SEMANTIC

end-to-end Slicing and data-drivEn autoMAtion of Next generation cellular neTworks with mobIle edge Clouds

Marie Skłodowska-Curie Actions (MSCA)
Innovative Training Networks (ITN)
H2020-MSCA-ITN-2019
861165 - SEMANTIC

WP2 – MEC/RAN integration and MEC-empowered enhancements

D2.1: SoA on MEC/RAN convergence and services

Contractual Date of Delivery:	M18
Actual Date of Delivery:	30/06/2021
Responsible Beneficiary:	FOG
Contributing beneficiaries:	POLITO, CTTC, UOA, FOG
Security:	Public
Nature:	Report
Version:	V2.1

Document Information

Version Date: 22/06/2021
Total Number of Pages: 88

Authors

Name	Organization	Email
Dr. Dionysis Xenakis	FOG	dionysis@fogus.gr
Diana Zhussip	FOG	dzhussip@fogus.gr
Madhura Adeppady	POLITO	madhura.adeppady@polito.it
Yevhenii Osadchuk	UOA	yosadchuk@di.uoa.gr
Vaishnavi Kasuluru	CTTC	vkasuluru@ctts.es
Prof. Carla Fabiana Chiasserini	POLITO	carla.chiasserini@polito.it
Prof. Paolo Giaccone	POLITO	paolo.giaccone@polito.it
Dr. Nikos Passas	UOA	passas@di.uoa.gr
Prof. Lazaros Merakos	UOA	merakos@di.uoa.gr

Document History

Revision	Date	Modification	Contact Person
V1.0	27/03/2021	Defining a preliminary table of contents	Dr. Dionysis Xenakis
V1.1	25/05/2021	Merging contributions of all ESRs	Mrs. Diana Zhussip
V2.0	01/06/2021	Editor changes	Mrs. Diana Zhussip
V2.1	02/06/2021	Editor changes	Prof. Carla Fabiana Chiasserini

Table of Contents

List of Acronyms and Abbreviations

Acronym	***Description***
3GPP	Third Generation Partnership Project
5G	5th generation (5G) mobile network
ABR	Adaptive Bitrate
AF	Application Function
AI/ML	Artificial Intelligence / Machine Learning
AMF	Access and Mobility Management Function
AP	Access Point
AUSF	Authentication Server Function
BBU	Baseband Unit
BFT	Byzantine Faulty Tolerant
biRNN	Bidirectional Recurrent Neural Network
BPP	Binomial Point Process
BS	Base Station
CAT	Cache Allocation Technology
CDF	Cumulative Distribution Function
CDN	Content Distribution Network
CFS	Customer Facing Service
CGI	Global Cell Identity
CHR	Cache Hit Ratio
CNN	Convolutional Neural Network
COTS	Common Off-The Shelf
CP	Content Provider
CRAN	Cloud Radio Access Network
CRF	Combined Recency and Frequency
CS	Content Server
D2D	Device-to-Device Communication
DASH	Dynamic adaptive streaming over HTTP
DDIO	Data Direct I/O
DN	Data Network
DNS	Domain Name System
DP	Dynamic Programming
DPDK	Data Plane Development Kit
EAS	Edge Application Server
EEC	Edge Enabler Client
EES	Edge Enabler Server
eMBB	Enhanced Mobile Broadband
EPC	Evolved Packet Core
ESN	Echo State Network
ETSI	European Telecommunications Standards Institute
EWMA	Exponential Weighted Moving Average
FBS	Femto Base Station
FIFO	First In First Out
FL	Federated Learning

FNN	Feedforward Neural Network
HAS	HTTP Adaptive Streaming
HCPP	Hard Core Point Process
HHM	Hidden Markov Model
IMEI	International Mobile Equipment Identity
KPI	Key Performance Indicator
LADN	Local Area Data Network
LFU	Least Frequently Used
LN	Lightning Network
LRFU	Least Recently/Frequently Used
LRU	Least Recently Used
LSTM	Long Short-Term Memory
MBA	Memory Bandwidth Allocation
MBS	Macro Base Station
MDF	Media Description File
MDP	Markov Decision Process
ME	Mobile Equipment
MEC	Multi-Access Edge Computing
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MLP	Multilayer Perceptron
mMTC	massive Machine Type Communication
MNO	Mobile Network Operator
MPD	Model Predictive Control
NAS	Non-Access Stratum
NEF	Network Exposure Function
NFV	Network Function Virtualization
NLP	Neural Language Processing
NN	Neural Network
NRF	Network Resource Function
NSSF	Network Slice Selection Function
O-RAN	Open-RAN
OSS	Operations Support System
OTT	Over-the-Top
PC	Payment Channel
PCF	Policy Control Function
PCP	Poisson Cluster Process
PDU	Protocol Data Unit
PoA	Proof-of-Authority
PoS	Proof-of-Stake
PoW	Proof-of-Work
PPP	Poisson Point Process
QoE	Quality of Experience
QOS	Quality of Service
RAN	Radio Access Network

RAT	Radio Access Technology
RDT	Resource Director Technology
ReLU	Rectified Linear Unit
RIC	RAN Intelligence Controller
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RRH	Remote Radio Head
RWP	Random Waypoint
SBA	Service-Based Architecture
SBS	Secondary Base Station
SC	Smart Contract
SCN	Small Cell Network
SINR	Signal to Interference Noise Ratio
SSC	Session and Service Continuity
TPS	Transactions per Second
UE	User Equipment
UPF	User Plane Function
URLLC	Ultra-Reliable Low Latency Communication
VL	Virtual Link
VNF	Virtual Network Function
VNFFG	Virtual Network Function Forwarding Graph
VoD	Video on Demand

List of Figures

List of Tables

1 Executive summary

This report includes the SEMANTIC ESR contributions towards the objectives of WP2 (MEC/RAN integration and MEC-empowered enhancements). More specifically, it summarizes the key findings of the ESRs towards Task 2.1 (SoA on MEC/RAN convergence and MEC services) that includes current scientific, industrial and standardization efforts on MEC architectures, services and ML tools, covering aspects of their integration in the 5G RAN. Architectural and functional upgrades for MEC/RAN integration are also discussed.

2 Introduction

The extremely prompt recent development of 5G and beyond mobile communication networks has established high technological requirements for ultra-low latency and high-capacity transmission. At the same time, telecommunication networks currently experience an ever-increased utilization, due to the appearance of more bandwidth-hungry applications, that require highly reliable and ultra-fast connectivity such as online video conferences, streaming platforms, AR/VR applications, and low-latency gaming. Furthermore, reinforced by the increasing number of wireless subscribers and extreme growth in mobile data traffic, digital media services experience huge demand over the recent years. This explosive demand for extremely high data rates increased by the growing demand for remote monitoring, control, and reconfiguration of cyber-physical infrastructures supporting our everyday life and working routine, has defined the technological direction of development of the 5th generation (5G) mobile network.

Supporting the vast amount of various demanding services and applications under the umbrella of the 5G ecosystem imposes an abstraction and cloudification of the heterogeneous network with the development of virtualization and softwarization along with effective edge utilization, depending on the performance objective. Dynamic pooling of edge network resources and capabilities using *multi-access edge computing* (MEC), specifically combined with the potential for migrating core network functions to the edge using *network function virtualization* (NFV), is the main enabler for reduced service creation times, massive edge network connectivity, multi-access and multi-service support in 5G. MEC, in particular, has brought enormous technological potential with the introduction of a novel concept of utilization control, computation, storage, and power resources closer to users.

Currently, the smooth service integration of available MEC capabilities in the 5G network is in its early development stage, meaning that the academic and industrial society performs exhaustive research towards the guarantee of service availability, smooth data delivery, and highly flexible resource utilization over joint MEC/5G Radio Access Network (RAN) infrastructures. The key challenging task for 5G in the context of homogeneous MEC/RAN convergence is creating a programmable and on-demand infrastructure, with logical and isolated pipes, that sets appropriate parameters and allocates resources and virtual network functions (VNFs) in the network slices, for exceptionally smooth and dynamic service provisioning.

The major objective of the project here is to design diverse architectural and functional improvements for the **integration of MEC services and resources** into the standard operation of 5G-and-beyond networks. What is more, this work aspires to fully realize the convergence of the radio, storage, and processing capabilities at the network's edge. This can be performed by developing MEC-empowered upgrades to the 5G cellular operation, which include machine learning (ML)-based optimization that focuses on predicting the user mobility to achieve service reliability and continuity. The prominent research aspects of the summarization and completion of the integration procedure are service provisioning, NFV and chaining, model-based optimization, data-driven clustering, and estimation, which emphasize the design of methodological tools such as MEC service continuity, mobility, and stability. Accompanying the aforementioned goals, the MEC-empowered localization combining the pre-5G positioning methods and MIMO-based estimation also considered as a major achieving indicator. Finally, regarding the MEC-empowered content-oriented techniques, the objective is to design and assess the performance of developed MEC-based services for content popularity, delivery, caching and users` mobility models, with resource orchestration leveraging reinforcement learning for massive connectivity at the network edge.

In conclusion, being in its early developing phase, the smooth integration of MEC capabilities in the 5G network requires a mutual combination of enormous academic and industrial effort, that applies full multidimensional state of the art research. This can only be done by integration of wide knowledge interchange, deep scientific collaboration, reinforced by large-scale analysis of 5G aspects and beyond. This is the exact essential impulse that motivates the consortium of the "SEMANTIC" project to establish a boundless international training network to impact the scientific aspects of telecommunication research and development society.

Following the incentives set above, this section elaborates on one of the key enabler technologies for 5G and beyond cellular standards – Muti-Access Edge Computing – describing its architecture, its integration with NFV and 5G as defined by European Telecommunications Standards Institute (ETSI) as well as presenting future directions for MEC along with service provisioning options that MEC has the potential to enable.

2.1 ETSI MEC architecture

This subsection describes a generic reference architecture for MEC framework as defined by ETSI summarizing information presented in the corresponding document [1] in terms of its functional elements and reference points. MEC environment can be divided into three levels as indicated on figure 2.1-1 below: system, host and network levels.

MEC entities residing on the two upper layers are described in Sections 2.1.1 and 2.1.2, respectively, whereas Section 2.1.3 defines reference points located between elements underlying in MEC environment. Some examples of MEC services and deployment options are presented in Section 2.1.4.

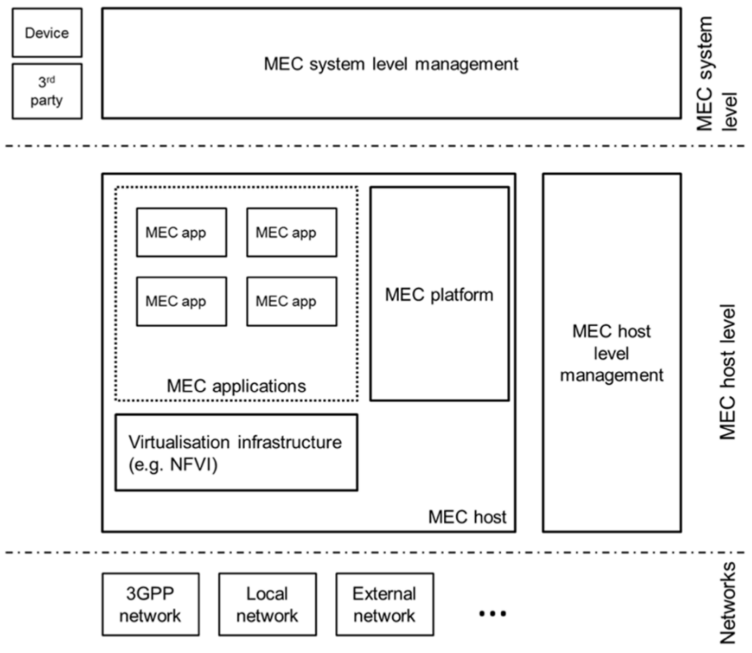


Figure 2.1-1 MEC environment structure[1]

2.1.1 System level elements

System level elements in MEC framework include management entity, device applications and third-party customers.

MEC system level management element comprises of multi-access edge orchestrator, operations support system and user application lifecycle management proxy. *Multi-access edge orchestrator* is the core element of MEC system level management responsible for maintaining MEC system view, selecting appropriate MEC hosts, on-boarding of application packages according to system status, triggering application instantiation, relocation and termination. It is the entity that controls MEC system basing decisions on such information as deployed hosts, available resources and MEC services as well as topology and system constraints. *Operations Support System (OSS)* is operator related element responsible for receiving and granting application instantiation or termination requests from device applications through Customer Facing Service (CFS) portal that allows operators' third-party customers to select and order necessary MEC applications. Once granted, these requests are forwarded to the multi-access edge orchestrator for further processing. *User application lifecycle management proxy* is optional MEC element responsible for authorizing on-boarding, instantiation, relocation and termination requests from device applications. It interacts with MEC orchestrator and OSS for further processing of these requests as well as informs device applications on user application status.

Device applications are applications or software running in the device capable of connecting and interacting with MEC system through user application lifecycle management proxy.

Third party customers span vast range of users that order and receive MEC services through CFS.

2.1.2 Host level elements

Host level elements include MEC host and corresponding management entity.

MEC host is an entity that provides compute, storage, and network resources for MEC applications. It also accommodates MEC platform and a virtualization infrastructure. *The platform* enables MEC applications to discover, advertise, consume and provide MEC services through its environment, as well as handles traffic rules and DNS records received from other MEC elements by instructing data plane and configuring DNS server/proxy respectively. *The virtualization infrastructure* is capable of executing traffic rules dictated by MEC platform as well as routing traffic between different entities such as applications, services, proxy, networks. *MEC applications* are virtual machines running on top of virtualization infrastructure offering and receiving MEC services via MEC platform, with which they mainly interact in the system to perform procedures related to lifecycle, availability etc. Some applications might need to comply with specific rules and requirements set by platform manager depending on the offered service, e.g. lower latency, in order to be validated by MEC system level management entities.

MEC host level management comprises platform and virtualization infrastructure managers. *MEC platform manager* provides element management functions to MEC platform, and mainly handles application related parameters and events such as lifecycle, rules and requirements (e.g., traffic rules, DNS configuration). It also interacts with virtualization infrastructure manager. *Virtualization infrastructure manager* configures and monitors virtualized infrastructure (computing, storage and networking resources) reporting their performance and fault data. It is also capable of performing application relocation to/from external environments when enabled.

2.1.3 Reference points

The components described in Sections 2.1.1 and 2.1.2 as well as the reference points between them are depicted on Figure 2.1-2 with generic multi-access edge system reference architecture. The reference points represent links between system entities and can be categorized into three groups based on the related component: MEC platform, management and external entities.

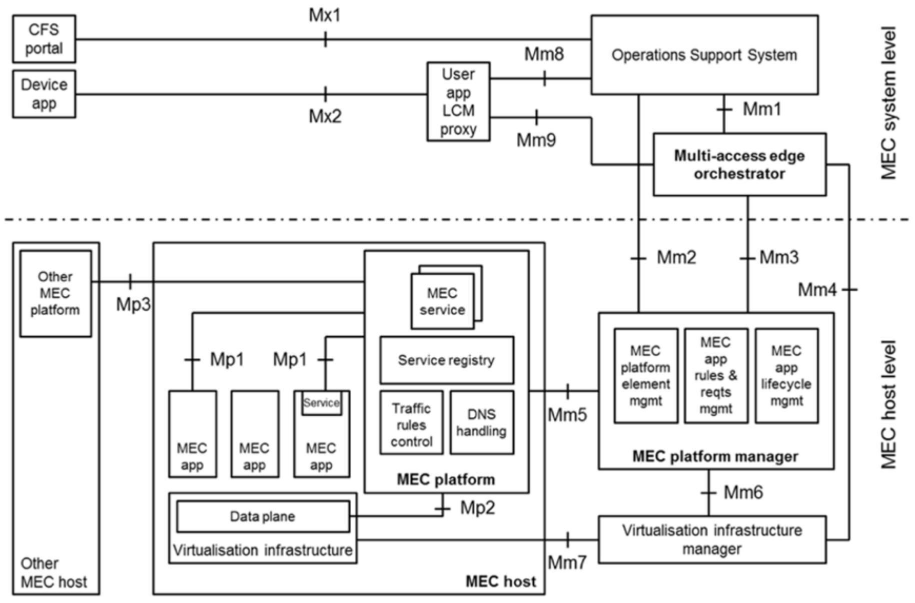


Figure 2.1-2 MEC reference architecture[1]

Reference points related to **MEC platform** functionality denoted as Mp on Figure 2.1-2 are Mp1, Mp2 and Mp3. *Mp1* is the reference point between platform and applications providing their service registration, discovery and communication support as well as other application related communication. *Mp2* connects platform with data plane of virtualization infrastructure and instructs the latter on traffic routing between applications, networks and services. *Mp3* communicates different platforms for exchanging control plane messages.

Management reference points, denoted as Mm, connect MEC system and host level management elements between each other and to other entities and exchange management messages. *Mm1* reference point lies between the multi-access edge orchestrator and the OSS and handles instantiation and termination of MEC applications in the system. *Mm2* connects OSS with MEC platform manager used to configure and manage performance of the latter. *Mm3* between multi-access edge orchestrator and the MEC platform is responsible for exchanging application lifecycle, rules and requirements as well as keeping track of available MEC services. *Mm4* communicates management information about virtualized resources of the MEC host and application images between the multi-access edge orchestrator and the virtualization infrastructure manager. *Mm5* lies between the MEC platform manager and the MEC platform and is used for configuration of platform, application rules and requirements as well as application lifecycle support and relocation procedures. *Mm6* communicates information on virtualized resources between MEC platform manager and virtualization infrastructure manager. Virtualization infrastructure management between virtualization infrastructure manager and virtualization infrastructure goes through *Mm7*. *Mm8* exchanges device applications requests for running applications in the MEC system between OSS and user application lifecycle management proxy. *Mm9* is located between user application lifecycle management proxy and multi-access edge orchestrator and communicates application management messages requested by device application.

