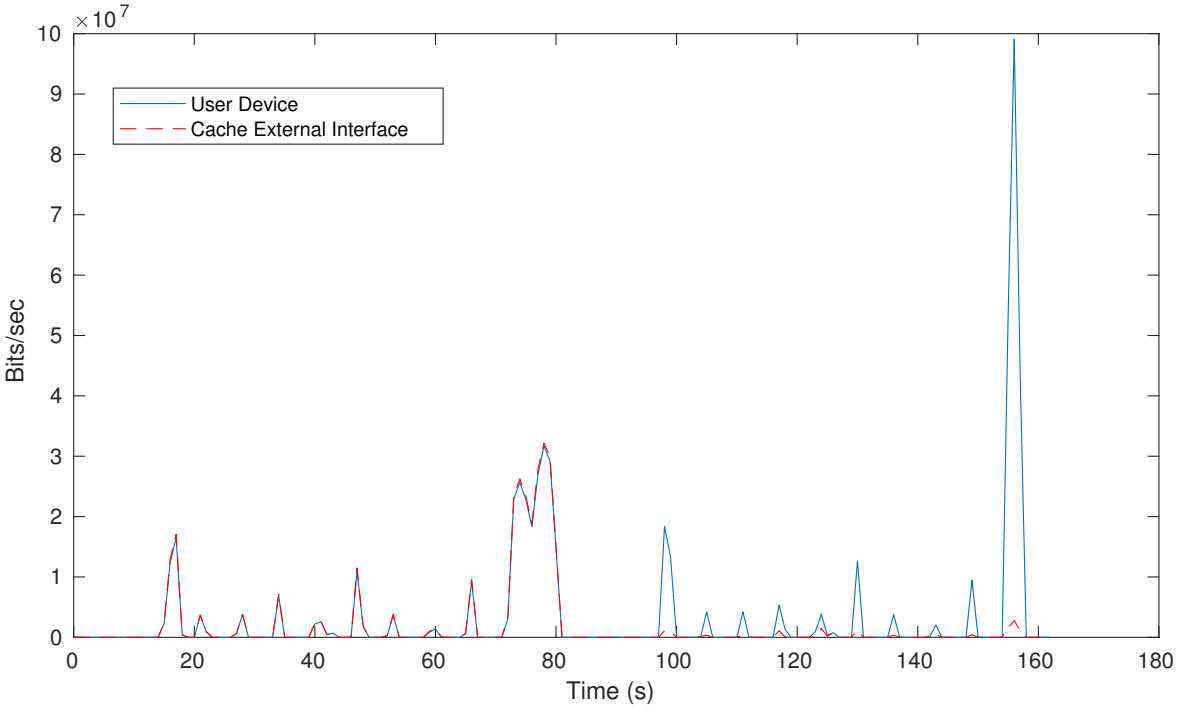


Figure 4.17: Cache Throughput

The TCP packets that are sent from external sources to the cache external interface are displayed, as well as the packets sent from the cache to the laptop. In figure 4.17 it is possible to observe the differences on the external interface, connected to the Internet, between the two experiences.

In the first run, the cache loads all the pages that are demanded by the browser, and the traffic from the cache to the laptop is the same as the traffic from the external interface to the cache. On the second run, the cache already has content of the previous webpages, so when requested to open those webpages, it checks the database to see if the webpages were open previously and check if they are in cache. If the requested content is in the cache, it does not need to download the content again from the Internet, hence the proxy sends it to the browser, resulting in improving the webpages loading speed, and saving backhaul traffic. The browser that has requested the webpages doesn't see any difference, apart from the upgraded loading speed.

In the second experiment, two laptops were used, both connected thought LAN. The first laptop used was the one specified on the first experiment, identified in the plots by "User Device 1" connected through ethernet interface using a Realtek Semiconductor Co., Ltd. RTL8101/2/6E PCI Express Fast/Gigabit Ethernet card, and the second laptop, "User Device 2", was an Intel Core I7 2640M, with 8GB RAM, running Ubuntu 18.04 LTS, connected to the same AP interface using a Intel Corporation 82579LM Gigabit ethernet card.

Before starting the data monitoring, the latency and throughput were measured, using iperf3 and ping tools: first from the User Device 1: with 2.16($\pm$0.98)ms of latency and with a throughput of 96.87($\pm$2.14)Mbits per second of downlink and 95.49($\pm$1.97)Mbits per second

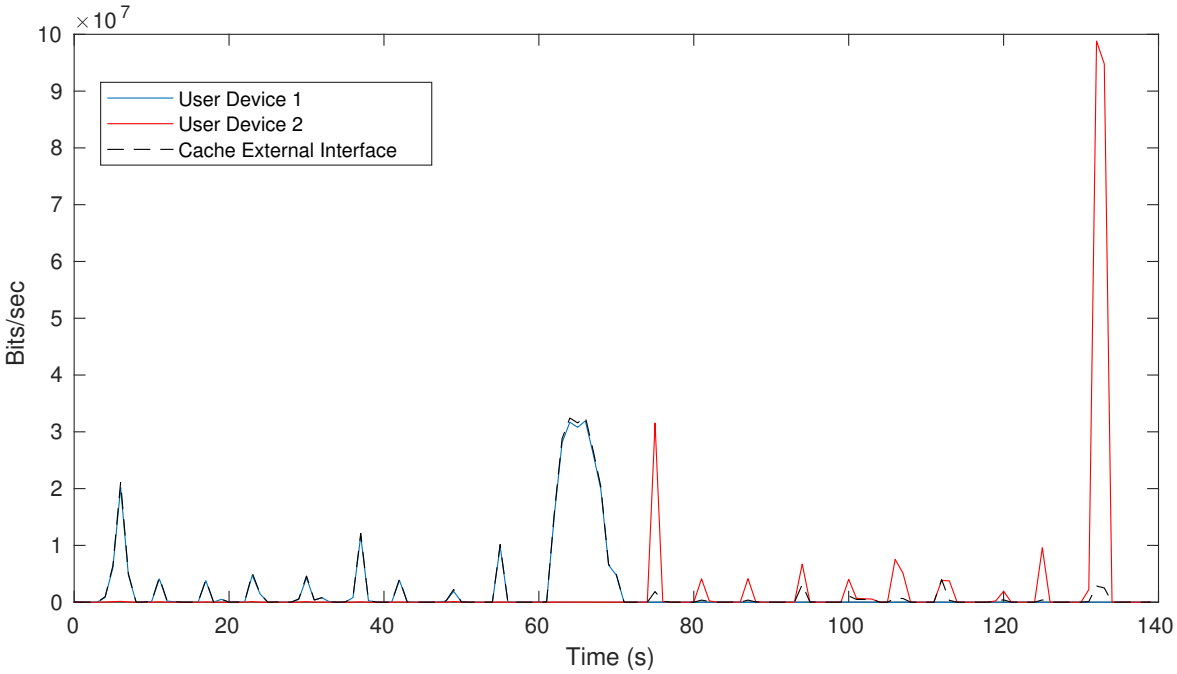


Figure 4.18: Cache Throughput using two different devices

of uplink; and then from User Device 2: with 1.79(±1.12)ms of latency and with a throughput of 96.57(±2.89)Mbits per second of downlink and 94.87(±2.88)Mbits per second of uplink.

After the performance indicators were measured, the proxy cache was cleaned, the proxy server was restarted, and both laptop browser caches were depleted. Tcpdump was then set to capture packets on the cache external interface. The script was then started on the first laptop, and after it finished loading the last webpage, the same script was then initialized on the second laptop.

This experiment was made to show that the cache is indifferent from the origin of the requests. If any source within the network requests a webpage, the content of that webpage is stored in the cache and available to everyone within the network. Figure 4.18 displays the cache storing the content as webpages were requested from one source, and then, after requested from a second source, the cache delivers the content without downloading it again.

4.2.4 Face Recognition

In this use case a front-end application was installed on a mobile device (Samsung Galaxy J5 2016 running Android 7.1), and the back-end server was installed on linux VMs deployed on centralized cloud services and on the proposed MEC server. The mobile device was connected to the AP using EAP-AKA method of authentication. After all was installed and running, the network rules of the instances had to be modified to allow traffic flow from/to the application (open port 9098 and 9101 for inbound/outbound TCP traffic), and the information of the back-end servers was introduced (server IP address) on the front-end application. On the android application some training images had to be added. This was made by collecting several pictures of the users that will be recognized, from different angles and distances, and upload those images to the application. The application was then started and latency times

and Frames per Second (FPS) values were measured. The application already provides those values on the mobile device screen, so the application was set to run from one minute, and the values were written down with 1 second of interval (60 values from each experiment).

Since several back-end servers were used to evaluate this application, care was taken to make the test environments as similar as possible: the tests were made using the same user model, light conditions, distances (from the user and the AP), and since the tests were made consecutive, the network conditions were roughly the same. The VMs specifications used to evaluate this application are summarized on table 4.18, and the data centers location where the VMs were instantiated is displayed on table 4.19.

Virtual Machine	CPU	RAM	Disk	OS
Edge VM	4vCPU	16Gb	60Gb	Ubuntu 14.04
Microsoft Azure	4vCPU	16Gb	32Gb	Ubuntu 14.04
Google Cloud VM	4vCPU	15Gb	40Gb	Ubuntu 14.04
Amazon AWS	4vCPU	16Gb	EBS	Ubuntu 14.04

Table 4.18: Virtual Machines Specifications

Virtual Machine	Continent	Region	Country	City
Microsoft Azure	Europe	west-europe	Netherlands	Amsterdam
Google Cloud Platform	Europe	europe-west4	Netherlands	Eemshaven
Amazon AWS	Europe	eu-west-1	Ireland	Dublin

Table 4.19: Vitual Machines Locations

For measuring latency a mobile device, Samsung Galaxy J5 2016 running Android 7.1, connected through Wi-Fi was used for sending the ICMP packets. To receive the packets virtual machines were deployed within the MEC servers or within centralized cloud servers. The packets were captured using tcpdump.

To measure the throughput on the implemented architecture the iperf3 tool was used. The mobile device (Samsung Galaxy J5 2016 running Android 7.1) with the Android's Magic Iperf application installed, was used for sending UDP packets to a VM deployed on the MEC server or on the centralized server, with iperf3 installed.

Several test runs were made, increasing the bitrate of the UDP packets until saturation was reached. After that, using the bitrate saturation values, ten tests were performed, for uplink and downlink and the values were recorded. The results for latency and throughput are displayed on table 4.20.

Table 4.21 displays the latency times and framerate values obtained in this section.

As noticed on previous use case scenarios, the delay on the edge server is lower using MEC server as a backend-server than using servers deployed on centralized cloud architectures (figure 4.19). With lower latency times, there are more available frames for the backend server to process, for this reason, the framerate is superior on the MEC VM, as displayed on figure 4.20.

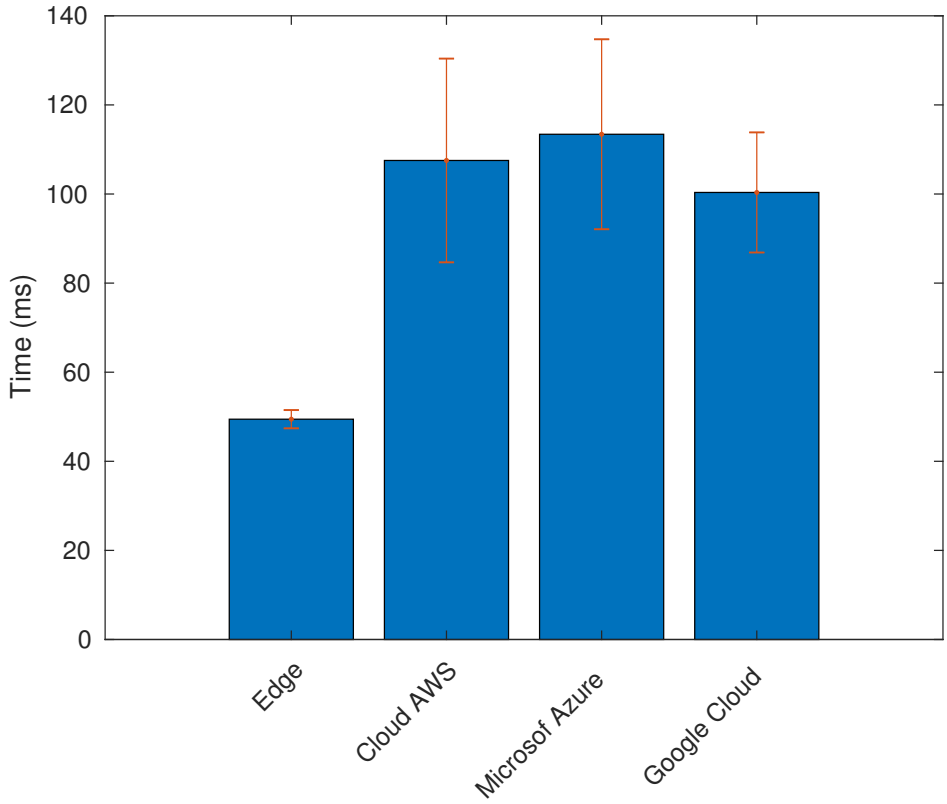


Figure 4.19: Face recognition delay comparison

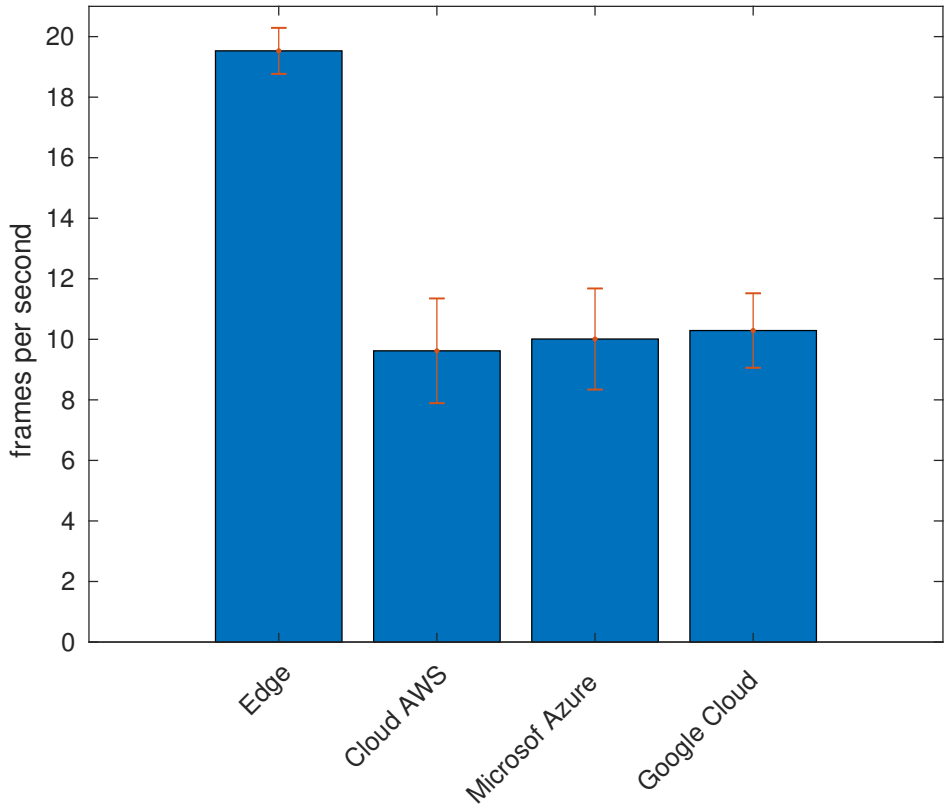


Figure 4.20: Face recognition frame rate comparison

Virtual Machine	Latency (ms)	Throughput (Mbps)	
		Downlink	Uplink
Edge VM	12.45(±1.17)	23.63(±0.96)	24.14(±0.79)
Microsoft Azure	61.25(±2.32)	20.11(±0.97)	19.29(±1.31)
Google Cloud Platform	49.85(±1.12)	19.97(±1.84)	19.08(±1.23)
Amazon AWS	56.14(±2.02)	19.33(±1.46)	18.63(±1.92)

Table 4.20: Virtual Machines Latency and Throughput

Virtual Machine	Frame Rate (FPS)	latency (ms)
Edge VM	19.53(±.76)	49.45(±2.05)
Microsoft Azure	10.01(±1.67)	113.41(±21.32)
Google Cloud Platform	10.29(±1.23)	100.35(±13.48)
Amazon AWS	9.62(±1.73)	107.53(±22.87)

Table 4.21: Face Recognition delay and frame rate comparison

For comparison proposes one more experiment was carried out, using a VM deployed on a centralized cloud but with greater resources (table 4.22). The VM was deployed on the Amazon AWS cloud service, using the same region as the previous experiment (eu-west-1, Ireland, Dublin), with 8 vCPUs and 32Gb RAM running Ubuntu 14.04. The latency of this VM was 52.25(±1.89)ms and the throughput measured was 22.23(±1.67)Mbps of downlink and 23.41(±1.58)Mbps of uplink.