Reference points connecting to **external entities** are referred to as Mx and the two external entities refer to customer facing service portal and device application. *Mx1* connects OSS and CFS portal and is used by third-party customers for applications instantiation requests. *Mx2* between user application lifecycle management proxy and device application is utilized by the latter for application related requests.

2.1.4 Services and use cases

MEC service is a service provided and consumed either by the MEC platform or a MEC application. The main idea MEC is to enable authorized third parties or device applications to make use of local services and caching capabilities at the edge of the RAN. Such presence of MEC at the operators' network edge offers a large number of new potential features to be developed in mobile industry.

MEC use cases vary largely and ETSI has identified three main categories for them in [2]: consumer-oriented services; operator and third-party services; network performance and QoE improvements. The deployment options belonging to the first category relate to MEC services that benefit the end-user applications directly, e.g. gaming, remote desktop applications. Operator and third-party services refer to use cases that take advantage of computing and storage facilities close to the edge of the operator's network, e.g. active device location tracking. Lastly, MEC services can be used to improve network performance and QoE thorugh application-specific or generic enhancements, e.g. content/DNS caching, video optimization. The two MEC usage scenarios discussed most frequently in the literature are computation offloading and distributed content delivery and caching [3].

2.2 ETSI MEC architecture and ETSI NFV

Network Functions Virtualization (NFV) and MEC are the critical enabler technologies of next-generation wireless communication. NFV replaces traditional middleboxes having dedicated hardware by their software complements known as Virtual Network Functions (VNFs). As compared to conventional middleboxes, VNFs are easier to deploy, manage, and terminate at lower expenditures. On the other hand, MEC aims to enable cloud computing capabilities at the edge of the access network in close proximity to end-users, thereby offering storage and computational resources at the edge network. With the successful integration of MEC to the mobile network, end-users can experience reduced latency and higher data rates. Moreover, MEC can reduce traffic bottlenecks in the core and backhaul networks by offloading data-intensive tasks towards the edge.

MEC is considered as the key concept to achieve various verticals of 5G and beyond networks, which include supporting enhanced Mobile BroadBand (eMBB), Ultra-Reliable Low-Latency Communication (URLLC), and massive Machine-Type Communication (mMTC). In recent years, several standardization initiatives are being conducted to successfully integrate MEC into mobile networks, among which the most prominent project being ETSI GR MEC 017 [4] that concentrates on deployment of MEC on NFV environment. Realizing MEC in the NFV environment has several benefits, including flexible provisioning of resources, faster deployment of new services, and reduced expenditure costs.

In the NFV framework, decoupling the software processing logic from the hardware adds new capabilities to the communication network and requires the management and orchestration (MANO) of functional blocks handling operation, administration, maintenance, and provisioning as proposed in ETSI GS NFV-MAN 001 [5]. The NFV-MANO framework defines three crucial building blocks of NFV, namely NFV-Orchestrator (NFVO), VNF-Manager (VNFM), and Virtual Infrastructure Manager (VIM). The NFVO is responsible for the lifecycle management of Network Services (NS) and orchestration of NFV-Infrastructure (NFVI) resources across VIMs. The lifecycle management of VNF instances is taken care by VNFM. Each VNF instance is assumed to be associated with a VNFM, and a VNFM may handle a single VNF instance or multiple VNF instances. VIM handles the NFVI resources such as compute, storage, and network. Although a successfully running MEC alongside the existing NFV environment could provide several advantages to the network operators, many challenges need to be addressed.

The ETSI MEC reference architecture in the NFV environment [4] deploys the ME platform, ME applications as VNFs, and the infrastructure resources are realized as NFVI and are managed by the VIM.

ME platform VNFs and ME application VNFs deployment require certain orchestration and lifecycle tasks to be handled by NFVO and VNFM components of NFV-MANO. The Mobile Edge Platform Manager (MEPM) in the reference architecture is responsible for managing the entire application lifecycle of a particular ME host, monitoring various application-level events, and processing status reports and performance measurements from the VIM. In the NFV environment, MEPM of MEC reference architecture is transformed into Mobile Edge Platform Manager - NFV (MEPM-V). The life cycle management of MEC application VNFs previously handled by the MEPM is now taken care of by one or more VNFMs of NFV. The Mobile Edge Orchestrator (MEO) is the core component of MEC system-level management, and it has an overview of the complete MEC system. In the deployment of MEC reference architecture alongside NFV, the MEO component of MEC is transferred into the Mobile Edge Application Orchestrator (MEA-O). The MEA-O depends upon the NFVO for resource orchestration and orchestration of a set of MEC application VNFs as VNF NSs.

In the NFV environment, VNFs can be chained with other VNFs and/or Physical Network Functions (PNFs) to form Network Services (NSs) and concerning the Os-ma-nfvo (between OSS and NFVO) reference point, any interaction with a VNF is associated with at least one NF instance. When ME applications are deployed as VNFs, MEAO would need to know the mapping of ME application VNFs and NSs. This requires possible modifications to the MEAO as well as NFVO. The solutions proposed by the ETSI-MEC deployment in NFV are to request NFVO to create one or more NSs for ME application VNFs and involve NFVO in the creation and termination path of ME application VNFs. That means the orchestration layer has to be involved in every ME application VNF instantiation and termination.

As defined in the ETSI GS NFV-IFA 014 [6], NSs not only involved in chaining VNFs, but they also include a description of the relationship between VNFs and PNFs, Virtual Link (VL) information, VNF Forwarding Graph (VNFFG) containing the topology of NS describing VNFs, PNFs, and VLs that connect them. The ETSI document recommends using the concept of NS to assist the MEC deployment in the NFV environment. In the MEC deployment in the NFV environment, an NS instance could contain a chain of ME application VNFs or ME platform VNFs or both. Reusing the concept of NS in the MEC deployment can also be used to describe the dependency between ME application VNFs and ME platform VNF serving them using Network Service Descriptor (NSD). Furthermore, NS can hold the connectivity information between the ME application VNFs and/or between ME application VNFs and ME platform VNFs using VNFFGs. However, a careful investigation is necessary to know whether the modification is required to NSD and LCM procedures of ETSI NFV to handle specific MEC use cases such as mobility.

As discussed previously, MEAO depends upon NFVO to manage ME application VNFs. To share various information about the ME application VNFs between MEAO and NFVO, a proper communication interface must be set between them. The ETSI standardization [4] defines a new interface named me1 for efficient communication between MEAO and NFVO. The MEAO functional block onboards ME application packages as a special kind of VNF package using the VNF package management interface. MEAO delegates the ME application descriptor (in the form of NSD) into NFVO so that NFV can manage one or more NSs containing ME application VNFs. The MEAO performs instantiation and termination of ME application VNF instances by requesting the NS update procedure of NFVO. In addition to mv1, a new reference point mv2, is also defined to support the management of ME application VNFs. The MEPM functional block of MEC is split into a MEC-specific part (MEPM-V) and a VNFM part responsible for the life cycle management of ME application VNF instances. To allow LCM-related notifications of the ME application VNF to be exchanged between the MEPM-V and VNFM, the mv2 reference point has been introduced. The functionality of the mv2 interface is similar to the ve-vnfm-em interface of ETSI-NFV, but the mv2 interface might not use all the functionalities of the ve-vnfm-vm interface. The VNFM needs to interact with the ME application VNFs to exchange the messages such as initial deployment-related configuration and lifecycle-related events. For this purpose, the mv3 reference point has been introduced, and it connects VNFM and ME application VNFs. The functionalities of the mv3 reference point are similar to the ve-vnfm-vnf of ETSI-NFV.

ETSI-NFV [7] specifies VNF descriptor (VNFD), a deployment template representing a VNF in terms of its deployment and operational behavior requirements. The VNFM uses VNFDs during VNF instantiation and lifecycle management. From the ETSI-MEC [8] side, the application descriptor (AppD) is defined to describe the application requirements and various rules required by the application provider. When the MEC is deployed alongside the NFV, it is worth investigating how these two descriptors work together and whether there is any impact on MEC application VNF onboarding and lifecycle management. There exist several similarities between the AppD and VNFD, such as vendorID, version number, etc. AppD provides a much more detailed description of the infrastructure and service availability than the VNFD. Another notable difference is that AppD can only model a single virtual compute resource to the VNF; however, VNFD can define templates for multiple virtual compute resources per VNF to support scalability. It is also required to analyze how MEC-related data structures/files can be integrated with the VNF package without impacting the existing package contents and how the MEC-related information present in the VNF packages is identified MEC management and orchestration functional blocks.

As specified in ETSI NFV-MANO [5], a VNF instance needs specific functionalities for its lifecycle management, and such functionalities are loaded into the VNF package. ETSI GS NFV-IFA defines onboarding procedures of VNF packages into the NFVO. In the MEC-enabled VNF architecture, VNF packages also contain MEC-specific descriptors. Two different deployment scenarios are possible on how these MEC-specific descriptors are handled. If the master of onboarding procedures is MEAO, VNF packages are first onboarded into MEAO by the OSS, followed by NFVO through the mv1 reference point. On the other hand, if NFVO is the master of the onboarding procedures, the OSS directly gives the VNF packages to NFVO. Via mv1, MEAO would get notifications about the onboarding procedures, and it will instantly fetch the entire VNF package or the part of the VNF package depending upon the requirements.

As described in the MEC architecture [8], a ME host contains a single instance of an NFVI and an instance of the ME platform. When MEC architecture is virtualized, the concept of ME host becomes antiquated. In the conventional MEC architecture, the MEC host is used during ME application instantiation and in mobility, as follows.

Selection of the ME host apriori to application instantiation is necessary to fulfill the constraints of the ME application, such as latency requirements.

To support the movement of UE, the ME host that is currently serving the UE may need to be updated either by launching a new ME application in another ME host, or the current ME application that serves the UE may need to hand over to another ME host.

Even though the concept of ME host has vanished in the NFV deployment, it still becomes the necessity to understand the complement of a ME hist in an NFV based deployment. NFV framework has defined several components to construct NFVI, such as NFVI-PoP and zones that could be considered possible counterparts of ME host in NFV deployment. MEC has a smart relocation feature in which an ME application instance needs to be relocated from one location to another while preserving the current state of the application instance. Although ETSI-ISG NFV [5] has defined VNF migration, the NFV Lifecycle Management interface does not offer any operation to support VNF migration. The integration of VNF migration in the NFV lifecycle management interface requires further research and standardization efforts.

The process of ME application instantiation includes two parts; the first part involves the deployment of ME application VNF started by creating required virtualized resources and initial configuration of ME application VNF, while the second part triggered by MEPM-V involves sending the configuration information to the ME platform. The process of ME application instantiation will be successful only if both the parts are successful. The configuration includes the traffic rules, DNS rules, etc. There are two possible solutions described to handle the ME application instantiation procedure. In the first solution, MEAO relies on the NFVO to deploy the ME application VNF. MEAO then waits for the NS change notification to decide whether the ME application VNF was created successfully or not. Once the ME application VNF is created successfully, the MEAO requests MEPM-V to send the necessary configuration parameters. In the second solution. In the second solution, MEPM-V acts as EM and subscribes to the VNF lifecycle change notification sent by the VNFM. The deployment of ME application VNF is performed by NFVO. After the successful deployment, MEPM-V will be notified. This notification triggers the MEPM-V to obtain the configuration from MEAO and send the configuration to the ME platform.

The ME application VNF termination procedure has two parts; the first part removes the configuration from the ME platform and data plane, while the second part terminates the ME application VNF. There exist two possible solutions to perform the VNF termination procedure. In the first solution, MEAO sends the ME application VNF instance termination request to the MEPM-V. If the graceful VNF termination is requested, MEPM-V informs the ME platform to give time for the application level termination of the ME application and wait until successful or timeout. Later, the MEPM-V requests the ME platform to remove the configuration. After receiving the termination response, MEAO requests NFVO to terminate the resources given to the ME application VNF, which in turn requests VNFM to shut down the ME application VNF and release the resources. In the second proposed solution, MEPM-V acts as an EM and is subscribed to life cycle management events generated by VNFM. The MEAO calls NFVO to terminate the ME application VNF. The NFVO requests VNFM to shut down the ME application VNF and release the resources. The NFVM then notifies the MEPM-V about the event of the VNF shutdown. Upon receiving the notification, MEPM-V removes the configuration information from the ME platform and data plane.

To conclude, this section has provided some insight into the architecture that defines how to deploy the MEC architecture in the NFV environment and run both the complemented technologies on the same NFVI successfully. Several encountered challenges during the integration of MEC and NFV are explored in this section, along with possible solutions to handle them.

2.3 ETSI MEC architecture and integration with 5G

2.3.1 Motivation

The four main functions of 5th Generation (5G) are communication, computation, control and content delivery. 5G is not only an advancement in mobile broadband (eMBB) services but it also extends to support vast diversity of massive Machine Type Communication (mMTC) and Ultra-Reliable Low Latency Communication (URLLC) based application services along with support for different verticals. It is foreseen that each application in 5G has different requirements in terms of latency, data rates, reliability/availability, mobility and transmit powers, inducing signification surge in demand for computation and storage resources closer to the users along with enhanced communication resources. Early 3GPP 5G releases alone are not enough for achieving such advancements. The edge computing power of MEC and its integration with 5G brings applications, 3rd party services and content closer to the user in efficient way and is one of the main enablers for turning telecommunication networks into versatile service platforms [9].

2.3.2 Key enablers for MEC integration with 5G [10]

- **5G Service-Based Architecture (SBA)**: In 5G core network, the 4th Generation (4G) monolithic Evolved Packet Core (EPC) is disaggregated to implement each function in such a way that it could run independently on a Common Off-The Shelf (COTS) server hardware. It allows 5G core function to be flexible and decentralized.

- **Application Function** has the ability to influence User Plane Function (UPF) (re)selection and traffic routing directly via the Policy Control Function (PCF) or indirectly via the Network Exposure Function (NEF), depending on the operator's policies
- **Local Routing and Traffic Steering:** The 5G Core Network selects a UPF close to the UE and executes the traffic steering from the UPF to the local Data Network via a N6 interface. This may be based on the UE's subscription data, UE location, the information from Application Function (AF)
- **Session and Service Continuity (SSC)** modes for different UE and application mobility scenarios
- **Local Area Data Network (LADN)** support.

2.3.3 MEC integration with 5G [10]

The 5G system architecture specified by 3GPP and shown in Figure 2.3-1, has been designed with significant changes in RAN and core part to handle wide range of use cases from a massive amount of simple IoT devices to the other extreme of high bit rate, high reliability mission critical services. The user plane and control plane network functions are worth introducing before diving in to MEC integration with 5G.

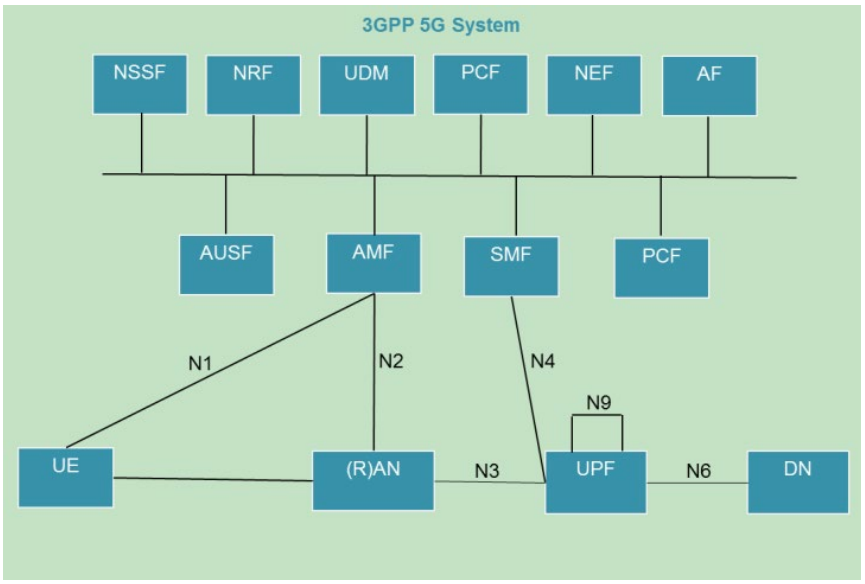


Figure 2.3-1: 5G Service-Based Architecture[10]

- **Network Resource Function (NRF)**: Contains the list of network functions and services they produce.
- **Network Exposure Function (NEF)**: Responsible for service exposure and authorization of service access requests.
- **Application Function (AF)**: Application influence on traffic routing, accessing NEF and interaction with Policy Control Function.
- **Authentication Server Function (AUSF)**: Authentication server
- **Network Slice Selection Function (NSSF)**: Responsible for network slice instances selection and allocating necessary AMF for users.
- **Access and Mobility Management Function (AMF) and Session Management Function (SMF):** Handle mobility related procedures, authentication and authorization for the access layer, security anchor functionality and responsible for the termination of RAN control plane and Non-Access Stratum (NAS) procedures, protecting the integrity of signalling, management of registrations, connections and reachability.

- **Policy Control Function (PCF):** Handles policies and rules of 5G system. It can be accessed directly or via NEF depending on whether the AF is considered trusted or not.
- **Unified Data Management (UDM):** Generates Authentication and Key Agreement (AKA) credentials and responsible for user identification handling, access authorization and subscription management.
- **User Plane Function (UPF):** Distributed and configurable data plane that supports packet routing & forwarding, packet inspection, Quality of Support (QoS) handling, acts as external Protocol Data Unit (PDU) session point of interconnect to Data Network (DN), and is an anchor point for intra- & inter-RAT mobility.