Virtual Machine	Frame Rate (FPS)	latency (ms)
Edge VM 4vCPU 16Gb RAM	19.53(±.76)	49.45(±2.05)
Amazon AWS 4vCPU 16Gb RAM	9.62(±1.73)	107.53(±22.87)
Amazon AWS 8vCPU 16Gb RAM	11.86(±1.58)	98.48(±17.94)

Table 4.22: Face Recognition delay and frame rate comparison between VMs with different resources

In this experiment it is visible that, despite an improvement in both framerate and delay, it is not a considerable difference, even with double the compute power (figures 4.21 and 4.22). This can be observed because the time that the backend server took to process the images is insignificant when compared to the time that the packets take to travel from the back-end server to the front-end server. Another explanation for these similar values is considering that the application running on the back-end server does not take advantage of a multi-core setup. Still, it is evident that the MEC server is a preferable choice in latency critical applications.

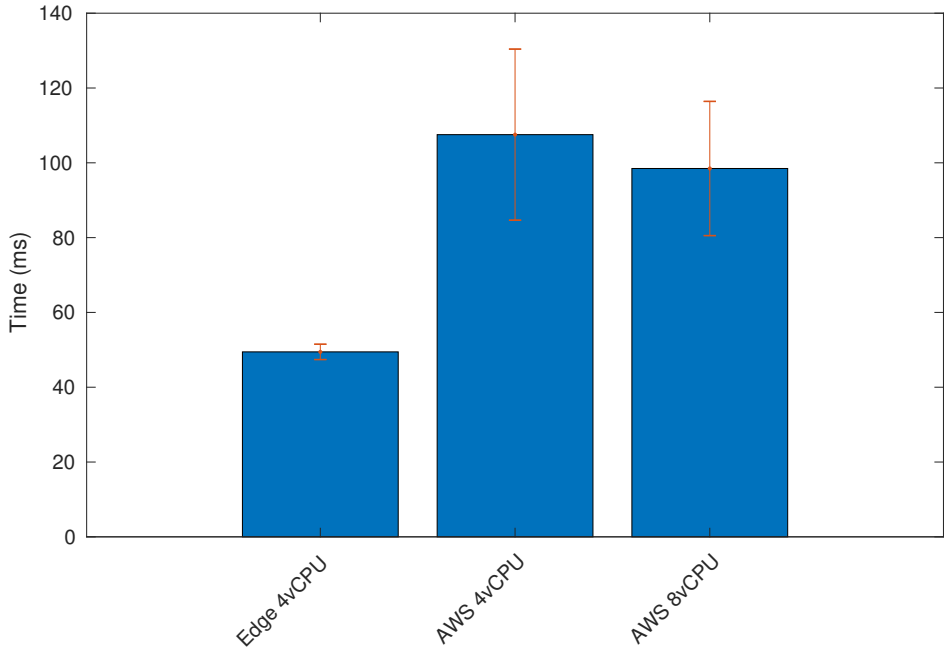


Figure 4.21: Face recognition delay comparison between VMs with different resources

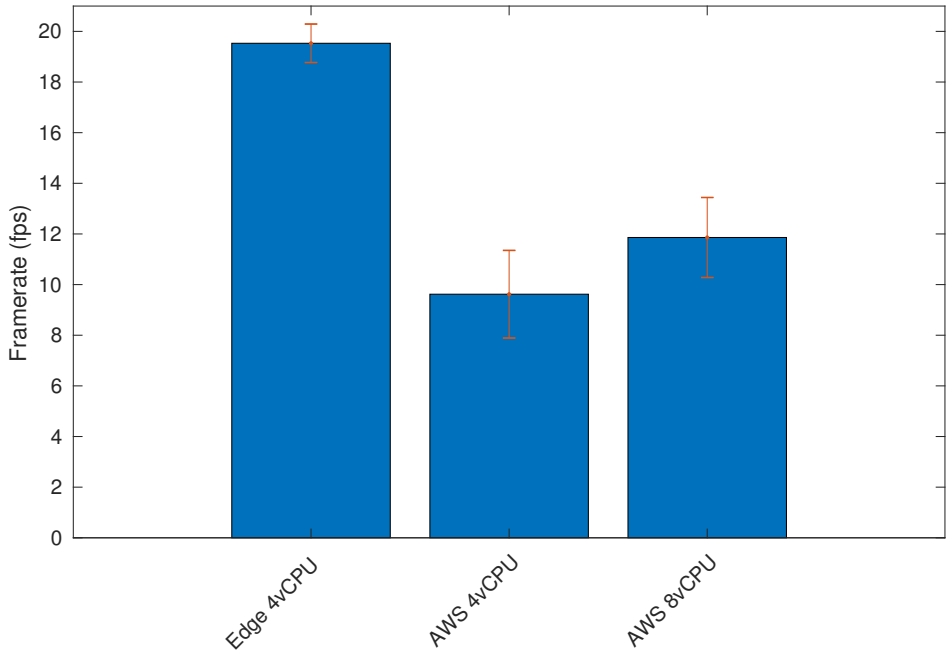


Figure 4.22: Face recognition frame rate comparison between VMs with different resources

4.3 Summary

In this chapter a performance evaluation of the proposed MEC architecture was presented. Different use case scenarios/applications were deployed both in the proposed architecture and in centralized clouds architectures, and the performance indicators were evaluated and compared. The results obtained in this chapter endorse the benefits of the MEC deployment, regarding bandwidth improvement and lower latency times.

CHAPTER 5

Final Remarks

5.1 Conclusion

MEC is an emerging technology that enables mobile operators to host content and applications in the 4G radio access, towards 5G networks requirements. The standardization effort by MEC ISG has given momentum to MEC research and early deployments for mobile operators. The implemented MEC architecture provided a virtualization environment, capable of accommodating multiple applications, services and VNFs and is able to cope with different types of access networks such as wireless, mobile and cable. When compared with traditional centralized cloud environments, the implemented architecture attains lower latency times and higher throughput.

In the mobile code offloading scenario, the computation time in the edge server presented in this thesis was significantly lower when compared with computing the code locally on the mobile device, which contributes to lower its power consumption. This allows to run compute-intensive applications that were not possible to run in useful time using the mobile devices computational power. Using the cache application/service, the web pages loaded faster when compared with a scenario where no caching was used. Furthermore, because the web pages content was already cached, the backhaul traffic towards the core network was diminished since the cache service only had to fetch the content when the first user requests it.

Despite all the unaddressed issues in this architecture deployment and evaluation, such as privacy and security, mobility management, and orchestration, the proposed environment can be seen as a base platform for future scenarios exploitation, to provide full satisfaction of all involved parties such as mobile operators, service providers, and users.

5.2 Main Contributions

This thesis execution results in a physical testbed for testing future use case MEC-enabled scenarios. The architecture implemented in this work will be used as a base platform for future working on the ongoing project "Mobilizador 5G".

The work made on this thesis contributed to two papers:

- "Using SDN and Slicing for Data Offloading over Heterogenous Networks Supporting non-3GPP Access", accepted to the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) 2018 with the authors Flávio Meneses, Rui Silva, David Santos, Daniel Corujo and Rui L. Aguiar.
- "An Integration of Slicing, NFV and SDN for Mobility Management in Corporate Environments" with the authors Flávio Meneses, Rui Silva, David Santos, Daniel Corujo and Rui L. Aguiar, submitted to the Transactions on Emerging Telecommunications Technologies journal.

This thesis was presented at the 25th Seminar of Rede Temática de Comunicações Móveis (RTCM) 2018.

5.3 Future Work

As future work, a network function can be defined in order to eliminate the need to manually configure the proxy for the web cache in the UE. The method should analyze that packets are coming from the UE and identify, modify and steer web related packets to the web cache. Also related with the web cache method, a new function could be used in order to let the SDN controller know which websites are already present in the cache, steering packets to it when it contains relevant content. Also, the web cache could be evolved to also support video caching, since video traffic accounts for a big part of total network traffic and it would benefit operators networks if the traffic between the edge and the core could be reduced.

Regarding MEC servers, an orchestrator should be integrated in order to manage resources, deploy in a simpler and faster way VNFs and applications, and additionally monitoring the system performance.

In this thesis, mobility management procedures were not attended, so it would be an essential enhancement to take into consideration, since the MEC paradigm is closely associated with mobility of the users due the physical location of the MEC servers.

Another complement to this architecture would be to tackle scalability of the services running on MEC servers. Since applications and services can have fluctuating resources requirements, depending on external factors (time of day, special events, load demand, network conditions) and it is crucial to prepare MEC servers with automatic scalability features according to performance requirements.

References

[1] M. Rouse, *Virtualization how-tos and learning guides - virtualization*, TechTarget. [Online]. Available: https://searchservervirtualization.techtarget.com/definition/virtualization (visited on 05/16/2018).

[2] OpenSource, *What is virtualization?*, Red Hat Enterprise Linux, Inc. [Online]. Available: https://opensource.com/resources/virtualization (visited on 05/16/2018).

[3] Red Hat Enterprise Linux, Inc., *What is virtualization?*, Red Hat Enterprise Linux, Inc. [Online]. Available: https://www.redhat.com/en/topics/virtualization/what-is-virtualization (visited on 05/16/2018).

[4] N. M. M. K. Chowdhury and R. Boutaba, «A survey of network virtualization», *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010, ISSN: 13891286. DOI: 10.1016/j.comnet.2009.10.017. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2009.10.017.

[5] PlanetLab Project, *An open platform for developing, deployind, and accessing planetray-scale services*, PlanetLab. [Online]. Available: https://www.planet-lab.org/ (visited on 05/16/2018).

[6] Zhao Liang et all, *D-3.2.1 virtualisation approach: evaluation and integration*, The FP7 4WARD Project. [Online]. Available: http://www.4ward-project.eu/index9d96.html?s=Deliverables (visited on 05/16/2018).

[7] Open infrastructure for at-scale networking and distributed systems research and education, *Global environment for network innovations (geni)*, Global Environment for Network Innovations (GENI). [Online]. Available: http://www.geni.net/ (visited on 05/16/2018).

[8] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, «In vini veritas : realistic and controlled network experimentation», *Proceedings of the 2006 conference on Applications technologies architectures and protocols for computer communications SIGCOMM 06*, pp. 3–14, 2006, ISSN: 01464833. DOI: 10.1145/1159913.1159916. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1159913.1159916.

[9] V. G. Nguyen, T. X. Do, and Y. H. Kim, «Sdn and virtualization-based lte mobile network architectures: a comprehensive survey», *Wireless Personal Communications*, vol. 86, no. 3, pp. 1401–1438, 2016, ISSN: 1572834X. DOI: 10.1007/s11277-015-2997-7.

[10] D. Kreutz and F. Ramos, «Software-defined networking: a comprehensive survey», *ArXiv preprint arXiv: ...*, p. 49, 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2371999. arXiv: 1406.0440. [Online]. Available: http://arxiv.org/abs/1406.0440.

[11] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, «A survey of software-defined networking: past, present, and future of programmable networks», *IEEE Communications Surveys and Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014, ISSN: 1553877X. DOI: 10.1109/SURV.2014.012214.00180. arXiv: 1406.0440.

[12] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, and M. Keshtgary, «A survey on sdn, the future of networking», *Journal of Advanced Computer Science & Technology*, vol. 3, no. 2, p. 232, 2014, ISSN: 2227-4332. DOI: 10.14419/jacst.v3i2.3754. [Online]. Available: http://www.sciencepubco.com/index.php/IJPE/article/view/3754.

[13] W. Stallings, «Sdn and openflow», *Acupuncture in Medicine*, vol. 16, no. 3, pp. 1–40, 2009, ISSN: 1521-4141. DOI: 10.1002/eji.201370053. [Online]. Available: http://www.cisco.com/c/en/us/

about/press/internet-protocol-journal/back-issues/table-contents-53/143-trill.html%7B%5C%%7D255Cnhttp://www.cisco.com/web/about/ac123/ac147/archived%7B%5C_%7Dissues/ipj%7B%5C_%7D14-3/143%7B%5C_%7Dtrill.html%7B%5C%%7D255Cnhttp://acupmed.bmjjournals.com/content/27/4/145.short%7B%5C%%7D25.

[14] O. N. Fundation, *Software-defined networking: the new norm for networks*, SDN White Paper, Open Network Fundation, Apr. 2012. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[15] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, *Software-defined networking (sdn): Layers and architecture terminology*, RFC7426, Jan. 2015. [Online]. Available: http://tools.ietf.org/rfc/rfc7426.txt.

[16] E. Haleplidis, *Overview of rfc7426: sdn layers and architecture terminology*, Overview of RFC7426: SDN Layers and Architecture Terminology, 2017. [Online]. Available: https://sdn.ieee.org/newsletter/september-2017/overview-of-rfc7426-sdn-layers-and-architecture-terminology.

[17] I. T. Union, *Itu-t recommendation itu-t y.3300: framework of software-defined networking (june 2014)*, ITU-T Recommendation ITU-T Y.3300, 2014. [Online]. Available: http://handle.itu.int/11.1002/1000/12168.

[18] O. N. Fundation, *Onf tr-502 sdn architecture (june 2014)*, SDN ARCH 1.0 06062014, 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR%7B%5C_%7DSDN%7B%5C_%7DARCH%7B%5C_%7D1.0%7B%5C_%7D06062014.pdf.

[19] J. Halpern and J. H. Salim, *Forwarding and control element separation (forces) forwarding element model*, RFC 5812 (Proposed Standard), Internet Engineering Task Force, Mar. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc5812.txt.

[20] O. N. Fundation, *Openflow switch specification - version 1.5.1 (protocol version 0x06)*, ONF TS-025, 2015. [Online]. Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[21] M. Bjorklund, *Yang - a data modeling language for the network configuration protocol (netconf)*, RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6020.txt.

[22] R. Presuhn, *Management information base (mib) for the simple network management protocol (snmp)*, RFC 3418 (INTERNET STANDARD), Internet Engineering Task Force, Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3418.txt.

[23] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, «Openflow: enabling innovation in campus networks», *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008, ISSN: 01464833. DOI: 10.1145/1355734.1355746. arXiv: arXiv:1406.0440v1. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1355734.1355746.

[24] O.N.F., «Software-defined networking: the new norm for networks», *ONF White Paper*, vol. 2, pp. 2–6, 2012. DOI: citeulike-article-id:12475417.

[25] O. N. Fundation, *Openflow switch specification - version 1.0.0 (protocol version 0x0)*, ONF TS-001, Open Network Fundation, Dec. 2009. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf.

[26] F. Hu, Q. Hao, and K. Bao, *A survey on software-defined network and openflow: from concept to implementation*, 2014. DOI: 10.1109/COMST.2014.2326417.

[27] D. Turull, M. Hidell, and P. Sjödin, «Performance evaluation of openflow controllers for network virtualization», in *2014 IEEE 15th International Conference on High Performance Switching and Routing, HPSR 2014*, 2014, pp. 50–56, ISBN: 9781479916337. DOI: 10.1109/HPSR.2014.6900881.

[28] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, «Nox: towards an operating system for networks», *Computer Communication Review*, vol. 38, pp. 105–110, 2008.

[29] Z. Cai, A. Cox, and E. T. S. Ng, «Maestro: a system for scalable openflow control», *Cs.Rice.Edu*, p. 10, 2011. DOI: Tech.Rep.TR10-08. [Online]. Available: http://www.cs.rice.edu/%7B~%7Deugeneng/papers/TR10-11.pdf.

[30] D. Erickson, «The beacon openflow controller», in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, 2013, p. 13, ISBN: 9781450321785. DOI: 10.1145/2491185.2491189. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2491185.2491189.

[31] Trema, *Trema an open source modular framework for developing openflow controllers in ruby/c*. [Online]. Available: https://github.com/trema (visited on 05/17/2018).

[32] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, «On scalability of software-defined networking», *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013, ISSN: 01636804. DOI: 10.1109/MCOM.2013.6461198. arXiv: arXiv:1408.6760v1.

[33] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, a. R. Curtis, and S. Banerjee, «Devoflow: cost-effective flow management for high performance enterprise networks», *Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks*, p. 1, 2010, ISSN: 1450304095. DOI: 10.1145/1868447.1868448. [Online]. Available: http://dl.acm.org/citation.cfm?id=1868448%7B%5C%%7D5Cnpapers2://publication/uuid/E7B32420-55BB-4C56-A3BF-7D9F1B9DE03D.