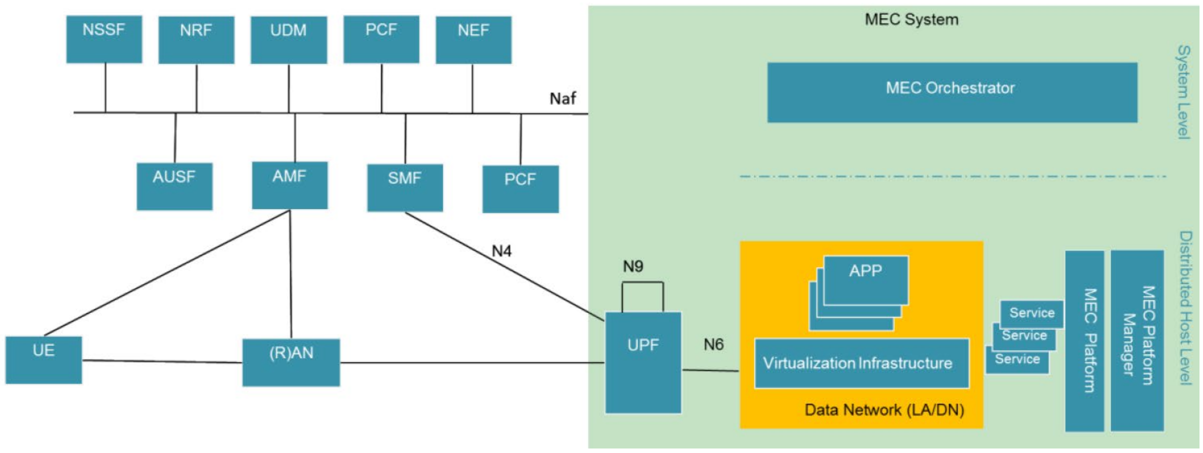


Figure 2.3-2: Integrated MEC deployment in 5G network[10]

As shown in Figure 2.3-2, UPF has a major role in MEC integration as it is a distributed and configurable data plane and, in some deployment, locally it can be a part of MEC implementation. MEC orchestrator acts as an AF and interacts with network functions either directly or via NEF. NEF can be deployed centrally together with similar network functions or an instance of NEF can also be deployed in the edge to allow low latency, high throughput service access from a MEC host. MEC host are most often deployed in edge or central data network. UPF steers the user plane traffic towards targeted MEC.

The distributed MEC host accommodates MEC apps, a message broker as a MEC platform service, and another MEC platform service to steer traffic to local accelerators. The choice to run a service as a MEC app or as a platform service is likely to be an implementation choice.

2.3.4 MEC deployment [10]

MEC can be flexibly deployed in different locations depending on requirements. It could be near the Base Station or near the central Data Network. The 4 standard examples are shown in Figure 2.3-3.

1. MEC and the local UPF collocated with the Base Station.
2. MEC collocated with a transmission node, possibly with a local UPF.
3. MEC and the local UPF collocated with a network aggregation point.
4. MEC collocated with the core network functions.

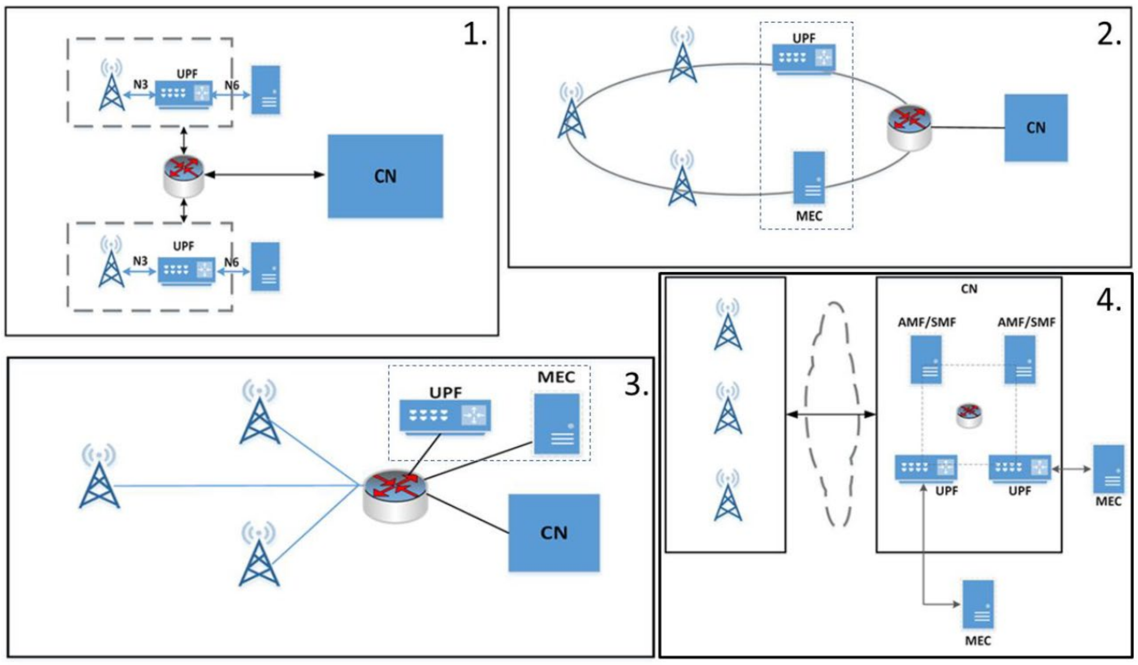


Figure 2.3-3: Examples of the physical deployment of MEC[10]

2.3.5 Open-RAN [11]

2.3.5.1 Motivation

The traditional wireless network infrastructure makes it difficult for data centers and service providers to tackle massive connections with different services requirements and provide reliable and efficient service to the users. The 5G network helps in solving the above problem with the help of Virtualization, MEC and Network Slicing.

The dependency of RAN part more on hardware and the vendor lock in is causing huge CAPEX and OPEX cost and restricting in building a intelligent cooperative reliable network. Hence, it is important to build 2G, 3G, 4G and 5G RAN solutions based on a general-purpose, vendor-neutral hardware and software-defined technology. Virtualization and RAN disaggregation in 5G helped in building such technology called Open-RAN(O-RAN) whose main principles are openness and intelligence.

2.3.5.2 Architecture:

The main components of O-RAN are:

- **O-RU, O-DU and O-CU** whose functionalities are similar to that in 5G disaggregated RAN except with added support of O-RAN based specifications and interface;
- **Near-Real Time RAN Intelligence Controller (RIC)** for control/optimization of RAN elements and resources based on fine grained data using online AI/ML. It is suitable for application with latency needs of between 10ms and 1s;
- **Non-Real Time RIC** for Control/optimization of RAN elements and resources based on coarse grained data using online AI/ML. It is suitable for application with latency requirement greater than 1s. It also provides policy-based guidance to near-Real Time RIC.

RIC components can be placed either in edge or core network depending on the usage scenario.

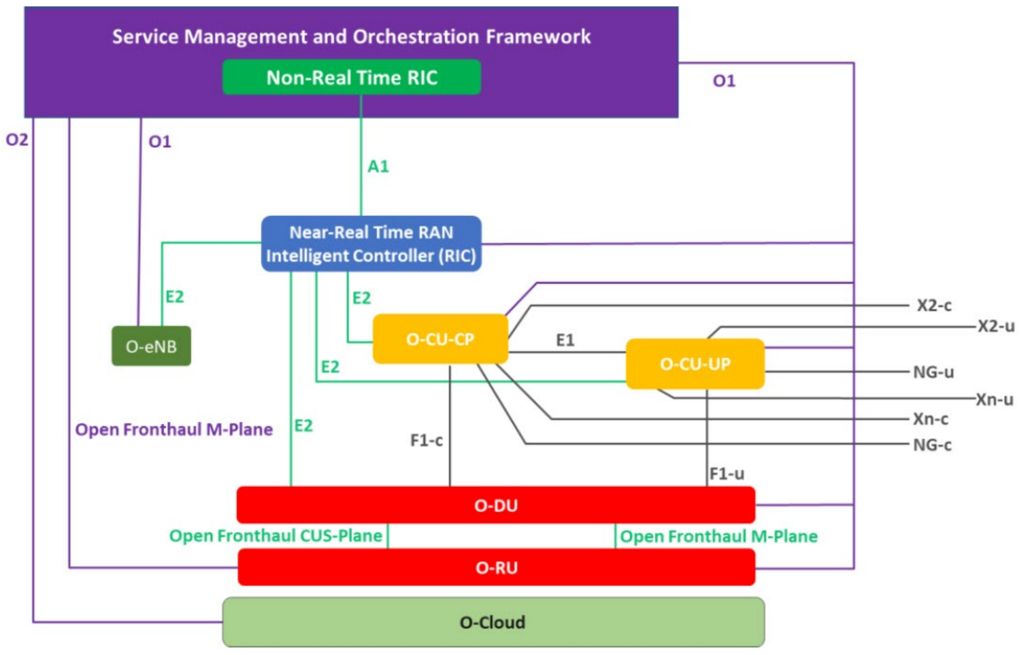


Figure 2.3-4: O-RAN Architecture [11]

2.4 Future directions for MEC and MEC service provisioning

Mobile Cloud Computing (MCC) facilitates capabilities to the end-user by providing them storage, computation, and energy resources offered by the centralized cloud by integrating cloud computing and mobile computing. In MCC, as services are provided to the end-users from the remote centralized cloud, end-users and mobile devices experience high latency, low coverage, low data rate, etc., which has become difficult to tackle in the next-generation mobile networks. In the MEC paradigm, services are deployed on the edge servers in close proximity to mobile devices and end-users to provide ultra-low latency and/or keep data locally. MEC can solve the challenges associated with MCC by offering context-aware services to the end-users with low end-to-end latency, high throughput, and availability of storage, computation capacities.

MEC has emerged as a critical enabling technology for realizing various use cases of next-generation mobile networks. One of the crucial issues in MEC is providing support for smart mobility. The movement of an arbitrary end user from one location to another could significantly increase the end-to-end latency, dramatic degradation in the QoS, and even cause service disruption. Therefore, to guarantee the service continuity of end-users, two solutions are possible as proposed in the ETSI GR MEC 017 [4]. The first solution creates a new service at the edge server near the end-user. In contrast, the second solution considers the service migration from one edge server to another following the movement of the end-user. Although creating a new service seems to be the better solution, it further requires the older service to transmit state perceived information about the end-user to the new service. Moreover, frequent migration of the end-user from one edge server to another requires frequent instantiation of new services, resulting in the wastage of scarce resources at the edge servers. Therefore, to guarantee the service continuity of the dynamic end-users, it is important to consider service migration. In addition to this, service migration should preserve the intermediate states of computation, making it a stateful service migration. That means, once the service is migrated to another edge server, the service should resume from the point where it had stopped previously. As MEC is deployed in the NFV framework by running the ME platform as VNFs and

ME applications as VNFs, service migration boils down to solving stateful migrations of VNFs between the edge servers.

The problem of VNF migrations is extensively studied in the literature from the NFV point of view. VNF migration in the MEC context is challenging because i) we need to select a new edge server for hosting the VNF in close proximity to the end-user having sufficient resources to serve the migrated VNF ii) Migrating the states between the VNFs is itself challenging even in the NFV context. iii) how and when to carry out the migration process. The last challenge needs particular attention as it deals with the tradeoff between the migration cost and the QoS improvement experienced by the end-user after migration. The recent works on live VNF migration concentrate on reducing the downtime period experienced by the end-users during the migration process. However, in the MEC context, one must consider reducing the total migration time as the end-user could experience deteriorating QoS continuously after moving to the new location. The effect on the destination server after VNF migration is one of the essential aspects to consider. The destination edge server might already have many VNFs running and accommodating one more VNF should not impact already running VNFs on it. The proposed solutions for VNF live migration benefit from shared resources in the data center servers for storing the checkpoints and/or state information, etc. However, in the MEC context, as we have scarce resources at the edge nodes, the benefits of shared resources are negligible.

For deploying the VNFs on the edge servers, network operators can either use Virtual Machines (VM) or containers. Migrating VMs and containers require different methodologies, and their migration cost could be different. As VMs provide proper isolation of resources from the host machine, they need to transmit considerable data to the destination edge server during migration. As compared to VMs, containers are lightweight, and they share most of the resources with the host machine; during migration, they need to send very little data. Considering the limited bandwidth, lesser storage, and low computational power at the edge servers, migrating the containers would be much easier than migrating VMs. Container-based VNF migration is a relatively new domain compared to VM live migration and hence demands further investigations.

To get the maximum benefit from NFV and MEC, two complementing and enabling technologies of next-generation mobile network, MEC is deployed in the NFV framework. As described in ETSI, the MEC platform and MEC application run as VNFs on top of shared infrastructure. As the edge servers have limited computational, storage, and network resources, to optimize the resources at the edge, it is a common strategy to consolidate multiple VNFs on top of commodity hardware. To provide performance similar to middleboxes, today's NFV frameworks leverage several optimization techniques such as Data Plane Development Kit (DPDK), Direct Data I/O (DDIO), etc. Still, it is challenging to achieve the predictable performance of VNFs as they contend for various resources of the underlying host server. Performance degradation caused by VNFs in the presence of other VNFs on shared hardware is termed performance interference.

Current VNF placement efforts place the VNFs on dedicated cores to provide isolation from other co-located VNFs. However, these strategies overlooked the fact that even though VNFs have their dedicated cores to run, they still compete for the underlying resources such as Last Level Cache (LLC), memory, and I/O controller, network resources, etc. As a result, co-located VNFs may experience performance interference, and hence their performance may degrade. In the MEC context, edge servers are equipped with fewer number of resources, and VNF placement should consolidate multiple VNFs on the same edge server to increase resource utilization. However, consolidating multiple VNFs at the edge could also result in performance interference. Hence VNF placement at the edge server must wisely utilize the resources while taking into account performance interference. In other words, there is a need for interference-aware

VNF placement at the edge network, which tries to optimize the resource utilization without violating the Service Level Agreements.

There are two directions to solve the interference-aware VNF placement at the edge network at the higher level. The first one is relying on partition-based VNF placement to avoid interference, i.e., considering the availability of multiple resources such as core, LLC cache, memory bandwidth, etc., while making the placement decisions. While the second direction is to develop a method to predict the performance of a VNF on a destination server considering other co-located VNFs and, based on the prediction, wisely choosing the edge server for deployment in which performance degradation is minimal. To facilitate the partitioning of the resources among co-located VNFs, one could use the tools such as Cache Allocation Technology (CAT) proposed by Intel Resource Directory technology (RDT). However, for predicting the performance of a target VNF in the presence of other co-located VNFs, various machine learning techniques need to be explored.

3 Background on Technologies Relevant to the Project

This section presents a brief introduction to technologies relevant to WP 2 on MEC/RAN integration and MEC-empowered enhancements providing their definitions, target KPIs, advantages and disadvantages along with some examples.

3.1 MEC service placement and function chaining

3.1.1 Virtual Machine and Containers

When MEC services are deployed on the NFV framework, they are either deployed as Virtual Machines (VMs) and containers (e.g., docker, lxc, etc.). Although VMs and containers both provide virtualized and isolated platforms on top of host machines, such as commercial off-the-shelf servers, they differ in interaction with the host machine resources. Depending on which virtualization technology we use, the performance of MEC services varies, and it is imperative to understand the difference between these two virtualization technologies.

VMs are primarily designed to run the software on top of physical hardware to emulate a particular hardware system. VMs provide complete isolation from the host machine by running their own operating system, including their own kernel. A hypervisor or a virtual machine monitoring system runs in between the host machine resources and the VMs. It manages all the VMs running on top of the host machine while providing an abstraction layer to the VMs; VMs are unaware of the available resources at the host. Despite their advantages, VMs are considered expensive in terms of the host resources they consume. Each VM not only runs an entire operating system but also maintains a virtual copy of the hardware that the operating system needs to run. A single VM instance will have a size in GBs, making it harder to migrate between the host machines.

In modern cloud computing capabilities, the containers are considered lightweight alternatives to VMs. Containers run on top of the physical server and operating system, sharing the host OS kernel and optional binaries and libraries; however, the shared components are read-only. The containers are lightweight, generally, a few MBs size, as they have a lot of dependency on the host machine. But, as compared to VMs, containers do not provide much isolation.

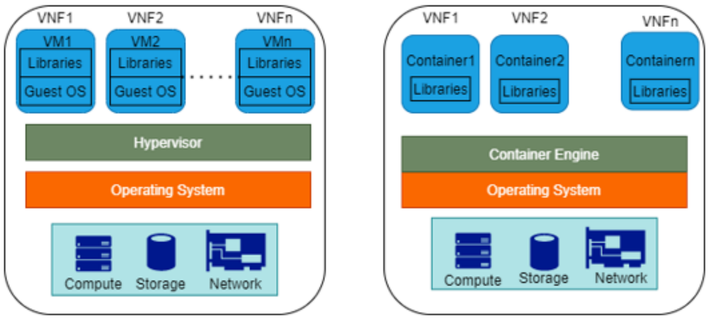


Figure 3.1-1 Virtual Machines and Containers

Figure 3.1-1 clearly distinguishes between the containers and virtual machines. Depending upon the use cases of the application, the operators need to decide whether to deploy their applications on VMs and containers.