[34] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, S. Shenker, *et al.*, «Onix: a distributed control platform for large-scale production networks», in *9th USENIX Conference on Operating Systems Design and Implementation*, 2010, pp. 1–6, ISBN: 978-1-931971-79-9. DOI: 10.1.1.186.3537. arXiv: 9809069v1 [arXiv:gr-qc]. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924968.

[35] D. Drutskoy, E. Keller, and J. Rexford, «Scalable network virtualization in software-defined networks», *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, 2013, ISSN: 10897801. DOI: 10.1109/MIC.2012.144.

[36] A. Azzouni, O. Braham, T. M. T. Nguyen, G. Pujolle, and R. Boutaba, «Fingerprinting openflow controllers: the first step to attack an sdn control plane», in *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, 2016, ISBN: 9781509013289. DOI: 10.1109/GLOCOM.2016.7841843. arXiv: 1611.02370.

[37] Linux Foundation, *Opendaylight*, Linux Foundation. [Online]. Available: https://www.opendaylight.org/ (visited on 05/17/2018).

[38] S. Kaur, J. Singh, and N. S. Ghumman, «Network programmability using pox controller», *International Conference on Communication, Computing & Systems*, p. 5, 2014. DOI: 10.13140/RG.2.1.1950.6961.

[39] Project Floodlight, *Floodlight*, 2017. [Online]. Available: http://www.projectfloodlight.org/floodlight/.

[40] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, and B. Lantz, «Onos: towards an open, distributed sdn os», *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN '14*, 2014. DOI: 10.1145/2620728.2620744.

[41] Ryu, *Ryu is a component-based software defined networking framework.* [Online]. Available: https://osrg.github.io/ryu/ (visited on 05/17/2018).

[42] European Telecommunications Standards Institute (ETSI), *Network functions virtualisation – introductory white paper*, European Telecommunications Standards Institute (ETSI). [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf (visited on 05/16/2018).

[43] M. Richart, J. Baliosian, J. Serrat, and J. L. Gorricho, «Resource slicing in virtual wireless networks: a survey», *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016, ISSN: 19324537. DOI: 10.1109/TNSM.2016.2597295.

[44] STANDARDS INSTITUTIONS, «Gs nfv 002 - v1.2.1 - network functions virtualisation (nfv); architectural framework», *Tbd*, vol. 1, pp. 1–21, 2014.

[45] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, «Network function virtualization: state-of-the-art and research challenges», *IEEE Communications Surveys and Tutorials*,

vol. 18, no. 1, pp. 236–262, 2016, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2477041. arXiv: 1509.07675.

[46] G. Specification, «Network functions virtualisation (nfv); terminology for main concepts in nfv», *Gs Nfv 003 - V1.1.1*, vol. 1, pp. 1–10, 2013. DOI: DGS/NFV-0011.

[47] ——, «Network functions virtualisation (nfv); management and orchestration», *ETSI GS NFV-MAN 001 V1.1.1 (2014-12)*, 2014.

[48] M. Chiosi, D. Clarke, P. Willis Cablelabs, C. Donley, L. Johnson Centurylink, M. Bugenhagen, J. Feger, W. Khan, C. China, H. Cui, C. Chen China Deng, Telecom, L. Baohua, S. Zhenqiang, and S. Wright, *Network functions virtualisation (nfv) network operator perspectives on industry progress (pdf download available).pdf*, 2013.

[49] NIST, *Nist definition of cloud computing*, 2016. [Online]. Available: http://www.nist.gov/itl/cloud/.

[50] B. Abbasov, «Cloud computing: state of the art reseach issues», in *8th IEEE International Conference on Application of Information and Communication Technologies, AICT 2014 - Conference Proceedings*, 2014, ISBN: 9781479941209. DOI: 10.1109/ICAICT.2014.7035932.

[51] 3GPP, «System architecture for the 5g system», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, Mar. 2018, Version 15.1.0. [Online]. Available: http://www.3gpp.org/%5C-DynaReport/%5C-23501.htm.

[52] ——, «Procedures for the 5g system», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.502, Mar. 2018, Version 15.1.0. [Online]. Available: http://www.3gpp.org/%5C-DynaReport/%5C-23502.htm.

[53] ——, «Policy and charging control framework for the 5g system; stage 2», 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.503, Mar. 2018, Version 15.1.0. [Online]. Available: http://www.3gpp.org/%5C-DynaReport/%5C-23503.htm.

[54] Cisco Mobile, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper*. 2017, pp. 2016–2021, ISBN: 1454457600. DOI: 1465272001663118. arXiv: 1454457600809267.

[55] E. Cau, M. Corici, P. Bellavista, L. Foschini, G. Carella, A. Edmonds, and T. M. Bohnert, «Efficient exploitation of mobile edge computing for virtualized 5g in epc architectures», in *Proceedings - 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2016*, 2016, pp. 100–109, ISBN: 9781509017546. DOI: 10.1109/MobileCloud.2016.24.

[56] G. Orsini, D. Bade, and W. Lamersdorf, «Computing at the mobile edge: designing elastic android applications for computation offloading», in *Proceedings - 2015 8th IFIP Wireless and Mobile Networking Conference, WMNC 2015*, 2016, pp. 112–119, ISBN: 9781509003518. DOI: 10.1109/WMNC.2015.10. arXiv: 1510.00888.

[57] P. Mach and Z. Becvar, *Mobile edge computing: a survey on architecture and computation offloading*, 2017. DOI: 10.1109/COMST.2017.2682318. arXiv: 1702.05309.

[58] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, «A survey of mobile cloud computing: architecture, applications, and approaches», *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013, ISSN: 15308669. DOI: 10.1002/wcm.1203. arXiv: arXiv:1405.1155v1.

[59] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, «Mobile edge computing: a survey», *IEEE Internet of Things Journal*, pp. 1–1, 2017, ISSN: 2327-4662. DOI: 10.1109/JIOT.2017.2750180. [Online]. Available: http://ieeexplore.ieee.org/document/8030322/.

[60] M.-E. C. I. Initiative, «Mobile-edge computing – introductory technical white paper», pp. 1–36, 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge%7B%5C_%7Dcomputing%7B%5C_%7D-%7B%5C_%7Dintroductory%7B%5C_%7Dtechnical%7B%5C_%7Dwhite%7B%5C_%7Dpaper%7B%5C_%7Dv1%2018-09-14.pdf.

[61] Y. Mao and C. You and J. Zhang and K. Huang and K. B. Letaief, «A survey on mobile edge computing: the communication perspective», *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. DOI: 10.1109/COMST.2017.2745201.

[62] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, «Internet of things: a survey on enabling technologies, protocols, and applications», *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, ISSN: 1553877X. DOI: 10.1109/COMST.2015.2444095. arXiv: arXiv:1011.1669v3.

[63] G. P. Fettweis, «The tactile internet: applications and challenges», *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014, ISSN: 15566072. DOI: 10.1109/MVT.2013.2295069.

[64] J. Moar, *Smart wireless devices and the internet of me*, Smart Wireless Devices and the Internet of Me, White Paper, 2015. [Online]. Available: http://itersnews.com/wp-content/uploads/experts/2015/03/96079Smart-Wireless-Devices-and-the-Internet-of-Me.pdf.

[65] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, «A survey of mobile cloud computing application models», *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 393–413, 2014, ISSN: 1553-877X. DOI: 10.1109/SURV.2013.062613.00160. arXiv: 1105.3232. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6553297.

[66] M. Satyanarayanan, «Fundamental challenges in mobile computing», *Annual ACM Symposium on Principles of Distributed Computing*, pp. 1–7, 1996. DOI: 10.1145/248052.248053.

[67] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, «The case for vm-based cloudlets in mobile computing», *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009, ISSN: 15361268. DOI: 10.1109/MPRV.2009.82.

[68] Q. Fan and L. Liu, «A survey of challenging issues and approaches in mobile cloud computing», in *Parallel and Distributed Computing, Applications and Technologies, PDCAT Proceedings*, 2017, pp. 87–90, ISBN: 9781509050819. DOI: 10.1109/PDCAT.2016.032.

[69] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, «Fog computing and its role in the internet of things», *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, 2012, ISSN: 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513. arXiv: arXiv:1502.01815v3. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513%7B%5C%%7D5Cnpapers2://publication/doi/10.1145/2342509.2342513.