3.1.2 Intel Resource Directory Technology (RDT)

One of the critical challenges in the widespread deployment of NFV is getting predictable and deterministic performances in terms of network throughput and end-to-end latency. As 5G uses the concept of NFV for providing support for various use cases, it is particularly important to get predictable performances. Multiple VNFs running on top of a host machine either as VMs or containers share the host resources in an unregulated manner, resulting in performance degradation. In the NFV context, one such resource that demands a controlling knob is the processor's LLC, which is shared by all the cores present in that processor. The VNF running in one core consuming a lot of LLC cache space can adversely degrade other VNFs' performance running on other cores. Towards this end, Intel has proposed Cache Allocation Technology (CAT) for partitioning the LLC and Memory Bandwidth Allocation (MBA) for throttling the MB. Other chip manufacturers like Qualcomm introduced similar mechanisms for resource partitioning.

CAT enables partitioning of the LLC and allocates the LLC partitions to specific cores of the node through software-programmable control over LLC. CAT introduces 16 Class of Services (CLOS) to achieve cache partitioning, and a core or a set of cores can be associated with a specific CLOS. The available cache ways can be allocated to each CLOS using a Capacity Bit Mask (CBM) dynamically, which indicates how much cache can be used by them. The values of CBMs specify the relative amount of LLC space available for a specific CLOS and defines the degree of overlapping or isolation. A CPU core has to be associated with a CLOS, and a CLOS can be associated with multiple CPU cores defining disjoint or overlapping cache partitions, respectively.

The MBA technique facilitates approximate per core control over the MB between L2 cache and LLC using a new programmable bandwidth controller. Like CAT, CLOS is used here too for throttling the MB for different cores. MBA levels in terms of percentage can be assigned to the required CLOS dynamically up to 90% in steps of 10%. However, upper-bound and granularity of MBA are machine-dependent.

An efficient VNF placement algorithm presented in [12], explores the usage of CAT for the placement of VNFs. It can also be used in the VNF migration problem.

3.1.3 Problems with Linux Networking Stack and Data Plane Development Kit (DPDK)

The commodity servers used in the NFV environment can cause performance penalty in terms of throughput and latency [5]. This is one of the primary reasons why service providers still prefer middleboxes over NFV. The major bottlenecks caused by the NFV environment are as follows.

1. Interrupt Based Packet Processing: The commodity servers still follow the traditional Linux packet processing, which is interrupt-driven i.e., whenever a packet arrives into the NIC, an interrupt is generated to notify the kernel about the packet arrival event. In the fast packet processing environment where NIC receives millions of packets per second, it causes increased interrupt activity resulting in low performance. Recent work [13] further clarifies the adverse effect of interrupt-based packet processing in the container environment.
2. Extra packet copy overhead: The kernel in Linux packet processing reads the packets from the NIC into the kernel space and later copies it into the userspace. The extra packet copy between the kernel space and the user space incurs additional overhead.
3. Complex Linux network stack: Packets arrived at the kernel space are processed by the Linux network stack, which has several expensive functions in terms of CPU cycles.

These concerns, as mentioned above, can be addressed using kernel bypass techniques in which packet arrived at the NIC are directly copied into the user space bypassing the kernel completely and thus avoiding the interrupts, extra packet copy, and Linux network stack. DPDK is one such networking framework incorporating kernel bypassing and is considered as a one of key enabling technologies for NFV. Several existing NFV platforms have been successfully built on top of DPDK to get the benefits of kernel bypassing. However, DPDK has a poll mode continuously listening to the NIC for incoming packets that makes the CPU resources unnecessarily busy, which leads to the wastage of resources. Furthermore, DPDK is heavily dependent upon Intel hardware such as specific type of NICs making it harder to deploy in cloud-centric environment.

3.2 MEC clustering

The techniques discussed in Section 3.1 are valid here too.

3.3 MEC service provisioning

Amazon Wavelength: To support MEC applications, Amazon has come up with an optimized solution known as AWS Wavelength. AWS wavelength enables the developers to build MEC applications that could serve the end-users with ultra-low latency. AWS wavelength deploys the standard AWS compute and storage devices at the edge of the telecommunication carrier's 5G network. One such example is the integration of AWS wavelength and Verizon's 5G core to form 5GEdge. With AWS wavelength, it is possible to extend the Amazon Virtual Private Cloud (VPC) to one or more wavelength zones and run the applications on top of Amazon Elastic Compute Cloud (EPC) instances to ensure ultra-low latency services.

Following are the few key concepts involved in Amazon wavelength.

1. **Wavelength zone**: A wavelength zone is present in the 5G carrier network's location, and in the zone area, wavelength infrastructures are deployed. Wavelength zones are associated with the AWS region and are considered as the logical extensions of the AWS region controlled by the control plane of the region.
2. **VPC**: A customer's VPC spans over availability zones, local zones, and wavelength zones and runs the Amazon EC2 instances in the subnets associated with the zones.
3. **Subnet**: It is possible to create one or more subnets in the wavelength zone and run the EC2 instances in those subnets.
4. **Carrier gateway**: A carrier gateway is responsible for allowing incoming traffic from the carrier network to a specific location and allowing the outgoing traffic to the carrier network and the internet.
5. **Network border group**: A specific location in the wavelength zone from AWS advertises IP addresses.

As of now, AWS wavelength is available in ten cities across the US with Verizon, Tokyo, and Osaka, Japan with KDDI, in Daejeon, South Korea with SKT. Although AWS wavelength provides a MEC platform for 5G carriers, it is dedicated to businesses and is not freely available and is not open-source.

Nokia AirFrame*:* The Nokia AirFrame open edge cloud infrastructure is another solution that addresses the requirements of the network edge. The AirFrame open edge server is the most important building block of the Nokia AirFrame infrastructure, and it is an x86 solution developed explicitly for edge and far-edge cloud deployments. As the AirFrame open edge server chassis is only 3U high, it is easier to deploy on existing base station sites. A single open-edge chassis supports up to five servers, each having a scalable single Intel Xeon processor. By leveraging Nokia ReefShark capabilities, the open edge server can implement acceleration capabilities for CloudRAN, machine learning/artifiial intelligence, and other workloads.

The Nokia AirFrame open edge cloud infrastructure provides an ultra-small footprint that can be complemented with a real-time optimized, OPNFV compatible cloud infrastructure with OpenStack distribution built to run in small datacentres while delivering performance and ultra-low latency required by the applications. It's the first edge-optimized cloud infrastructure with carrier-grade high availability and versatile scalability, ranging from single-server edge cloud to multi-rack solutions. It allows for easy deployment of the solution, as well as remote update automation and configuration management.

3.4 MEC mobility and service area formation

Provision of uninterrupted services to a frequently on-the-move client is a challenge in MEC. Users move across various geographical locations and at different speed if they are in vehicle and their services are to be migrated to follow them so that the benefits of MEC are maintained. The main challenge here is when and where to migrate the service. Migrating a service may incur service interruption, network overhead and if MEC has some capacity limit it might disturb other running services, whereas not migrating a service may increase the data transmission delay between the user and the edge that hosts its service when the user moves away from its original location. It is challenging to make an optimal migration decision because of the uncertainty in user mobility as well the complex trade-off between the costs related to migration and distant data transmission. This opens different problem areas like,

- MEC service migration.
- Optimal/dynamic choice of services supported by MEC located in a particular area depending on its capacity limit.
- User mobility aware task assignment to reduce end-to-end delay of MEC task.

3.5 MEC-enabled services based on AI

The aim of service providers has always been to reduce network cost and have efficient content control. RAN is considered as one of the most expensive sections of network to which the user first connects to. With the increase in network capacity and traffic, mobile networks have become more complex to control. Intelligent network control with the help of Open RAN would help service providers to use available resources efficiently without overloading the network and provide best quality of service to the users, but it is a great challenge. The two main problems focused here are:

- **Traffic steering**
  It is an evolution of mobile load balancing and is mostly used network solution to achieve optimal traffic distribution based on different categories like service type, location, interference, QOS, load balancing, optimization of network/user performance etc. Artificial Intelligence/Machine Learning (AI/ML) based traffic steering algorithm help in customization of user centric strategies and proactive optimization by predicting network condition.
- **Resource management**
  RAN resources in some areas may not be enough to fulfil the requirements for all users using the same service so dynamic intelligent decision for resource management must be made like slicing the network or QOS based resource management so that at least certain prioritized users would have satisfactory level of QOS etc.

3.6 Optimal access point selection techniques in cell-free scenario

It is known that in a cell-free scenario multiple Access Points (APs) simultaneously serve many users in the same frequency resource. Optimal selection of APs based on user location, channel quality and interference would help improving sum spectral efficiency and maintain balanced load

3.7 Edge content caching

To fulfill the ever-increasing resourceful demand and follow the technological advances – the study of efficient techniques to enable efficient data transmission has led to the introduction of a novel concept of edge-content caching. This concept holds a simple but promising key idea – bringing popular content closer to the end users by exploiting storage resources available in their near proximity. The content caching model can reduce the number of content requests addressed to the origin servers and lower the traffic load entrusted to these content servers. This can heavily reduce the traffic congestion during the peak-traffic transmission times, significantly decrease the delivery latency, and as a consequence provide a better Quality of Experience (QoE) to the network clients.

Generally, the term *content caching* is defined as a performance optimization algorithm that provides an efficient solution for data delivery from the closest content servers to the end-users for optimal performance. Typically the content caching problem can be divided into two major sublayers of caching – higher *Layer 1 Cache* – which is based on caching at the wired content routers, and lower *Layer 2 Cache*, where the content is cached closer to the users – at the base stations and proxies. The collective caching model of two hierarchical layers: *Layer 1* and *Layer 2* is shown in Figure 3.7-1. Each of the mentioned layers can be described with a set of corresponding subproblems of – *what to cache*, *where to cache*, *dimensioning of caching*, and *delivery [14]*.

The representative end-to-end path between the content server (CS) that holds original content and the end-user typically consists of the original CS itself, router, the Macro base station (MBS), Femto base station (FBS), and user equipment (UE). The set of popular content can be stored and served from the edge servers that are located much closer to the end-users than the CS marked with *A*. The caching facility can be implemented in a variety of different spots in the current paradigm – core routers with the storage – that are marked with *B, C, D,* and *E*, MBS – marked as *F*, and FBS – marked as *H*. Additionally, content caching can be also introduced to the UE devices in the case of the device-to-device (D2D) communication, shown by *I*. Considering the case of cloud radio access network (CRAN) – content caching can also be potentially realized at the baseband unit (BBU), that are clustered into the BBU pool (marked as *J*), and the remote radio heads (RRH) (*K)* that are attached to the BS`s antennas.

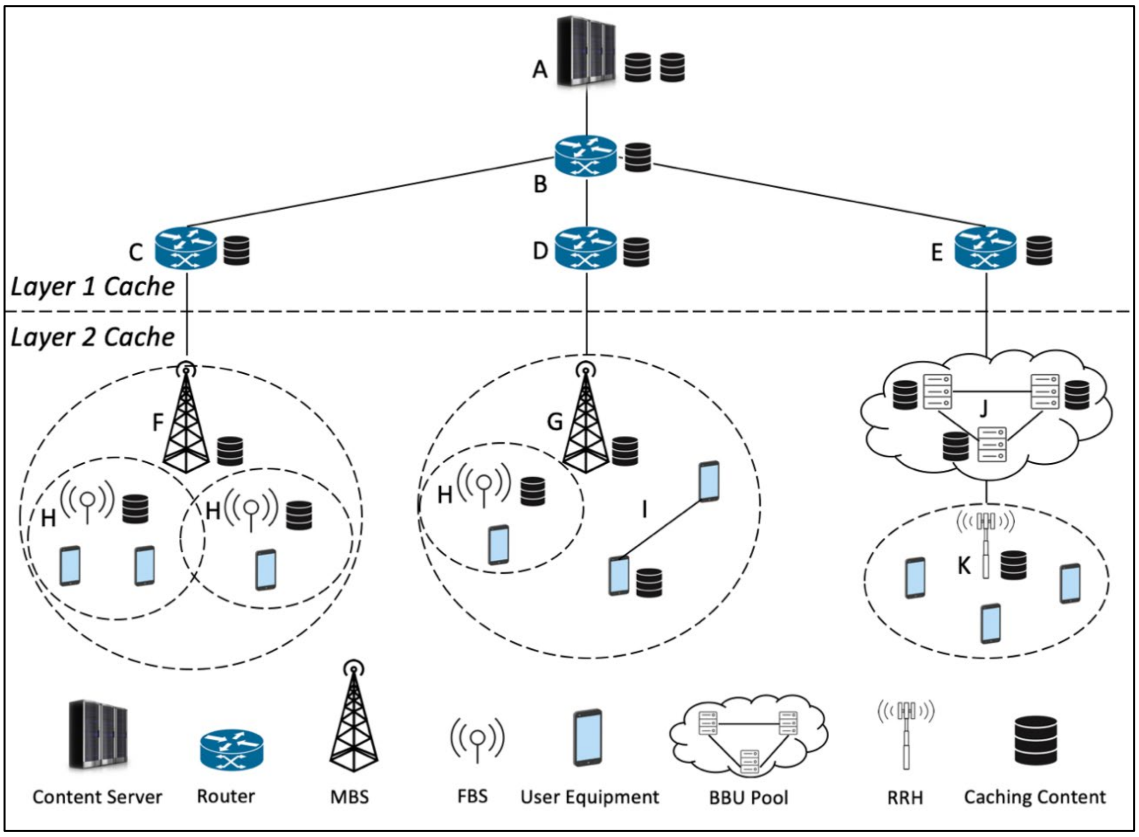


Figure 3.7-1 MEC-aware caching paradigm

The higher *Layer 1 Cache* can accommodate the content at the network routers, that along with a basic router functioning are also accompanied by storage. Important to notice, that usually, the routers in the network are connected by wired links.

The *where to cache* subproblem of the *Layer 1* targets the challenge of which edge routers to choose for optimal caching. It can be seen that the copies of content can be placed on router marked *B* as well as at the lower hierarchical routers *C, D,* and *E* in the network topology. This implementation idea can significantly reduce the cost of transmission from the original content servers to the users connected to the particular router and reduce the latency. The cost of caching at the edge routers can be measured in terms of storage occupation or power consumption metrics.

The subproblem of *what to cache* is focused on choosing an appropriate set of content to be cached to maximize the cache hit ratio (CHR). The optimal set selection can be based on the patterns of past users` requests or advanced content popularity estimation algorithms that will be discussed in Section 3.9. An additional ingredient in the *what to cache* subproblem is an appropriate cache eviction design. While the list of popular content may significantly change within an hourly period, the cache eviction has to be intelligently implemented to avoid storage space overflow. Some of the very well-studied cache eviction policies include: least frequently used (LFU), first in first out (FIFO), least recently used (LRU), or randomized replacement [14].

The *dimensioning of caching* targets the problem of the amount of storage allocation at the edge routers. The underlying concept here is to implement dynamic and flexible management of the optimum storage to be available for caching. One would suggest allocating as much storing space as possible to have flexibility in potential content accommodation, however, excessive storage may lead to the overconsumption of power resources for sustaining storage function.

The *delivery* subproblem takes into consideration the transferring function of the cached content to the clients. There are two prevailing traffic types: file downloading and video streaming. Considering the file downloading – the major metric that is taken into account is the delivery time – as the file cannot be accessed before being fully downloaded. As for the video streaming, the delivery process is much complicated if, for example, the video file is divided into chunks – the time scheduling of each particular content piece becomes challenging and the downloading time only cannot reflect the overall QoE of the clients anymore.

Typically the content caching implementation at the *Layer 2 Cache* is much more challenging, as it is performed based on wireless links. Therefore, in most cases, the wireless concept requires radical changes in the caching strategy, due to highly stochastic wireless channels, very limited storage and power resources, and rapid changes in users` mobility.

In the *where to cache* problem at the *Layer 2 Cache* the main target now is to find the optimal places for content to be cached – either MBSs, FBSs, BBU pools, RRHs, or a combination of them. Implementing content caching at MBS and FBS can significantly reduce the congestion of the backhaul links, however, the storage space is usually very limited and not enough for caching all available popular content. Here, the optimization caching techniques come into play, which will be discussed in Section 4.3. Additionally, if the client holds some parts of the content, they can be cached for later use by other user equipment that requests the same content. In the case of the cloud-based network architecture, the use of caching can be also introduced to the BBU and RRHs to reduce the user-perceived latency and congestion, however, it can also increase the signaling overhead and weakens the processing capability.

As it was introduced in *what to cache* for *Layer 1 Cache*– the decision and eviction procedures are mostly depend on the accurate content popularity estimation. However, unlike the routers in *Layer 1 Cache* the MBSs and FBSs are of different coverage sizes and hence must have flexible popularity estimation strategies focused on the particular coverage area, spatial and temporal dynamics of user preference.

In the case of Layer 2 Cache caching, the *cache dimensioning* now has to take into consideration the backhaul link status and characteristics of the stochastic wireless channel. To decrease the backhaul traffic occupation, guarantee strong wireless QoS and high QoE in the content network – the caching memory size has to be intelligently optimized for implementation case.

Content delivery in the *Layer 2 Cache* can be described by three delivering subproblems defined by the number of transmitters and receivers: content delivery from one cache holder to one UE, content delivery from one cache holder to multiple UE, and coordinated content delivery from multiple cache holders to multiple numbers of UE. When the popular content is cached at the BS, it then can be requested and transmitted to other clients connected to the same BS. One of the key objectives of the content delivery subproblem here is to provide dynamic scheduling, that can be scaled up and down depending on the number of transmitters and receivers. The additional objective is to improve spectral efficiency by sharing the channel state information and the content among different BSs or UEs.

3.8 Adaptive video streaming techniques

According to Cisco the share of video traffic in the total mobile data traffic is expected to reach 79% in 2022[10]. At the same time, users’ expectations for video quality have greatly increased due to the widespread penetration of high-speed network accesses. The study of efficient techniques to enable an effective video transmission with a focus to provide a better QoE to the network clients has led to the introduction of a novel concept of Dynamic Adaptive Streaming over HTTP (DASH) approach for online video delivering.

The concept of DASH was first presented in 2007 by Move Networks [15] [16]. Being an active field of research with growing interest from industrial players such as Netflix and YouTube, DASH has become an essential technique for enhanced video delivery systems beyond 5G cellular networks. Such adaptation method allows the network clients to select which video quality to request from the server for each segment, relying its decision on the varying network conditions. Additionally, if the player implements an intelligent adaptation logic, it can prevent the video buffer from emptying out and consequently stopping the playback.

The main idea of the DASH – is dividing a media file into fragments of the same time duration. The term *fragment* in this domain is used along with *chunk* or *segment* with the same meaning. Each video segment is then encoded into different video rates that can be then downloaded depending on the end user`s devices or network conditions. The set of representations can be placed on one or many different servers along with the Media Description File (MDP) which contains the description of the encoded segments, such as video rates used per segment, duration of a full video file, and duration of each segment in seconds.

Another essential feature of the DASH algorithm is its compatibility with Content Distribution Networks (CDN) or technologies of proxy servers that are used by standard HTTP facilities. Being transparent to the end-users, this compatibility allows the content from one provider to be cached on a third-party CDN infrastructure for the reason of saving cost and decreasing the overall latency. The client interaction only involves sending the HTTP GET request to request the segments, while the rate adaptation procedure is usually performed by an appropriately assigned intelligent logic.

The DASH concept relies on two major parts of the model – media representation and adaptive bitrate selection (ABR). The media representation part involves the set of available bitrates into which the video segments can be encoded. It has been industrially standardized and therefore there is no high research interest is innovation here. On the other hand, the ABR part involves the decision-making logic that defines the selection of the bitrate according to the particular channel parameters. This is the novelty part where the competitive intelligent adaptation procedure can be introduced to improve the users` QoE according to specific metrics.

The classical definition of the ABR algorithm can be divided into three key subcomponents, where each component of this concept is assigned by a particular functioning:

- *Resource estimation*
- *Segment request scheduling*
- *Bitrate adaptation*

This way of drawing the concept of ABR provides highly flexible management and scaling of the adopted models. Important to notice that these components can be separated and located on different systems – client, edge, or content server. However, because content provides an aim to target a huge number of clients, most of the works adopt the implementation concept of the client-side logic. Additionally, the parameters that play a key role in the ABR decision-making process, such as currently experienced bandwidth, power level, playing device parameters, and buffer occupancy, can be observed and estimated exactly at the user device – what makes the intelligent ABR logic be often located at the UE side.

The *resource estimation* module represents the function of collecting the system`s resources that are available now, which can include channel parameters, throughout estimation and power level of the device. The *segment request scheduling* takes as input the estimated buffer occupation and is responsible for the schedule when the next segment is going to be downloaded. This module includes two major types of scheduling – sequential, where the segments are requested one at a time, and parallel, where segments can perform multiple segments downloading at the same time. The *bitrate adaptation module* takes into account the input from the segment request scheduling module and the resource estimation module and is responsible for decision making of which bitrate resolution is going to be downloaded next. Being the key functioning element, the adaptation module has involved into different types of adaptation logics present in literature – heuristic-based, control theory-based, optimization techniques, machine learning-based. The majority of them will be discussed in Section 4.3. The ABR framework is shown in Figure 3.8-1.

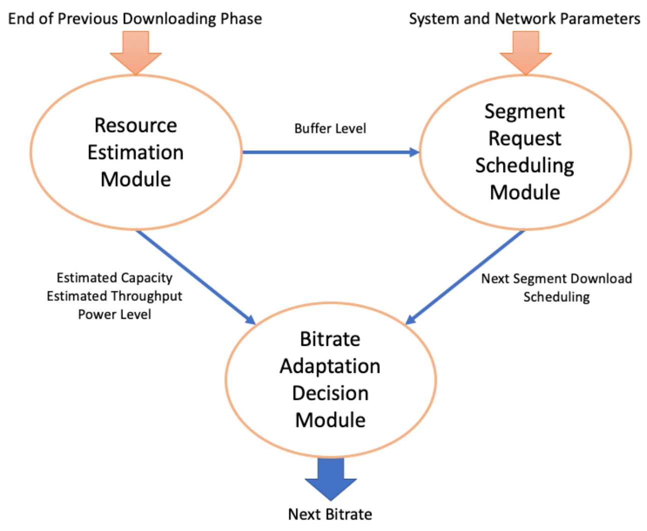

End of Previous Downloading Phase
System and Network Parameters
Resource Estimation Module
Buffer Level
Segment Request Scheduling Module
Estimated Capacity
Estimated Throughput
Power Level
Next Segment Download Scheduling
Bitrate Adaptation Decision Module
Next Bitrate

Figure 3.8-1 Integral ABR framework.

One of the key blocks that define the adaptation logic algorithm is a *resource estimation* module. Understanding the measurement and monitoring capabilities of the module and the effect of estimated resources is halfway to building an intelligent decision-making logic. Three benchmarking resource estimation scopes are currently presented in literature and efficiently adopted by industry:

- *Throughput-based*
- *Buffer-based*
- *Power level-based*

The throughput estimation-based algorithms rely on a simple idea of selecting the highest video bitrate exactly lower than the previously estimated systems throughout. This implementing way can establish high assurance of avoiding the rebuffering events as well as providing high-quality video levels. However, the throughout itself cannot provide an efficient estimation of highly varying network condition and design an efficient ABR selection model. Another resource estimation algorithm is buffer-based. The estimator module monitors the buffer occupancy as its main source and passes the measurements to further adopt the decision based on the buffer level. Whilst power-based estimation algorithm is focused on adaptation the ABR logic based on the power level device, rather than on the throughput or buffer. For example, when the charge of the user device is low, but longer watching time is preferable – the logic may downgrade the video quality level. Finally, the current estimation. techniques may rely on a combined resource measurement and include the collected information about the through, buffer level as well as power level parameters [17].

Another important factor in implementing efficient bitrate adaptation is the metrics that are used to evaluate the performance. The key problem that the introduction of DASH algorithms aims to solve is a direct enhancement of users` quality of experience (QoE). The purpose of the QoE enhancement is to improve the perceived user experience and to establish an enhanced long-term engagement between user and content provider. To provide a flexible and comprehensive evaluation of the proposed techniques, the model must consider different QoE assessing metrics. Although it is hard to estimate the QoE functions for each user particularly, the key parameters in the evaluation models can be identified as:

- *Average Video Quality*: Average quality over all chunks:
$$\frac{1}{F}\sum_{f=1}^{F} q(R_f),$$
- *Average Quality Variations*: Evaluation is performed by monitoring the magnitude of the changes in the quality from one chunk to another:
$$\frac{1}{F-1}\sum_{f=1}^{F-1}\left|q(R_{f+1})-q(R_f)\right|,$$
- *Rebuffering Events*: For each video chunk f rebuffering event occurs if the download time t_f is higher than the playout buffer level when the chunk downloading started. The total rebuffering time can be find as:
$$\sum_{f=1}^{F}\left(\frac{d_f(R_f)}{C_f}-B_f\right),$$
- *Startup Delay: T_s* where $T_s \ll B_{max}$ – maximum threshold buffer occupancy.

Based on the previous parts of the evaluation metrics, the QoE of a video segment f_1 through F can be found as a weighted sum of the components:

$$QoE_1^F = \sum_{f=1}^{F} q(R_f) - \lambda \sum_{f=1}^{F-1}\left|q(R_{f+1})-q(R_f)\right| - \sum_{f=1}^{F}\left(\frac{d_f(R_f)}{C_f}-B_f\right) - \mu_s T_s,$$

where λ, μ, μ_s are non-negative weighting parameters corresponding to video quality variations, rebuffering time and startup delay, respectively [18].

3.9 Content Popularity estimation techniques

This includes implementation of dense small-cell base stations, introduction of novel multiple-input, multiple-output (MIMO) techniques, utilization of millimeter waves, and applying advanced application layer capabilities. One of such recent improvements towards efficient video delivery is the estimation of video content popularity. Popularity estimation techniques have been introduced as an essential basis for efficient content caching at the edge. That is, an accurate estimation of the list of popular content allows intelligent caching to increase the average cache hit ratio (CHR) and hence improve the user`s QoE. Additionally, the intelligent estimation of popular content helps to tackle the resource utilization problems, as with increasing volumes of video content produced and shared – more storage, computational and delivery resources are required.

The main focus of service providers is to keep up with the increasing demand and at the same time put most of their effort into high QoE provisioning. Based on the estimated demand and popularity of a specific list of video content, providers can place their media files at the strategic edge locations, and hence this popularity list has to be as close to optimal as possible. The most influencing factors that typically have a strong impact on content popularity are reviews, ratings, advertising, recommendations, rankings, and discounted pricing. However, due to the remarkable variability of the content popularity features, some of these factors are hard to be practically evaluated.

Typically, the simplest way to create a list of popular content – is based on the number of previously collected requests. The requests for each particular media file are counted from a given group of users under consideration and then stored in a list. Finally, the popularity estimation task is to predict the number of future requests for each video file in the list. One of the challenging aspects here is the fact that the popularity of the content is highly dynamic and time-varying. Additionally, the observed request patterns from users are typically not uniform and therefore, the patterns have to be monitored continuously. The request-based popularity estimation technique is a solid method as this approach can rapidly adjust to fresh requests of new video files [19].

The content popularity estimation research domain is rich in a variety of baseline methods, that have been shown efficient in implementation. Here, we will describe the fundamental popularity estimation methods that rely on the requests records patterns and eviction policy-based renewal. Typical representatives of such estimators are – Least Frequency Used (LFU), Least Recency Used (LRU), Least Recently/Frequently Used (LRFU), and Exponential Weighted Moving Average (EWMA).

The *LFU algorithm* is responsible for identifying the least popular objects in the popularity list, by implementing a cache replacement policy of erasing the objects from the memory that have been requested the least number of times in the past. This algorithm is usually implemented at the edge cache servers by utilizing the requested frequency pattern to pinpoint the least popular files, meaning that the video content with the lowest number of requests most likely will be evicted from the cache. The main drawback of the LFU algorithm is capturing only the long-term popularity pattern, while slowly reflect on the new ones For instance, the piece of content that is no longer relevant to users, but has the high requests number from the past will be kept in the top rank of the popularity list [19].

The *LRU method* has an underlying idea of time distance dependence. The LRU identifies the content object that is requested least recently – by defining the time window from the last request to the current time. This object then will be erased from the popularity list of the cache server. However, the frequency of request and the long-term popularity of objects are not considered by default, and hence the LRU can react faster to the rapid changes in objects` popularity, but slower to the popular objects in the past. As an example, implementation of LRU may result in the eviction of popular video content while keeping the high rank of the videos that had a recent short-term jump in requests.

When the popularity of the content is relatively stable, the LFU is considered an effective algorithm, however, on the opposite occasion, the LRU may perform better. Typically, the content providers prefer the short-term popularity to be identified faster and then quickly respond by caching that content at the edge.

The key idea of the *LRFU algorithm* is to combine the two previously described – LFU and LRU methods. The LRFU was found to perform better than its two predecessors, by evaluating the popularity list by using a so-called – combined recency and frequency (CRF) value. The CRF assigns the value of the likelihood that particular piece of content will be requested soon. Consequently, when the proxy needs to update the popularity list – the objects with the lowest CRF values will be evicted.

The *EWMA approach* implements the idea of calculating the averaged access interval of each piece of content. The past pattern in *EWMA* is exponentially forgotten over the time and the update of the average request time interval of the media file is updated by:

$$\tau_{interval} = (1-p) * \tau_{interval} + p * (t - t_{last}),$$

where $\tau_{interval}$ is the average access interval of a file, t_{last} is the time of that files` recent request, $p \in (0,1]$ is a forgetting rate, t as a current time, and $(t - t_{last})$ represents the latest requested interval. It is important to notice that, when the p is equal to 1, the approach turns to LRU.

3.10 Blockchain technology for network service provisioning

Blockchain that is mostly known as the technology underlying the infamous Bitcoin cryptocurrency is a specific database where the data is stored in blocks that are chained together. Its main idea leverages on peer-to-peer(P2P) network architecture such that information is managed by all of its participants in a decentralized manner. This database can store any kind of data; therefore, its applications span various industries from supply chains to public sector with the most widespread application being ledger for economic transactions. Some of its emerging usage scenarios include IoT, smart city and future networks.

In order to better understand this technology and its potential or utility for network service provisioning, the next subsections 3.10.1 and 3.10.2 will define its main elements and describe key characteristics, respectively, following the definitions provided in [20], whereas blockchain scalability trilemma and possible solutions are explained in 3.10.3 based on the discussion presented in [21].

3.10.1 Blockchain elements

The main elements of blockchain include transaction, data block, distributed ledger and consensus. Additionally, definitions of smart contract, public and private blockchains will be presented.

Transaction is a record of exchanging data or some value between blockchain participants.

Transactions are stored in **data blocks** together with blockchain header and other important information required to maintain, preserve and extend the database. Each block is linked to its preceding block through a hash label such that all blocks in the chain can be traced back. Also, once the block is sealed, no modifications to block data are possible.

The entire blockchain is never stored in a central location; instead, it is spread and replicated among participants of a peer-to-peer network. Whenever a new block is chained to the database, every peer updates its blockchain to reflect the change, hence it is also referred to as **distributed ledger.** In other words, the process of recording transaction is translated into data exchange among peers.

Lack of central management and regulation entity requires blockchain participants to establish trust and reach agreement about transaction history on their own without jeopardizing reliability. Such trust and agreement in blockchain are achieved through validation protocol called **consensus**. It identifies the authorized parties and validates transactions to be inserted into the database. Different blockchain based cryptocurrencies rely on different consensus algorithms, e.g. Bitcoin adopts a Proof-of-Work (PoW) algorithm that requires consensus nodes to collectively solve complex mathematical tasks to identify a hash function output required to seal blocks[22]. Due to its high power consumption, alternative algorithms such as Proof-of-Stake (PoS), Proof-of-Authority (PoA), Byzantine Faulty Tolerant (BFT) and others have been proposed and deployed [23].

In addition to consensus mechanism, some recent blockchains systems are equipped with **smart contracts** (SC) that are programmable self-executing scripts running on top of blockchain network to further improve its automatism. They are capable of performing credible transactions without requiring the involvement of third-party intermediaries thus allowing more efficient and cheaper operation. Ethereum [24] is an open-source public blockchain platform with a full implementation of the smart contract concept.

Distributed ledgers can also be classified according to the degree of accessibility into **public** and **private** systems. Public blockchains are permission-less meaning that anyone can access it, e.g., Bitcoin and Ethereum, whereas private ones require invitation and are more centralized, e.g., Hyperledger Fabric [25].

3.10.2 Key characteristics

Combination of the elements listed in Section 3.10.1 makes blockchain immutable, decentralized, transparent and secure database.

Immutability is the property of being able to keep transaction history unchangeable over time. When the transactions are verified, they are timestamped and inserted into a block linked to a previous block through a hashing process. Therefore, neither sequence of blocks not data inside those blocks cannot be altered making transaction history serialized and crystallized over time. As a result, this property allows secure data storage.

Decentralization refers to the distributed nature of transactions management and storage through consensus mechanism bringing such advantages as tamper resistance and increased reliability due to absence of single point of failure.

All the information about transactions recorded in blockchain is visible to blockchain peers because it is spread across the network for public verifiability such that all participants are equally allowed to access, verify and track transactions. Such **transparency** further improves blockchain integrity and data immutability yet preserving openness and equity.

Blockchain provides high degree of **security** thanks to the usage of cryptographic tools for transaction verification and member authorization such that blockchain records are protected against potential attacks. Despite the transparency of data, the blockchain systems are capable of sustaining user **privacy** by enabling data owners to control information disclosure through SCs.

Above properties make blockchain a promising technology for 5G and beyond network service provisioning, yet **scalability** issue of blockchain serves as the main obstacle towards proliferation of blockchain in this area. Scalability refers to the ability to process transactions mainly measured in terms of number of validated *transactions per second* or *transaction confirmation latency*. Both of these performance metrics have not reached satisfactory levels and are extensively researched [21].

3.10.3 Blockchain scalability trilemma

Blockchain suffers from a scalability trilemma – a trade-off between decentralization, security and scalability degree, where at most only two can be satisfied. For example, larger number of peers would be in line with higher tamper-resistance and decentralization but will result in a longer block formation period, therefore higher transaction confirmation time. On the other hand, introduction of centralized coordinator might improve transaction throughput by reducing time needed for reaching consensus among peers but will serve as a single point of failure jeopardizing security. Similarly, increasing block size might result in higher transaction capacity as it could fit more transactions but will compromise decentralization and fairness degree in the long term by prioritizing the entities with larger processing capacity. Balancing these three aspects is an essential **challenge** to be addressed for integration of blockchain technology into future cellular networks and require finding scalability solutions that are both well decentralized and secure.

Existing scalability **solutions** can be classified as on-chain and off-chain referring to Layer 1 and Layer 2 of blockchain respectively. The Layer 1 solutions focus on the blocks' structure, consensus algorithm and the structure of the chain, whereas Layer 2 solutions focus on offloading on-chain transactions and related computational tasks to off-chain platforms thus reducing main-chain transactions number [21].

One such solution is a **payment channel** (PC) that is a temporary off-chain trading channel capable of processing transactions between two peers that previously agreed on-chain to lock fixed amount of funds to the payee. Therefore, multiple off-chain payments can be performed incrementally (in terms of balance update) during channel's lifetime while requiring sporadic on-chain interactions for instantiation, modification, update and termination of the channel.