[70] Cisco Systems, «Fog computing and the internet of things: extend the cloud to where the things are», *Www.Cisco.Com*, 2016, ISSN: 23274662. DOI: 10.1109/HotWeb.2015.22. arXiv: arXiv:1502.01815v3.

[71] S. Yi, C. Li, and Q. Li, «A survey of fog computing: concepts, applications and issues», *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata '15*, 2015. DOI: 10.1145/2757384.2757397.

[72] Cisco Systems, *Cisco iox*, Cisco Systems. [Online]. Available: https://www.cisco.com/c/en/us/products/cloud-systems-management/iox/index.html (visited on 05/19/2018).

[73] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, «On multi-access edge computing: a survey of the emerging 5g network edge cloud architecture and orchestration», *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017, ISSN: 1553877X. DOI: 10.1109/COMST.2017.2705720.

[74] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, «Impact of cloudlets on interactive mobile cloud applications», in *Proceedings of the 2012 IEEE 16th International Enterprise Distributed Object Computing Conference, EDOC 2012*, 2012, pp. 123–132, ISBN: 9780769547855. DOI: 10.1109/EDOC.2012.23.

[75] M. E. C. Initiative, *Executive briefing – mobile edge computing (mec) initiative*, Executive Briefing – Mobile Edge Computing (MEC) Initiative, 2014. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mec%20executive%20brief%20v1%2028-09-14.pdf.

[76] M.-E. C. (E. I. S. G. (ISG), «Mobile edge computing (mec); framework and reference architecture », pp. 1–16, 2016. [Online]. Available: http://www.etsi.org/deliver/etsi%7B%5C_%7Dgs/MEC/001%7B%5C_%7D099/003/01.01.01%7B%5C_%7D60/gs%7B%5C_%7DMEC003v010101p.pdf.

[77] Mobile-Edge Computing (MEC) ETSI Industry Specification Group (ISG), «Mobile edge computing (mec) terminology», pp. 1–16, 2016. [Online]. Available: http://www.etsi.org/deliver/etsi%7B%5C_%7Dgs/MEC/001%7B%5C_%7D099/001/01.01.01%7B%5C_%7D60/gs%7B%5C_%7Dmec001v010101p.pdf.

[78] M.-E. C. (E. I. S. G. (ISG), «Mobile-edge computing (mec); service scenarios - group specification», pp. 1–16, 2015. [Online]. Available: http://www.etsi.org/deliver/etsi%7B%5C_%7Dgs/MEC-IEG/001%7B%5C_%7D099/004/01.01.01%7B%5C_%7D60/gs%7B%5C_%7DMEC-IEG004v010101p.pdf.

[79] Sophia Antipolis, *Etsi first mobile edge computing proof of concepts at mec world congress*, 2016. [Online]. Available: http://www.etsi.org/news-events/news/1119-2016-09-news-etsi-first-mobile-edge-computing-proof-of-concepts-at-mec-world-congress.

[80] Guy Daniels, *Edge computing prepares for a multi-access future*, 2016. [Online]. Available: https://www.telecomtv.com/content/mec/edge-computing-prepares-for-a-multi-access-future-13986/.

[81] ETSI, «Mobile edge computing (mec); technical requirements», *ETSI White Paper*, vol. 1, pp. 1–40, 2016. [Online]. Available: http://www.etsi.org/standards-search.

[82] N. Takahashi, H. Tanaka, and R. Kawamura, «Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing», in *Proceedings - 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, MobileCloud 2015*, 2015, pp. 233–234, ISBN: 9781479989775. DOI: 10.1109/MobileCloud.2015.23.

[83] Y. Zhang, H. Liu, L. Jiao, and X. Fu, «To offload or not to offload: an efficient code partition algorithm for mobile cloud computing», in *2012 1st IEEE International Conference on Cloud Networking, CLOUDNET 2012 - Proceedings*, 2012, pp. 80–86, ISBN: 9781467327985. DOI: 10.1109/CloudNet.2012.6483660.

[84] R. Buyya and A. V. Dastjerdi, *Internet of Things: Principles and Paradigms*. 2016, pp. 1–354, ISBN: 9780128093474. DOI: 10.1016/C2015-0-04135-1. arXiv: arXiv:1309.7735v4.

[85] J. Dolezal, Z. Becvar, and T. Zeman, «Performance evaluation of computation offloading from mobile device to the edge of mobile network», in *2016 IEEE Conference on Standards for Communications and Networking, CSCN 2016*, 2016, ISBN: 9781509038626. DOI: 10.1109/CSCN.2016.7785153.

[86] Z. Chen, L. Jiang, W. Hu, K. Ha, B. Amos, P. Pillai, A. Hauptmann, and M. Satyanarayanan, «Early implementation experience with wearable cognitive assistance applications», *Proceedings of the 2015 Workshop on Wearable Systems and Applications*, pp. 33–38, 2015. DOI: 10.1145/2753509.2753517. [Online]. Available: http://doi.acm.org/10.1145/2753509.2753517.

[87] T. Verbelen, P. Simoens, F. D. Turck, and B. Dhoedt, «Cloudlets : bringing the cloud to the mobile user», *Mcs 2012*, pp. 29–36, 2012. DOI: 10.1145/2307849.2307858.

[88] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Mendez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung, «Augmented brain computer interaction based on fog computing and linked data», in *Proceedings - 2014 International Conference on Intelligent Environments, IE 2014*, 2014, pp. 374–377, ISBN: 9781479929474. DOI: 10.1109/IE.2014.54.

[89] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, «Vr is on the edge : how to deliver 360° videos in mobile networks simone», *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network - VR/AR Network '17*, pp. 30–35, 2017. DOI: 10.1145/3097895.3097901. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3097895.3097901.

[90] P. A. Heidenreich, J. G. Trogdon, O. A. Khavjou, J. Butler, K. Dracup, M. D. Ezekowitz, E. A. Finkelstein, Y. Hong, S. C. Johnston, A. Khera, D. M. Lloyd-Jones, S. A. Nelson, G. Nichol, D. Orenstein, P. W. Wilson, and Y. J. Woo, «Forecasting the future of cardiovascular disease in the united states: a policy statement from the american heart association», *Circulation*, vol. 123, no. 8, pp. 933–944, 2011, ISSN: 00097322. DOI: 10.1161/CIR.0b013e31820a55f5.

[91] Y. Cao, S. Chen, P. Hou, and D. Brown, «Fast: a fog computing assisted distributed analytics system to monitor fall for stroke mitigation», in *Proceedings of the 2015 IEEE International Conference on Networking, Architecture and Storage, NAS 2015*, 2015, pp. 2–11, ISBN: 9781467378918. DOI: 10.1109/NAS.2015.7255196.

[92] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, «Fog computing: principles, architectures, and applications», in *Internet of Things: Principles and Paradigms*, 2016, pp. 61–75, ISBN: 9780128093474. DOI: 10.1016/B978-0-12-805395-9.00004-6. arXiv: 1601.02752.

[93] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, «Smart items , fog and cloud computing as enablers of servitization in healthcare», *Sensors & Transducers*, vol. 185, no. 2, pp. 121–128, 2015, ISSN: 23068515.

[94] X. Sun and N. Ansari, «Edgeiot: mobile edge computing for the internet of things», *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016, ISSN: 01636804. DOI: 10.1109/MCOM.2016.1600492CM. arXiv: 1709.00462.

[95] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, «Edge computing enabling the internet of things», in *IEEE World Forum on Internet of Things, WF-IoT 2015 - Proceedings*, 2015, pp. 603–608, ISBN: 9781509003655. DOI: 10.1109/WF-IoT.2015.7389122.

[96] M. Aazam and E. N. Huh, «Fog computing and smart gateway based communication for cloud of things», in *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*, 2014, pp. 464–470, ISBN: 9781479943586. DOI: 10.1109/FiCloud.2014.83. arXiv: 1305.0982.

[97] H. Tanaka, M. Yoshida, K. Mori, and N. Takahashi, «Multi-access edge computing: a survey», *Journal of Information Processing*, vol. 26, pp. 87–97, 2018.

[98] G. C. Fox, S. Kamburugamuve, and R. D. Hartman, «Architecture and measured characteristics of a cloud based internet of things», in *Proceedings of the 2012 International Conference on Collaboration Technologies and Systems, CTS 2012*, 2012, pp. 6–12, ISBN: 9781467313803. DOI: 10.1109/CTS.2012.6261020.