3.11 Mobile data access model beyond 5G

Generally, mobile data refers to the internet service delivered to mobile devices such as smartphones or tablet over a wireless connection, while mobile data access designates the underlying procedures required to establish, maintain and terminate that mobile internet connection. Section 3.11.1 briefly summarizes the status quo of mobile data access in pre-5G networks motivating the need for new designs, followed by an overview of a future mobile data access model by [26] in Section 3.11.2.

3.11.1 Traditional mobile data access model

Currently, mobile users access internet mainly through cellular network access points (APs) or wireless local area networks (WLAN). In the first case, the user needs to have entry credentials such as subscriber ID, IMSI, keys, that are usually recorded on a SIM card provided by mobile network operator (MNO). Mobile data consumption is governed by the prescribed data plans or pay-as-you-go deals agreed and paid beforehand to a home MNO. In the second scenario, the user either needs credentials for accessing private APs, or immediately buy short-term access coupons to access public APs. Only in few cases, user can get the wireless internet access for free and most likely under condition of allowing to process their personal data. In a nutshell, mobile data access strongly depends on the availability of AP (in terms of coverage) and access rights of the user.

However, in view of extremely heterogeneous beyond 5G mobile networks spanning vastly diverse vertical (applications), horizontal (tiers) and service (MNOs, CPs, OTT) domains, such data access model is considered rather rigid and inefficient in meeting relentlessly increasing demand for data and connectivity, hence necessitating a paradigm shift towards distributed, self-managed yet cooperative network access control. These requirements can be potentially resolved by using blockchain technology running on top of such cellular network and managing some of its features.

3.11.2 Future mobile data access model

The main idea of the data access model for 5G and beyond networks proposed in [26] is to allow on-the-fly mobile data access not bounded by a-priori service license agreements. Instead, the clients will be free to choose among available access points (in coverage) which they want to connect to and pay for the consumed services using blockchain backed payment system in the application layer. It should be noted that other stages of network access such as service discovery, pairing, negotiation, parametrization and management will still be governed by standard network-level protocols with some apt modifications. Moreover, they will also be able to share and trade their network assets to other entities, e.g., local cached content, available internet connection, acting as servers. In the scope of the following discussion, client refers to the consumer of service, whereas server is the one providing mobile network assets.

During **service discovery and pairing**, the clients broadcast their service requirements (content and QoE) and shall receive offers (including pricing) from interested servers. The client chooses a server based on offered service cost, RAT parameters, QoE KPIs, server reputation and other parameters, formed and exchanged during **service negotiation and parametrization** phase. Once client-server pairing is performed, both peers carry out network-level procedures to establish, maintain and terminate the mobile service exchange referred to as **service management**. Similarly to service discovery, service management is user-driven, meaning that client is responsible for informing and handling any changes related to consumed service. At the same time, the two parties should perform **service charging** procedures at blockchain level following previously negotiated payment timeplan. The charging mechanism depends on the blockchain characteristics such as consensus.

4 State-of-the-art Solutions and KPIs

This section provides a literature review of state-of-the-art solutions relevant to the scope of research in which ESRs are involved. Their individual contributions are presented in separate subsections named according to the corresponding research area. Each state-of-the-art solution or study is discussed in terms of their main idea, system model, methodology, performance metrics, key findings and evaluation (advantages/disadvantages analysis) in separate subsubsections. The considered works are then compared in the last subsubsection leading to future research directions based on their comparative analysis.

4.1 Machine learning-based optimizations for seamless MEC service support

4.1.1 Contention-Aware Performance Prediction for Virtualized Network Functions [27]

Main Idea: The primary cause of resource contention in the NFV framework is the memory subsystem. As multiple VNFs are consolidated on the same server, they contend for memory subsystem resources; thus, observe performance drop relative to when they run alone. Because of the contention-induced performance drop in the NFV enabled servers, it is hard to ensure the Service Level Agreements (SLAs). Thus, NFV orchestrators need an efficient mechanism for performance prediction. Recent works on NFV performance prediction function poorly because they treat memory as a monolithic unit and use linear models to correlate a system metric such as cache utilization by the competitor and the observed performance drop of the target VNF. In contrast, this work memory subsystem has been identified with three different checkpoints, and the contention can happen simultaneously and independently across these checkpoints: Main memory access bandwidth, packet I/O subsystem that delivers the packets to Last Level Cache (LLC), evictions from the LLC. The contention-aware model built considers all these checkpoints.

The contention-aware performance prediction model is based on i) contentiousness that captures the pressure an NF places on shared hardware and ii) sensitivity which shows how susceptible the target VNF in the presence of aggregate contentiousness of competitors. For successfully measuring the contentiousness, a carefully selected set of hardware counters is used. The hardware counters measure the resource utilization at CPU-socket and server-level granularities, which are sufficient to quantify the contentiousness. The sensitivity of each VNF can be considered a non-linear and non-continuous function of a multivariable contentiousness vector, which can be carefully modelled using ensemble techniques from the machine learning literature. Based on the contentiousness vector and the sensitivity model for each VNF, SLOMO, a novel performance prediction framework is designed. SLOMO accurately predicts the throughput with an average end-to-end prediction error less than or equal to 8%.

System Model and Methodology: The workflow of SLOMO consists of two conceptual, logical parts i) an offline profiler responsible for profiling the contentiousness vector of each VNF and modelling the sensitivity ii) a prediction module that predicts the performance of a target VNF in the presence of other VNFs. SLOMO has three key components to realize the logical parts: contentiousness characterization, sensitivity modelling, and contentiousness composition. These components are designed by following the data-driven approach.

The contentiousness vector of a VNF is measured by running it on the commodity server along with synthetic workloads and measuring various system-level and CPU-socket level hardware counters. A set of synthetic workloads is carefully chosen so that each workload applies incremental pressure on various dimensions of the memory subsystem, namely LLC, DDIO cache, etc. For each synthetic workload, a single contentiousness vector is generated. Later, the contentiousness vectors are grouped together based on how many co-runners (utilized cores) were simulated.

The most challenging task is the estimation of contentiousness composition. For example, suppose there are three VNFs running, NF_A, NF_B, and NF_C. Now the network operators want to know the performance of NF_A in the presence of NF_B and NF_C. Although operators know the pre-computed contentiousness vectors of NF_B and NF_C, operators do not know $V_{B,C}$ - the aggregate contentiousness generated by NF_B and NF_C. For measuring the aggregate contentiousness, operators need to consider the contentiousness vector of NF_B and NF_C in the presence of two competitors. These contentiousness vectors can be obtained from the contentiousness vector creation phase. The composition vector $V_{B,C}$ can be obtained by applying the simple linear operation such as addition or average on each metric of individual contentiousness vectors.

The most important phase of SLOMO is the modeling sensitivity phase. The sensitivity modeling can be viewed as a regression problem, which takes the contentiousness vectors of the competitors as input and predicts the throughput of target VNF as output. The sensitivity models are trained using synthetic, NF-specific contentiousness metrics and the observed performance of the target VNF in the presence of synthetic workload. During the prediction phase, the contentiousness vector of synthetic workloads is replaced by the contentiousness vector of real competitors of the target VNF. It has been observed that the sensitivity cannot be modeled using simple regression techniques as the sensitivity displays a non-linear and non-continuous correlation with the contentiousness vector. Thus, sensitivity is modeled as the piecewise function of its inputs, and for that purpose, gradient boosting regression is used.

Performance Evaluation: For evaluating the accuracy of SLOMO approach, absolute mean prediction error of all experiments across different NF types is considered. The results shows that SLOMO could predict the performance of a target VNF in the presence of interference with a mean prediction error of 5.4% which is about 64% improvement over prior prediction techniques. In addition to the mean prediction error, it is important to understand does SLOMO under/overestimates the performance. To see if SLOMO under-predicts or over-predicts the performance, signed prediction error across various NF types is considered. Compared to previous prediction techniques, SLOMO produces much better predictions. Furthermore, the prediction error of various machine learning algorithms are compared inorder to show why the choice of gradient boosting regression was the best decision. The results indicate that gradient boosting regression was the only technique that produced the prediction error less than 10%.

Advantages and Disadvantages: SLOMO efficiently predicts the performance of target VNF with prediction error less than 10%. SLOMO can also be easily incorporated into existing systems and it easily adapts to new NFV systems. However, inorder to apply SLOMO, it requires to each VNF in the presence of various synthetic workloads which is a very expensive and time consuming procedure.

4.1.2 ResQ: Enabling SLOs in Network Function Virtualization [12]

Main Idea: When multiple VNFs are consolidating on the same server, providing performance guarantees is harder. Current NFV solutions place VNFs on isolated cores and believe that it is sufficient to achieve performance SLOs. However, even though VNFs run on their dedicated cores, they still share resources such as the last-level cache, memory, and I/O controllers known as uncore resources. In NFV, contending for uncore resources results in performance degradation of as much as 40%.

In the NFV context, the primary source of performance degradation arises from contention for the LLC. For solving the LLC contention problem, Intel has introduced a new feature known as Cache Allocation Technology (CAT) which provides hardware-based mechanisms for partitioning the LLC cache across cores. Usage of CAT to provide performance isolation among competing VNFs has not been widely tested and applied in NFV infrastructure. This paper focuses on whether or how operators can use CAT to provide performance isolation among competing VNFs in such as way that performance SLOs are satisfied. Through extensive experiments, it has been found that CAT alone is not sufficient to provide performance isolation among the competing VNFs. Careful investigation on this problem shows that to provide proper performance isolation among the competing VNFs, a simple buffer sizing policy in addition to cache partitioning is required.

The paper later investigates how to apply the CAT and simple buffer policies in a practical NFV context. For applying CAT to facilitate performance isolation, there is a need for a scheduler that assigns resources to NFs so that resource allocation is accurate, efficient, and scalable. Towards this end, ResQ, a cluster resource scheduler is designed that provides performance guarantees for NFV workloads. Firstly, ResQ computes the number of NF instances required to satisfy SLOs and allocates LLC and cores to those NF instances. To identify how much LLC is needed for each NF to meet the SLOs, ResQ depends upon an offline profiling phase. The profiling phase understands how the performance of a VNF varies as a result of LLC allocation. For scalability, ResQ tries to minimize the number of servers required to admit the NFs to guarantee the SLOs using a greedy approach.

Methodology: For investigating whether CAT is sufficient to guarantee performance isolation in the NFV environment, a few NF workloads from the research community and industry are considered, such as Efficuts, EndRE, snort, suricata, vEPC, etc. All the experiments are run on a server equipped with Intel Xeon E5-2695 v4 server having 10Gbps and 40Gbps network ports. For each NF, two scenarios are considered:

1. Solo run: In this scenario, NF under test runs alone in the server with CAT configured to allocate 5% of LLC to the NF along with a dedicated core, and the performance of target NF is noted. This scenario provides the baseline results and highlights how the NF behaves when allocated with a dedicated LLC slice, without contention for other resources.
2. Shared run: As in the previous scenario, NF under test gets 5% of LLC and its own dedicated core. In this scenario, 11 instances of different competing VNFs run on the remaining cores using 95% of LLC.

The result of the above experiments shows that CAT is insufficient to provide performance isolation, particularly when the packet size is larger (1518 B). Further careful investigation of this problem indicates that CAT and careful packet buffer sizing policies are required to achieve performance isolation.

System Model: ResQ is a cluster resource manager designed to schedule NFs without compromising performance SLOs efficiently. ResQ has three logical aspects to scheduling VNFs successfully.

1. Service Interfaces: Conventionally, network operators achieve performance SLOs by overprovisioning the resources. It is possible to achieve the performance SLOs while utilizing the resources efficiently. Given a fixed set of resources, the performance of an NF depends upon two factors. Firstly the NF configuration, such as the ruleset of a firewall., secondly, the traffic pattern. ResQ considers these aspects of VNF while making resource allocation decisions. The operators need to provide the details of NF in terms of its configuration, traffic profile, and performance target, etc., to ResQ as SLOs. Operators can specify how kinds of performance SLOs: reserved SLOs, and on-demand SLOs. The reserved SLOs are more suitable for VNFs having static workloads that do not change much over time. However, on-demand SLOs are suitable for NFs with highly variable workloads.
2. ResQ profiler: For a given NF having reserved SLO, ResQ needs to determine its resource requirements which are based on NF configuration, traffic pattern, and the platform on which NF is running. ResQ profiler measures how NF performance varies as a function of LLC allocation. Building a general NF profile is particularly challenging because it requires us to explore unbounded space and is considered highly infeasible. ResQ relies on interpolation to quickly create the NF profiles. The output of the ResQ profiler is given as input to the ResQ scheduler.
3. ResQ scheduler: ResQ scheduler works at two levels. ResQ scheduler takes NFs, SLO requirements, and profiling information as input. The scheduler determines the number of NF instances required

to be launched to ensure performance SLOs, the server(s) on which the NF instances must be placed, and the amount of LLC to assign to each instance. The centralized scheduler is responsible for admission control, placement of all SLOs, and allocating resources for the reserved SLOs. During this process, the centralized scheduler also sets the resources apart for on-demand SLOs. The centralized scheduler runs an online greedy algorithm for faster admission. Each server runs a server agent responsible for configuring the server and monitoring resource utilization and SLO violations. The server agent has a local scheduler for allocating resources to NFs having on-demand SLOs.

Performance Evaluation: To show how intelligently ResQ schedules the competing VNFs, three sets of reservation-based SLOs are generated, each having 200 terms. The first set contains only cache-insensitive NFs, the second set contains only cache-sensitive NFs, and the third set contains the mix of both. Each SLO has target throughput and latency in the range of 90% to 100% of what a single instance of the NF could achieve while running alone. The performance of the cluster-based NF scheduling methodology proposed by ResQ is evaluated by considering the accuracy and efficiency.

1. Accuracy: To show how accurately ResQ allocates resources, the accuracy of ResQ in terms of SLO violations is compared with a simple contention agnostic scheduler and an equal LLC partitioning method. In the experiment, each NF runs alone without restricting the cache access to measure the throughput and latency. Then in the simple contention agnostic scheduler, multiple NFs are packed together on the single server. For cache-insensitive applications, no SLO violations were observed. However, for cache-sensitive application, an SLO violation of 91% has been recorded, and in case of the mixed set, 11% of SLO violations has been observed. Later, to verify whether a naive cache isolation strategy (equal LLC partitioning) is sufficient to reduce the violations or not, the same VNFs are consolidated by allocating equal partitions of LLC to each one of them. This kind of scheduling policy resulted in much higher SLO violations; in the combination workload, 55% of the SLOs are violated. For cache-sensitive workloads, 100% of the SLOs are violated. The schedules determined by ResQ have no violations in all the cases.
2. Efficiency: The efficiency of the ResQ mechanism is captured by determining the number of machines required to place the SLOs in such as way that there are no SLO violations. The efficiency of ResQ has been compared with two states of art solutions: i) online scheduling, which scales the NF instances in the event of SLO violations. ii) Prediction-based scheduling uses performance predictors to measure the performance of target VNF and uses this information for scheduling. Prediction-based scheduling performs very poorly as compared to ResQ and online scheduling.

4.1.3 Iron: Isolating Network-based CPU in Container Environment [13]

Main Idea: Containers are used to deploy the computation in the cloud data centers. Compared to other virtualization platforms such as VMs, providing resource isolation in the container environment is challenging because they share the components of the underlying operating system. Resource isolation is considered an important aspect for both cloud providers and application developers. To provide resource isolation, cloud providers need to allocate resources to the containers in a fine-grained manner. Overprovisioning or underprovisioning the resources typically leads to performance degradation, especially in the case of latency-sensitive applications. In addition to this, in server computing, insufficient resource isolation can cause users to be overcharged. For providing resource isolation in the container environment, Linux uses cgroups. Using cgroups, one could allocate, meter and enforce resource usage for the containers.

A container can consume more CPU than allocated by its cgroup when it is co-located with other containers sending or receiving the network traffic, thus breaking the resource isolation. The culprit behind this problem is the interrupts raised when the NIC receives or transmits the packets, and time spent handling these interrupts might not be charged to the container sending or receiving the network traffic. Without properly charging the interrupts to the containers responsible for it, the kernel cannot provide resource isolation. The experiment results show that containers dealing with high traffic rates could slow down other co-located containers' performance by a factor of 6. The overhead is significantly high because the kernel needs to perform network processing which involves servicing the interrupts, protocol handling, and implementing network functions virtualization.

Linux tries to reduce the overhead associated with interrupt handling by servicing interrupts in two parts: a top half responsible for handling hardware interrupts, and a bottom half responsible for managing software interrupts. The hardware interrupts can occur at any point in time regardless of which process is running currently. The bottom half or the software interrupts are considered lightweight, and they perform critical actions required to serve the interrupts. Linux networking stack employs softirqs to implement the bottom half. Softirqs are responsible for transmitting deferred transmissions, managing packet data structures, and send the received packets through the network stack. The main culprit for breaking the resource isolation is Linux's way of handling softirqs. Software interrupts run in process context. That means whichever unlucky process running will have to use its scheduled time for handling softirqs. The kernel further limits the overhead associated with softirqs by limiting the softirq handler to run for a fixed amount of time. After the budget has been exhausted, schedulers use a special kernel thread called ksoftirqd to service the remaining softirqs. As ksoftirqd is a kernel thread, time spent handling interrupts by ksoftirqd is not charged to any container, breaking the isolation.

Iron uses a set of kernel instrumentations to measure the cost associated with interrupt handling of packets without compromising on the efficiency and responsiveness of the interrupt handling procedure. Iron can be integrated with the Linux scheduler such that the cost is always charged to the container responsible for handling the network traffic. Charging alone is insufficient to provide performance isolation because the container can still process the traffic, breaking the isolation. Iron leverages a hardware-based mechanism to drop packets destined to a container that has exhausted its allocation.

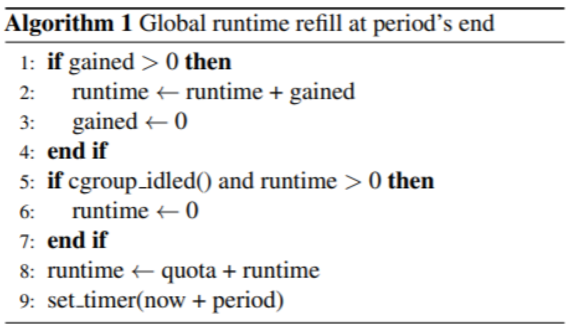
System Model: Iron first measures the cost associated with handling softirqs per-packet basis. Later on, with the help of the Linux scheduler, Iron associates the computed cost to the responsible container. When the container's runtime is exhausted, Iron uses a hardware-based technique to drop the incoming packets to the container.

Accounting of time spent for handling interrupts is different for packet reception and transmission.

1. Receiver-based accounting: Once the NIC receives packets from the network, the packets traverse through a series of nested function calls. For instance, the IP handler of a packet will call the transport handler. Thus, the function at the lower level of the network stack has the potential to capture the time spent processing a packet effectively.
2. Sender-based accounting: For transmitting the packet, the kernel has to obtain a lock on a NIC queue. As getting locks per-packet basis is expensive, Linux uses packet batches for transmission, and then an equal share of batch cost is charged to each packet.

Once the cost associated per packet is obtained, Iron relies on the Linux CFS scheduler to charge the cost to the appropriate container by instrumenting the kernel carefully. Linux CFS scheduler imposes resource isolation using a hybrid mechanism that works at local and global states. The global state works at a coarse-grained level, whereas the local state works at a fine-grained level per-core basis. The containers are allowed to run for a given quota within a period. At the beginning of each period, the global state maintains a variable named runtime and sets its value to quota. Later, the scheduler subtracts a slice from the runtime allocates it to the local core. This process continues until the value of runtime reaches zero or the period expires. At the end of each period, the value of runtime again initialized with quota. The local state has a rt_remain variable assigned with slice intervals subtracted from the runtime variable. As a process within the cgroup uses the CPU, the value of rt_runtime will be decremented. When the value of rt_runtime reaches zero, the scheduler requests a new slice from the runtime variable of the global state. If successful, rt_runtime gets refilled with slice, otherwise the local cgroup must wait until the period ends.

The Figure 4.1-1 shows the global scheduler updated by the Iron mechanism. Using a variable gained, the global scheduler tracks the extra time given to a container because it processed the sofirqs of other containers. Line 2 of the algorithm indicates this procedure, where gained is added to runtime. As the CFS scheduler does not allow the reusage of unused cycles from the previous period, the runtime is reset to 0 at the end of each period (Lines 5-7). Line 8 refills the runtime with quota.



Algorithm 1 Global runtime refill at period's end

```
1: if gained > 0 then
2:     runtime ← runtime + gained
3:     gained ← 0
4: end if
5: if cgroup_idled() and runtime > 0 then
6:     runtime ← 0
7: end if
8: runtime ← quota + runtime
9: set_timer(now + period)
```

Figure 4.1-1 Iron's Global State Algorithm[13]

Iron's local algorithm is given in Figure 4.1-2. When the value of rt_remain reaches 0, the local scheduler calls this function. A variable named cpuusage is used to track the accounting for each container. A negative cpu usage value indicates that the container has processed another container's network traffic, and it needs credit for that (Lines 10-12). A positive cpuusage value shows that the container must be charged for unaccounted network traffic processing (Lines 3-9). The remaining lines are unchanged, and they are part of the original local scheduler.

Algorithm 2 Local runtime refill

```
amount $\leftarrow 0$
min_amount $\leftarrow$ slice - rt_remain
**if** cpuusage $> 0$ **then**
    **if** cpuusage $>$ gained **then**
        runtime $\leftarrow$ runtime - (cpuusage - gained)
        gained $\leftarrow 0$
    **else**
        gained $\leftarrow$ gained - cpuusage
    **end if**
**else**
    gained $\leftarrow$ gained + abs(cpuusage)
**end if**
cpuusage $\leftarrow 0$
**if** runtime $= 0$ and gained $> 0$ **then**
    refill $\leftarrow$ min(min_amount, gained)
    runtime $\leftarrow$ refill
    gained $\leftarrow$ gained - refill
**end if**
**if** runtime $> 0$ **then**
    amount $\leftarrow$ min(runtime, min_amount)
    runtime $\leftarrow$ runtime - amount
**end if**
rt_remain $\leftarrow$ rt_remain + amount
```

Figure 4.1-2 Iron's Local State Algorithm[13]

The next component involved in the Iron's design is responsible for dropping the packets so that a throttled container cannot accrue more network-based processing. When a container sending network traffic gets throttled, it cannot generate more outgoing traffics by default. However, a traffic receiving container needs a mechanism to drop the incoming packets so that it will not consume the CPU quota allocated to other containers. Iron uses a hardware-based packet dropping mechanism that can be easily integrated with current architectures. When a NIC receives a packet, it DMAs the packet to a ring buffer in the shared memory. Then NIC generates the hardware interrupt, which in turn calls the interrupt handler. NAPI manages network interrupts in modern systems. NAPI, upon receiving the packet, disables hardware interrupt and notifies the operating system to schedule a polling mechanism to poll the packets. More packets arriving at the NIC will be added into the ring buffer by the NIC during this time. The kernel, while retrieving the packets from the NIC, retrieves a few packets bounded by the budget. In modern systems, NIC uses multiple queues for inserting incoming packets. Assuming that there are as many queues as containers, Iron maps each queue to a container using task_group. When a container is throttled, Iron modifies a bit in the task_group. This procedure overrides the NAPI default procedure so that kernel will not retrieve the packets from the queue whose containers are throttled. When the container gets unthrottled by the scheduler, the task_group bit will be reset.

Performance Evaluation: The experiments are conducted on Super Micro 5039MS-H8TRF servers with Intel Xeon E3-1271 CPUs having four cores and a CPU frequency of 3.2 GHz. The server runs Ubuntu 16.04 LTS with Linux kernel 4.4.17. The NICs are set with 25 Gbps NIC for UDP traffic and 10 Gbps for TCP traffic. The containers are created with lxc and open Virtual Switch as the virtual switch. For creating containers with sending and receiving traffic, simple TCP and UDP sender and receiver programs are utilized. As an evaluation metric, the penalty factor is used.

In each experiment, n containers per core are instantiated, and each container is configured to get an equal share of the CPU quota. One container per core is considered a victim, and the victim's performance is measured under a baseline scenario and interference scenario. The baseline scenario is one in which all the containers run CPU-intensive tasks such as sysbench. The interference scenario is the one in which all the containers except victim run network flooding applications. In both scenarios, the completion time of the victim is measured. The penalty factor is the ratio of interference scenario and baseline scenario. A penalty factor greater than 1 indicates the isolation is broken. Without incorporating Iron, when the sysbench is co-located with multiple UDP senders, the penalty factor is as high as 1.11 when n=10. However, when the Linux schedulers incorporate Iron, for the UDP sender case, the penalty factor remains below 1.04, which indicates significant improvement. In the case of TCP sender, Iron experiences a penalty factor of 1.04. For the same experiment without Iron, a penalty factor of 1.85 is observed. On similar lines, Iron shows significant improvement in penalty factor for both UDP receiver and TCP receiver.

4.1.4 Summary

There are very few works in the literature concentrating on Interference-aware VNF placement, which can be classified into prediction-based techniques and partitioning-based techniques. These categories complement one another. SLOMO approach is a prediction-based technique that successfully predicts the performance of a target VNF in the interference environment. However, it involves an expensive and very time-consuming dataset creation phase. Also, SLOMO considers the performance interference scenario in a DPDK enabled clickOS platform. While applying the SLOMO technique to the servers which rely on kernel-based network stack, software interrupts, as discussed in Iron, could be a significant source of contention that is not addressed in SLOMO. ResQ has a partionining based VNF placement technique that takes advantage of CAT, a hardware-based LLC cache partitioning technique. But ResQ has only considered the LLC as the primary source of contention, and contention for other resources is not considered. Moreover, both of these approaches did not address dynamic VNF placement or VNF migration which is most important in the MEC context as mobile terminals tend to move from one location to another.

4.2 Flexible localization for fine-grained proximity services in 5G and beyond network

Flexible localization for fine-grained proximity services in 5G and beyond network is the allotted topic according to WG2, but currently I am working in above mention topic. In future I might work on the localization topic in cell-free environment.

4.2.1 Reinforcement learning-based joint task offloading and migration schemes optimization in mobility aware MEC network [28]

Main Idea: Intelligent edge computing performs data collection, computation and analysis near to the users and gives a timely feedback. If the user with Mobile Equipment (ME) are moving offloaded task/feedback is not returned to user successfully leading to the need for efficient task migration. The paper proposes Joint task offloading and migration schemes in mobility-aware MEC network based on Reinforcement Learning (RL) to obtain the maximum system revenue.

System model

- **Network model**: As shown in Figure 4.2-1, two layer cellular network consisting of Macro Base Stations (MBS) and many Small Cell Networks (SCNs) each with a Secondary Base Station (SBS) is considered. MEC are at the center of SCN and MEs are distributed randomly and are considered to be either idle or moving.

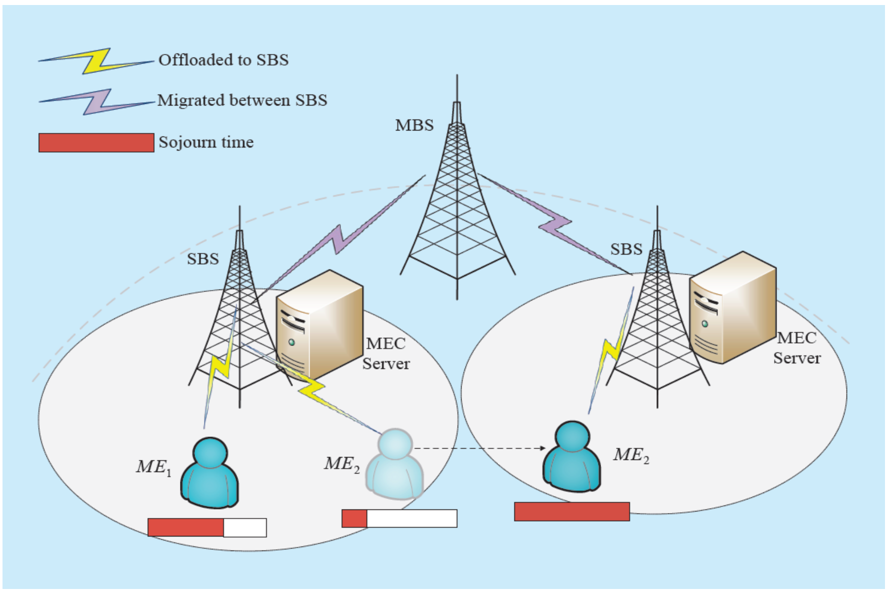


Figure 4.2-1 System setup [28]

- **Computation model**: Computation intensive tasks are requested by MEs, so offloading these helps in fast processing, saving time and energy consumption for devices. Tasks can be locally calculated by MEs i.e. Mode0-Local computing or handled on the MEC server at SBS side i.e. Mode1-Edge computing
- **Mobility model**: Let d_i be the distance between ith user ME_i and SBS and R_s be the radius of SCNs. The equation $\boldsymbol{d_i} \leq \boldsymbol{R_s}$ is always true if ME stays in its SCN until the task is completed. **Cell sojourn time** is calculated based on duration of time period before which d_i breaks the limit. It is considered to be the time spent by the ME in the given cell and is considered to be an important for planning network resources and improving QOS

Problem statement: Utility function is defined for offloading resource optimization problem which includes revenue and cost. It describes MEs performance. Revenue consist of two part: spare time and spare local energy and costs sources are MEs selection of Mode1-Edge Computing and possible task migration. The resource cost includes energy consumption of data to be transferred and MEC execution to be allocated.

According to time-length relation between edge-calculated time and estimated sojourn time utility function leads to two cases,

- Before leaning the current cell task has been smoothly completed. i.e offloading time < sojourn time eg. ME1 in Figure 4.2-1

- Situation where ME2 has a shorter stay in previous cell. It leaves the cell before task is completed causing migration of task from previous SBS to new SBS and then migrating the updates back to original/first SBS . This induces additional task migration cost in to the utility function

With the help of utility function a joint optimization problem of tasks offloading decisions and computation resource allocation is described with appropriate constraints.

Methodology: The joint optimization problem described above is a Mixed Integer Non Linear Programming (MINLP) problem. It is non-convex and NP-hard, so reinforcement learning is used to find optimal solution.

- **Marcove Decision Process** (MDP) algorithm is used to find exact state action and reward of reinforcement learning solution.

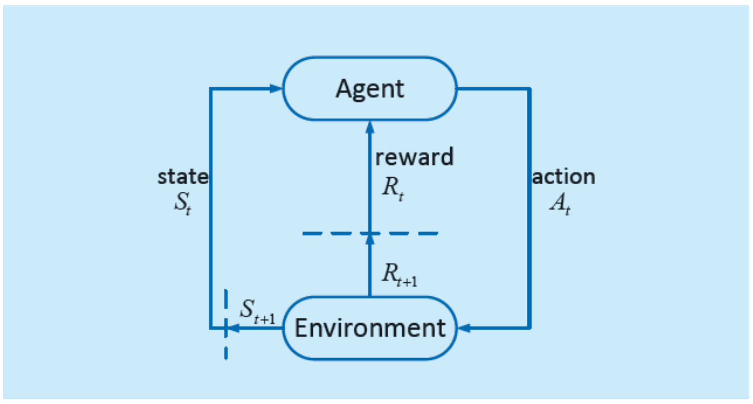


Figure 4.2-2 Reinforcement Learning [28]

where,

- Agent – MEs
- Environment – Channel conditions and MEC servers
- Action – MEs selection and execute an action in current state according to policy used at every time sept t
- State – number of MEs, unsolved tasks property i.e size of computation resource required by task and movement of MEs
- Reward – After each agent-environment interaction MEs as an agent will get feedback i.e reward from the environment.

The main goal of RL is to find maximum benefits of all MEs with maximum return.

- **Q-learning based algorithm**

It is a value-based reinforcement learning. The key step here is the formation of Q-table consisting of Q values obtained from state-action pair. Q value is the expectation benefit than an action ´a´ can be taken to obtain a state ´s´. The most profitable choice is taken based on the Q-table so that the agent is able to obtain maximum benefit.

Key findings: The main take away point of comparing reinforcement learning based algorithm when compared with traditional algorithm are,

- The convergence time of reinforcement learning is large. It is also impacted by number of MEs
- The goal of attaining high system revenue is achieved better by reinforcement algorithm

4.2.2 Joint radio and computation resource allocation in 5G open radio access network [29]

Main idea: Artificial Intelligence (AI) technique is to address the problem of radio (resource blocks) and computing (virtual machines) resource allocation and minimize total network latency. Reinforcement learning based scheduler helps in making efficient scheduling decisions and can support the rapid dynamics in practical O-RAN network taking advantage of the decentralized learning process.

System model: Open-RAN and the core network serve multiple users with different QOS requirements. Each user requests its serving O-RU a number of resources to access a service. The traffic corresponding to the user is queued in transmission buffer maintained at each O-RU and the traffic of each O-RU is queued in transmission buffer maintained at each vO-DU. At every TTI cycle the scheduler in the Near-RT RIC allocates radio resources to the active users and computation resources to the O-RUs based on QOS demands of users.

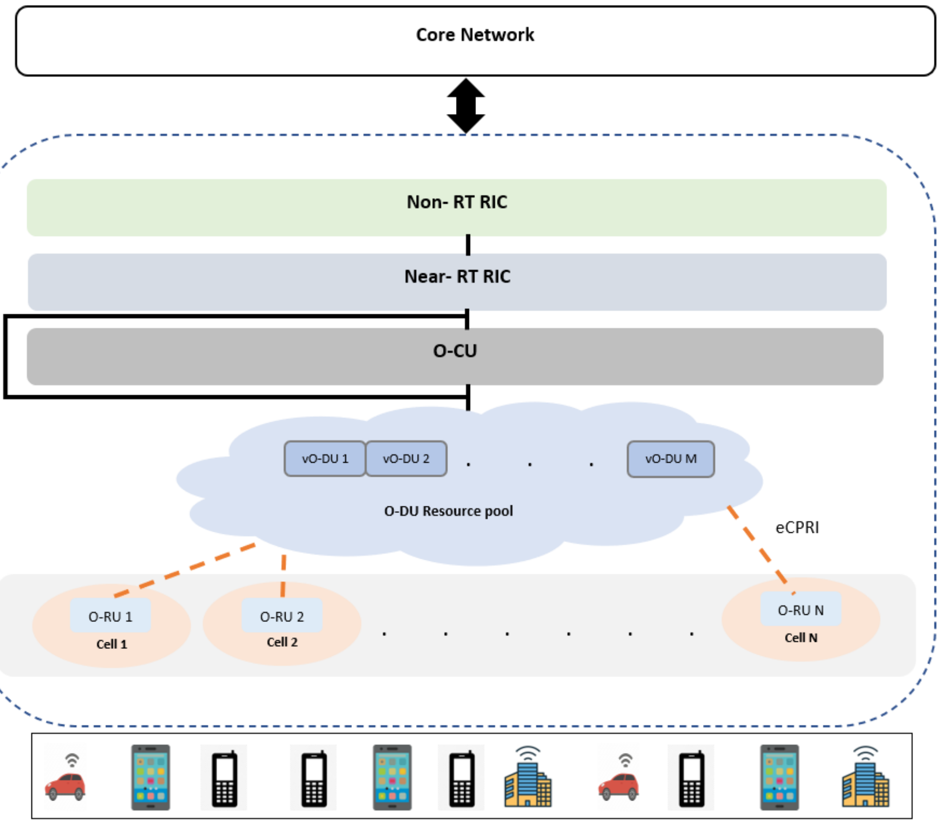


Figure 4.2-3 O-RAN based system setup [29]

The end-to-end latency of each user request is decomposed in to 5 components: HARQ retransmission delay, transmission delay, queuing delay, fronthaul delay and computing delay.