[99] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, «Osmotic computing: a new paradigm for edge/cloud integration», *IEEE Cloud Computing*, vol. 3, no. 6, pp. 76–83, 2016, ISSN: 23256095. DOI: 10.1109/MCC.2016.124.

[100] Docker Website, *What is docker?*, 2016. [Online]. Available: docker.com/what-docker/.

[101] Kubernetes. (2018). {production-grade container orchestration} - automated container deployment, scaling, and management, [Online]. Available: https://kubernetes.io/.

[102] M. Sapienza, E. Guardo, M. Cavallo, G. La Torre, G. Leombruno, and O. Tomarchio, «Solving critical events through mobile edge computing: an approach for smart cities», in *2016 IEEE International Conference on Smart Computing, SMARTCOMP 2016*, 2016, ISBN: 9781509008988. DOI: 10.1109/SMARTCOMP.2016.7501719.

[103] A. Anjum, T. Abdullah, M. Tariq, Y. Baltaci, and N. Antonopoulos, «Video stream analysis in clouds: an object detection and classification framework for high performance video analytics», *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, p. 1, 2016, ISSN: 2168-7161 VO - PP. DOI: 10.1109/TCC.2016.2517653.

[104] A. Perrott, *Computer technology increasingly aids traffic management*, The move to IP, 2009. [Online]. Available: http://www.itsinternational.com/categories/detection-monitoring-machine-vision/features/computer-technology-increasingly-aids-traffic-management/.

[105] K. Hong and D. Lillethun, «Mobile fog: a programming model for large-scale applications on the internet of things», *Proceedings of the second ACM SIGCOMM Workshop on Mobile Cloud Computing*, pp. 15–20, 2013. DOI: 10.1145/2491266.2491270. [Online]. Available: http://dl.acm.org/citation.cfm?id=2491270.

[106] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, «Edge analytics in the internet of things», *IEEE Pervasive Computing*, vol. 14, no. 2, pp. 24–31, 2015, ISSN: 15361268. DOI: 10.1109/MPRV.2015.32.

[107] H. P. Enterprise, *Edge video analytics*, Edge Video Analytics - HPE Edgeline IoT systems with IDOL Media Server enable exceptional Scene Analysis & Object recognition performance, 2016. [Online]. Available: https://support.hpe.com/hpsc/doc/public/display?docId=c05336736.

[108] E. Uhlemann, «Introducing connected vehicles [connected vehicles]», *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, 2015, ISSN: 15566072. DOI: 10.1109/MVT.2015.2390920.

[109] P. Papadimitratos, A. D. L. Fortelle, M. Paristech, K. Evenssen, and Q.-f. Asa, «Vehicular communication systems : enabling technologies , applications , and future outlook on intelligent transportation», *IEEE Communications Magazine*, no. November, pp. 84–95, 2009, ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.5307471.

[110] 5. A. Association, *{5g automotive association} - the case for cellular v2x for safety and cooperative driving*, The Case for Cellular V2X for Safety and Cooperative Driving, 2016. [Online]. Available: http://5gaa.org/wp-content/uploads/2017/10/5GAA-whitepaper-23-Nov-2016.pdf.

[111] 5G-PPP, *5g automotive vision*, White Paper on Automotive Vertical Sectors, 2015. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Automotive-Vertical-Sectors.pdf.

[112] 5.-P. A. W. Group, *{5g-ppp automotive working group } - a study on 5g v2x deployment*, A study on 5G V2X Deployment, 2018. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/02/5G-PPP-Automotive-WG-White-Paper%7B%5C_%7DFeb.2018.pdf.

[113] DENSO Corporation, Ericsson (NASDAQ-ERIC), Intel Corporation, Nippon Telegraph and Telephone Corporation (NTT), NTT DOCOMO, INC., Toyota InfoTechnology Center Co., Ltd. and Toyota Motor Corporation, *Industry leaders to form consortium for network and computing infrastructure of automotive big data*, Industry leaders to form consortium for network and computing infrastructure of automotive big data, 2017. [Online]. Available: https://newsroom.toyota.co.jp/en/detail/18135029/.

[114] D. Grewe, M. Wagner, M. Arumaithurai, I. Psaras, and D. Kutscher, «Information-centric mobile edge computing for connected vehicle environments: challenges and research directions», in *Proceedings of the Workshop on Mobile Edge Communications*, ser. MECOMM '17, New York, NY, USA: ACM, 2017, pp. 7–12, ISBN: 978-1-4503-5052-5. DOI: 10.1145/3098208.3098210. [Online]. Available: http://doi.acm.org/10.1145/3098208.3098210.

[115] M. Amadeo, C. Campolo, and A. Molinaro, «Information-centric networking for connected vehicles: a survey and future perspectives», *IEEE Communications Magazine*, vol. 54, no. 2, pp. 98–104, 2016, ISSN: 01636804. DOI: 10.1109/MCOM.2016.7402268.

[116] Nokia, *Utm infrastructure and connected society*, 2016. [Online]. Available: https://rpas-civops.com/wp-content/uploads/2016/11/S7.2%7B%5C_%7DNokia%7B%5C_%7DDE%7B%5C_%7DV1.pdf.

[117] S. Retal, M. Bagaa, T. Taleb, and H. Flinck, «Content delivery network slicing: qoe and cost awareness», in *IEEE International Conference on Communications*, 2017, ISBN: 9781467389990. DOI: 10.1109/ICC.2017.7996499.

[118] P. A. Frangoudis, L. Yala, A. Ksentini, and T. Taleb, «An architecture for on-demand service deployment over a telco cdn», in *2016 IEEE International Conference on Communications, ICC 2016*, 2016, ISBN: 9781479966646. DOI: 10.1109/ICC.2016.7510921.

[119] Y. Jararweh, L. Tawalbeh, F. Ababneh, and F. Dosari, «Resource efficient mobile computing using cloudlet infrastructure», in *Proceedings - IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2013*, 2013, pp. 373–377, ISBN: 9780768551593. DOI: 10.1109/MSN.2013.75.

[120] W. Zhu, C. Luo, J. Wang, and S. Li, «Multimedia cloud computing», *IEEE Signal Processing Magazine*, vol. 28, no. 3, pp. 59–69, 2011, ISSN: 10535888. DOI: 10.1109/MSP.2011.940269.

[121] Y. Jin, Y. Wen, and C. Westphal, «Optimal transcoding and caching for adaptive streaming in media cloud: an analytical approach», *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1914–1925, 2015, ISSN: 10518215. DOI: 10.1109/TCSVT.2015.2402892.

[122] D. Liu, B. Chen, C. Yang, and A. F. Molisch, «Caching at the wireless edge: design aspects, challenges, and future directions», *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016, ISSN: 01636804. DOI: 10.1109/MCOM.2016.7565183.

[123] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, «Mobile edge computing potential in making cities smarter», *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017, ISSN: 01636804. DOI: 10.1109/MCOM.2017.1600249CM.

[124] European Telecommunications Standards Institute (ETSI), *Mobile edge computing - a key technology towards 5g*, ETSI White Paper No. 11 - Mobile Edge Computing - A key technology towards 5G, 2015. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi%7B%5C_%7Dwp11%7B%5C_%7Dmec%7B%5C_%7Da%7B%5C_%7Dkey%7B%5C_%7Dtechnology%7B%5C_%7Dtowards%7B%5C_%7D5g.pdf.

[125] International Telecommunication Union - Radiocommunication Study Groups, *International mobile telecommunications (imt) vision – "framework and overall objectives of the future development of imt for 2020 and beyond"*, International Mobile Telecommunications (IMT) Vision – "Framework and overall objectives of the future development of IMT for 2020 and beyond", 2015. [Online]. Available: https://www.itu.int/dms%7B%5C_%7Dpubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I%7B%5C%%7D21%7B%5C%%7D21PDF-E.pdf.

[126] S. Antipolis, *Etsi multi-access edge computing keeps pace with 5g*, ETSI Multi-access Edge Computing keeps pace with 5G, 2018. [Online]. Available: http://www.etsi.org/news-events/news/1277-2018-02-news-etsi-multi-access-edge-computing-keeps-pace-with-5g.

[127] K. A. J. C. Y. F. W. F. F. F. D. F. A. L. A. M. D. P. D. S. C. W. K.-W. W. Z. Z. Fabio Giust Gianluca Verin, *Mec deployments in 4g and evolution towards 5g*, ETSI White Paper - MEC Deployments in 4G and Evolution Towards 5G, 2018. [Online]. Available: http://www.etsi.org/images/files/ETSIWhitePapers/etsi%7B%5C_%7Dwp24%7B%5C_%7DMEC%7B%5C_%7Ddeployment%7B%5C_%7Din%7B%5C_%7D4G%7B%5C_%7D5G%7B%5C_%7DFINAL.pdf.

[128] European Telecommunications Standards Institute, *Ts 123 401 - v9.3.0 - lte; general packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access (3gpp ts 23.401 version 9.3.0 release 9)*. [Online]. Available: https://ia800905.us.archive.org/6/items/etsi_ts_123_401_v09.03.00/ts_123401v090300p.pdf (visited on 05/19/2018).

[129] Next Generation Mobile Networks Alliance, *Ngmn 5g white paper*, NGMN, Feb. 2015. [Online]. Available: https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf (visited on 05/16/2018).

[130] 3GPP, «Study on architecture for next generation system», 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.799, Dec. 2016, Version 14.0.0. [Online]. Available: http://www.3gpp.org/%5C-DynaReport/%5C-23799.htm.

[131] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, *From network sharing to multi-tenancy: the 5g network slice broker*, 2016. DOI: 10.1109/MCOM.2016.7514161. arXiv: 1605.01201.

[132] M. A. Puente, Z. Becvar, M. Rohlik, F. Lobillo, and E. C. Strinati, «A seamless integration of computationally - enhanced base stations into mobile networks towards 5g», in *First 5G architecture worksohp, VTC'15*, 2015, pp. 1–5, ISBN: 9781479980888.

[133] F. Lobillo, Z. Becvar, M. A. Puente, P. Mach, F. Lo Presti, F. Gambetti, M. Goldhamer, J. Vidal, A. K. Widiawan, and E. Calvanesse, «An architecture for mobile computation offloading on cloud-enabled lte small cells», in *2014 IEEE Wireless Communications and Networking Conference Workshops, WCNCW 2014*, 2014, pp. 1–6, ISBN: 9781479930869. DOI: 10.1109/WCNCW.2014.6934851.

[134] H2020 European Project, *Small cells coordination for multi-tencancy and edge services*, H2020 European Project. [Online]. Available: http://www.sesame-h2020-5g-ppp.eu/ (visited on 05/16/2018).

[135] I. Giannoulakis, E. Kafetzakis, I. Trajkovska, P. S. Khodashenas, I. Chochliouros, C. Costa, I. Neokosmidis, and P. Bliznakov, «The emergence of operator-neutral small cells as a strong case for cloud computing at the mobile edge», *Transactions on Emerging Telecommunications Technologies*, 2016, ISSN: 21613915. DOI: 10.1002/ett.3077. arXiv: arXiv:1307.8198v1.

[136] S. Wang, G.-H. Tu, R. Ganti, K. He, Ting and Leung, H. Tripp, K. Warr, and M. Zafer, «Mobile micro-cloud: application classification, mapping, and deployment», *Proc. Annual Fall Meeting of ITA (AMITA)*, 2013.

[137] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van Der Merwe, and K. Webb, «Mobiscud: a fast moving personal cloud in the mobile network *», *ACM SIGCOM 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, 2015. DOI: 10.1145/2785971.2785979.

[138] T. Taleb and A. Ksentini, «Follow me cloud: interworking federated clouds and distributed mobile networks», *IEEE Network*, 2013, ISSN: 08908044. DOI: 10.1109/MNET.2013.6616110.

[139] T. Taleb, A. Ksentini, and P. Frangoudis, «Follow-me cloud: when cloud services follow mobile users», *IEEE Transactions on Cloud Computing*, 2016, ISSN: 2168-7161. DOI: 10.1109/TCC.2016.2525987.

[140] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, «Concert: a cloud-based architecture for next-generation cellular systems», *IEEE Wireless Communications*, 2014, ISSN: 15361284. DOI: 10.1109/MWC.2014.7000967. arXiv: 1410.0113.

[141] OPEN BATON, *Open baton - an extensible and customizable nfv mano-compliant framework*. [Online]. Available: https://openbaton.github.io/ (visited on 05/19/2018).

[142] Open Source Mano, *Osm - open source mano is an etsi-hosted project to develop an open source nfv management and orchestration (mano) software stack aligned with etsi nfv*. [Online]. Available: https://osm.etsi.org/ (visited on 05/19/2018).

[143] Cloudify, *Cloudify - cloud and nfv based orchestration*. [Online]. Available: https://cloudify.co/ (visited on 05/19/2018).

[144] Open Networking Foundation, *M-cord is an open source reference solution for carriers deploying 5g mobile wireless networks*. [Online]. Available: https://www.opennetworking.org/m-cord/ (visited on 05/19/2018).

[145] ——, *Onos - open source network operating system*. [Online]. Available: https://onosproject.org (visited on 05/19/2018).

[146] ——, *Xos:service orchestration for cord - technical white paper*. [Online]. Available: http://onosproject.org/wp-content/uploads/2015/06/Technical-Whitepaper-XOS.pdf (visited on 05/19/2018).

[147] TNOVA, *Tnova - network functions as-a-service over virtualized infrastructures*. [Online]. Available: http://www.t-nova.eu (visited on 05/19/2018).

[148] ONAP - Open Network Automation Platform, *Onap - case solution architecture - white paper*. [Online]. Available: https://www.onap.org/wp-content/uploads/sites/20/2018/06/ONAP_CaseSolution_Architecture_0618FNL.pdf (visited on 05/19/2018).

[149] E. Piri, P. Ruuska, T. Kanstren, J. Makela, J. Korva, A. Hekkala, A. Pouttu, O. Liinamaa, M. Latva-Aho, K. Vierimaa, and H. Valasma, «5gtn: a test network for 5g application development and testing», in *EUCNC 2016 - European Conference on Networks and Communications*, 2016, pp. 313–318, ISBN: 9781509028931. DOI: 10.1109/EuCNC.2016.7561054.

[150] 4G-Portal. (2016). C-ran trial inside sports stadium conducted by nokia and china mobile, [Online]. Available: http://4g-portal.com/c-ran-trial-inside-sports-stadium-conducted-by-nokia-and-china-mobile.

[151] Nokia. (). Small cells deliver cost-effective capacity and coverage, indoors and outdoors, and are key to network innovation, [Online]. Available: https://networks.nokia.com/products/small-cells.

[152] N. Networks, *Airframe data center solution*, Nokia. [Online]. Available: https://networks.nokia.com/solutions/airframe-data-center-solution (visited on 05/16/2018).

[153] S. Daeuble. (). Small cells and mobile edge computing cover all the bases for taiwan baseball fans, [Online]. Available: https://www.nokia.com/en%7B%5C_%7Dint/blog/small-cells-mobile-edge-computing-cover-bases-taiwan-baseball-fans.

[154] R. Roman, J. Lopez, and M. Mambo, «Mobile edge computing, fog et al.: a survey and analysis of security threats and challenges», *Future Generation Computer Systems*, 2018, ISSN: 0167739X. DOI: 10.1016/j.future.2016.11.009. arXiv: 1602.00484.

[155] A. Mtibaa, K. A. Harras, and H. Alnuweiri, «Friend or foe? detecting and isolating malicious nodes in mobile edge computing platforms», in *Proceedings - IEEE 7th International Conference on Cloud Computing Technology and Science, CloudCom 2015*, 2016, ISBN: 9781467395601. DOI: 10.1109/CloudCom.2015.40. arXiv: 1510.00888.

[156] I. Stojmenovic and S. Wen, «The fog computing paradigm: scenarios and security issues», *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, 2014, ISSN: 2300-5963. DOI: 10.15439/2014F503.

[157] European Telecommunications Standards Institute (ETSI), «Mec deployments in 4g and evolution towards 5g - etsi white paper nº 24», ETSI, Tech. Rep. DOI: ISBNNo.979-10-92620-18-4.

[158] D. Kristol and L. Montulli, *Http state management mechanism*, RFC2109, Feb. 1997. [Online]. Available: http://tools.ietf.org/rfc/rfc2109.txt.

[159] N. Mohabbati Kalejahi, H. Akbaripour, and E. Masehian, *Basic and hybrid imperialist competitive algorithms for solving the non-attacking and non-dominating n-queens problems*, 2015.

[160] J. Postel, *Internet control message protocol*, RFC 792 (INTERNET STANDARD), Updated by RFCs 950, 4884, 6633, 6918, Internet Engineering Task Force, Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc792.txt.