Problem formation: The objective is to find best scheduling strategy for resource block allocation (between O-RU and users) and computation resource mapping (between vO-DU and O-RU) to achieve a minimum network latency with following constraints,

- Each user is associated with single O-RU
- Interference avoiding – Avoid allocation of same radio resource to two users served by same O-RU
- System stability – total number of scheduled resource blocks must not exceed network capacity
- Queuing stability

- SINR threshold – User is eligible to use resource blocks only when instantaneous SINR exceeds the threshold
- Power budget and computation resource capacity.

Methodology: Continuous time Markov Decision Process (MDP) for reinforcement learning. The reward here is considered to be inverse of latency, so achieving minimum latency will lead to high reward. Here the workflow is as shown in the Figure 4.2-4.

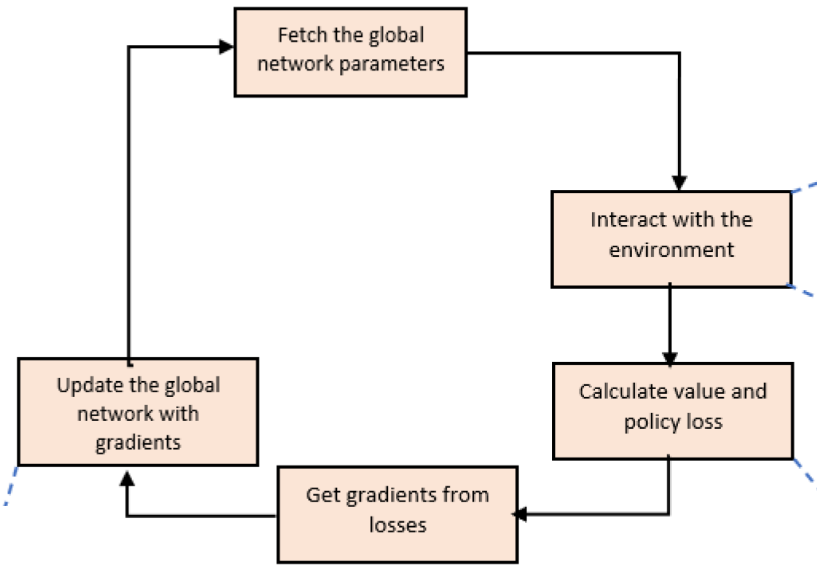

Fetch the global network parameters
Interact with the environment
Calculate value and policy loss
Get gradients from losses
Update the global network with gradients

Figure 4.2-4 RL based scheduler [28]

Key findings: Performance evaluation is based on two main metrics: 1. Total scheduling time and 2. Total network latency. The reinforcement learning based model is compared other state-of-the-art models against the above 2 main metrics. The key points to be noted are,

- The reinforcement learning based model outperforms few other state-of-the-art models with acceptable scheduling time and in terms of latency it is close to optimal
- The convergence of algorithm is efficient

4.2.3 Cross-silo model-based secure federated transfer learning for flow-based traffic classification [30]

Main idea: Traffic classification is important for autonomous network management. Federated learning based traffic classification methods are in demand because of their ability to accurately classify even encrypted traffic with privacy preservation. A secure federated transfer learning is performed for improvising the training-time and accuracy of the target federated model for traffic classification.

System model: Source module and target module are the two federations, and each federation is considered to have two organizations as shown in figure 4.2-5. Each organization has a particular data set D_i where i = {I,J,K,L} with sample space. Organization I and J collaboratively train source module M_S^F with M_I^F and M_J^F being the local model. The same scenario applies to target module.

Problem statement: A federated transfer learning problem is formulated with tasks assigned for source and target module. The aim of target module is to increase learning accuracy and decrease the training time based on knowledge from source module data and target task. Traffic classes considered here are VPN and non-VPN based traffic, broadcast, mail, streaming, VOIP, etc.

Methodology used: Federated learning implemented in tenser flow platform.

Key findings: The knowledge transfer based federated model takes less time to train when compared to model which is used to train the target module from scratch also is shown to be more accurate.

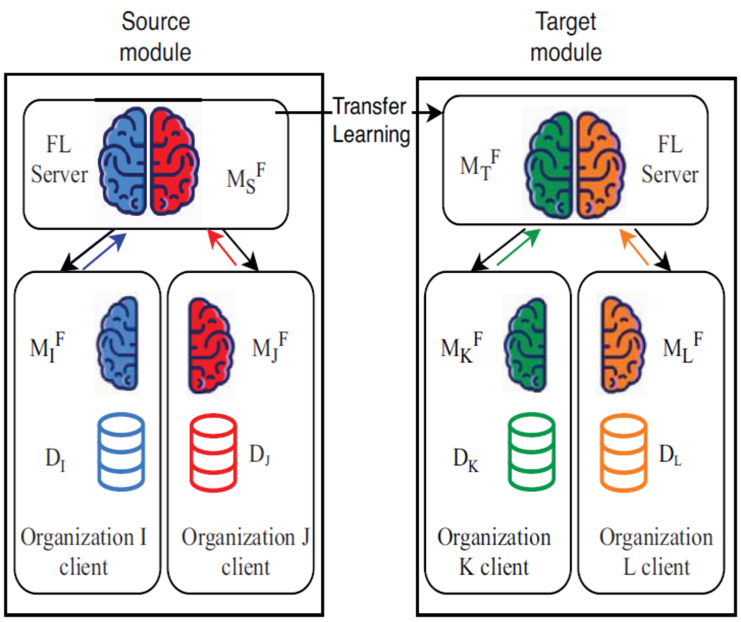


Figure 4.2-5 Federated Learning based system setup [30]

4.2.4 MEC enabled cell selection for micro-operators based 5G open network deployment [31]

Main idea: Heterogeneous networks consist of multiple frequency bands and combination macro and micro cells. Depending on various services and propagation properties of spectrum an optimum cell selection algorithm is designed to enable 5G users to select the cell based on MEC framework to achieve maximum throughput with better power efficiency.

System model and problem formulation: As indicated on Figure 4.2-6, the environment has micro and macro-operator. The Edge Enabler Server in the network maintains the information about Edge Application Servers (EAS) in the area provides MEC services, UE identity and location accessed from 5G core network, number of micro-operators in area and their location, cell load information's etc., which would help in selection of suitable cell. The Edge Enabler Client (EEC) gets information about the suitable cell to select from EES.

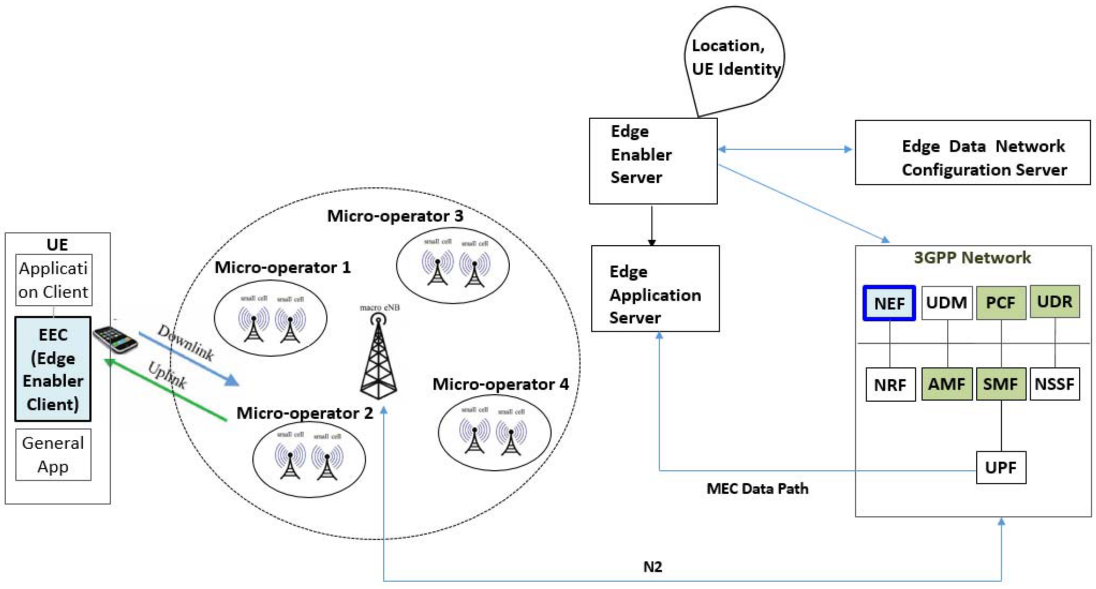


Figure 4.2-6 MEC based framework [31]

Methodology: MES collects all the required information about UE location and cell load conditions. Later it measures throughput and power requirements for each cell. Based on the results obtained it selects the cell with better measurements and communicates it to UE.

Key findings: The proposed MEC based cell selection outperforms the traditional 5G cell selection in terms of achieving power optimization and better throughput. This can be in future combined with AI/ML based algorithms and improve a bit to take more intelligent and fast decisions.

4.2.5 Future Work

- Find and solve challenges related to RAN/MEC integration using AI/ML algorithms
- Survey on integration of cell-free with O-RAN based 5G
- Flexible localization for fine-grained proximity services in 5G and beyond.

4.3 Edge content placement and delivery in converged MEC/RAN

Content placement and delivery techniques have established high technological potential towards establishing efficient and reliable content interchange. Currently, there is a plethora of state-of-the-art (SotA) publications that focus on the efficient utilization of available resources in order to obtain high QoE of the end-clients. The key subsections that are included in the content delivery domain are contentment caching, content popularity estimation, DASH, mobility prediction techniques, and federated learning, some of which will be described in this section. The domain of dynamic bitrate adaptation has already been very well studied, however, with the fast development of novel ML tools, the implementation logic for DASH algorithms can be significantly improved in terms of utilization complexity and achieved QoE. Based on that, the works [32]provide not only knowledgeful large-scale analysis of the underlying structure of the DASH algorithm but also advanced decision-making methods, that can be potentially industrially implemented. Regarding caching, the works [35], [36], [37] provide a thorough analysis of the SotA tools for caching, along with techniques caching optimization improvements. The selected works release particularly practical and applicable logic based on modern ML algorithms with high accuracy and relaxed complexity.

4.3.1 D-DASH: A Deep Q-Learning Framework for DASH Video Streaming [32]

Main idea: The authors in [32] proposed a client-side Deep Learning-based approach for efficient implementation of the dynamic adaptive streaming over HTTP (DASH) called – D-DASH. The key goal of the work is to provide a Deep Q-learning bitrate adaptation algorithm that can enhance the user-perceived QoE and at the same time tackle the application-essential complexity level.

System model: In D-DASH, the system definition consists of a single DASH client that downloads a video file from the original server segment by segment. The streaming is modeled as a sequence of scenes with exponentially distributed duration, and therefore in this case the model can be represented by the Markov Decision Process. The authors proposed the RL-based Q-learning integration which keeps the table of estimates $Q(s,q)$of the expected long-term reward for each state-action pair (s,q), which is a typical reinforcement learning transition. In order to deal with the curse of dimensionality of the states and optimization computational complexity, the estimation of Q-values has been performed by integration of three different neural network models – 1 layer multilayer perceptron (MLP), 2 layer MLP, and long short-term memory (LSTM) network. The model of the D-DASH algorithm is a throughput based, hence the baseline parameters that are predefined to the model are – currently experienced throughput, buffer time size, rebuffering time, and the current and previous quality of the segment.

The functional design of the D-DASH algorithm can be described as follows. First, the state $s_t$is updated and is fed into the designed Network as an input. The schematic of the system model update operation is shown in Figure 4.3-1. The key function of the network here is to perform a function estimation and output the probabilistic set of Q-values – that depict the combinations of values of given states corresponding to policy available policies – $Q(s,q)$. Based on the produced Q-values, the next block represents the “choose action” strategy, which basically implements $softmax$ or $\varepsilon - greedy$ functions to select the best video bitrate (typically highest Q-value) out of the set. After that step, the new quality rate has been triggered for the request and therefore, the model is now ready to perceive the updated environment and update the network with reward. As can be seen in Figure 4.3-1 after interaction with the environment (sending the request of q_t, the reward and new state s_{t+1} are produced. The reward produced depends on the quality metrics assigned in the work, and in case of rebuffering event occurrence, smoothness, or quality degradation – the reward becomes a penalty factor that corrects the network respectively. The reward is then stored in the replay memory block, along with other models' beneficial information as previous state s_t, current state s_{t+1}, and previously assigned bitrate q_t. The replay memory block is used to train the neural network, implementing basic loss function integration and updating the network`s weights correspondingly. Additionally, the authors proposed to clone the primary network each $K$ time steps to use it as an update in case of extreme networks` weight changes that can significantly degrade the performance. Finally, the new state is then updated and fed into the network by repeating the consecutive update operation.

An important notice here, that the authors wisely utilize three cases of different NNs, that combine not only the approximation efficiency properties, but also bring additional training complexity (in case of LSTM) and extra memory utilization. This way of comprehensive evaluation of ML-based adaptability types increases the research interest of this work.

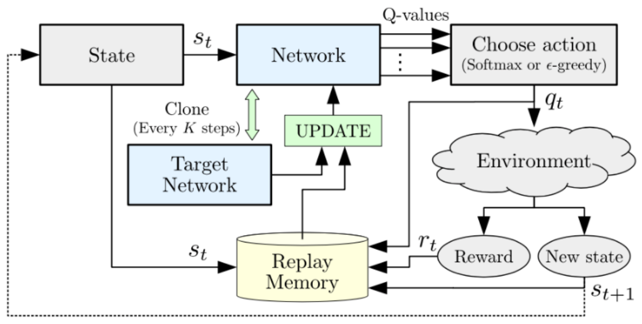


Figure 4.3-1. Schematic of the system model update operation [32].

Evaluation: Another important aspect of any DASH adaptation algorithm is the system evaluation metrics. The authors used the SSIM as the instantaneous quality metric, which has already been widely used in the literature [38] and is shown to be highly correlated with the perceived user QoE. Additionally, the authors really their performance evaluation on two factors – video quality stability, video rebuffering events.

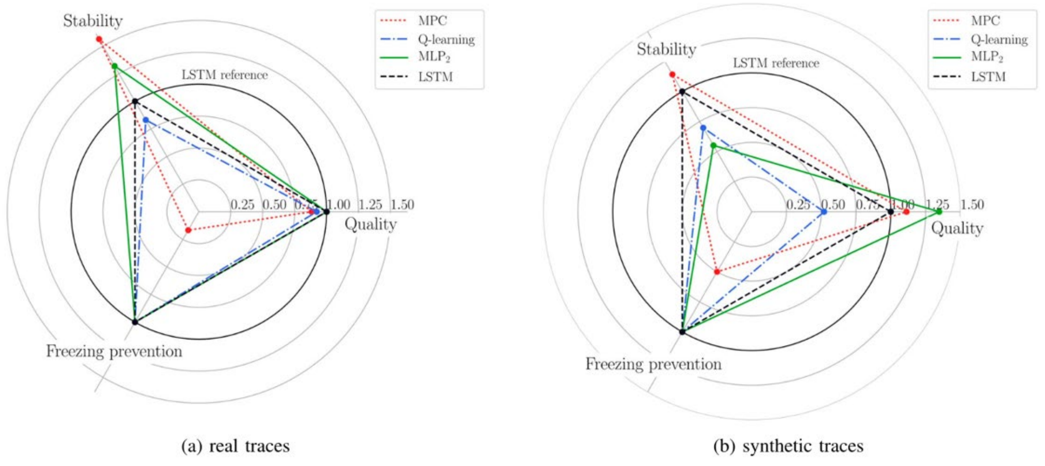


Figure 4.3-2. Summary of performance of video adaptation algorithms [32].

As can be seen in Figure 4.3-2 the proposed D-DASH algorithm, that combines MDP with Q-learning and Deep Learning is shown to achieve high freezing prevention on the real and synthetic traces, while keeping the overall SSIM quality at a similar level in comparison to the benchmarked Model Predictive Control (MPC) model [39]. However, the D-DASH algorithm losses to the MPC in terms of keeping the video quality variations stable in time in both real and synthetic traces evaluation.

4.3.2 Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale [15]

Main idea: Despite the novel proposed adaptive video streaming technique, the work [15] provided exhaustive research on the throughput estimation and variability of bandwidth sharing, when multiple HAS streams compete for bandwidth. The knowledge of how the throughput is shared between clients and by probing the channel with rate incrementation the authors proposed and a novel probe and adapt algorithm – PANDA.

System model: In PANDA, the multiple HAS clients compete for the available bandwidth and try to download the video files segment by segment from the original HTTP content server. The system model is defined by the client-server interaction process, which contains predefined parameters such as measured TCP throughput, segment downloading time, segment bitrate level, and buffer dynamics. The conventional

HAS adaptation algorithm is modified by implementing four unique decision making and adaptation sub-steps. The PANDA algorithm is shown in Figure 4.3-3.

1. *Estimating* – first, the client continuously increments the target average data rate by a constant value per unit time as a probe of the available capacity, which makes the algorithm very agile to the bandwidth changes.
2. *Smoothing* – second, the smoothing function is applied for the estimated throughput to noise-filter the available data, to remove outliers.
3. *Quantizing* – third, the continuous smoothed version of estimated throughput is then mapped to the discrete video bitrate, possibly with the help of side information such as client buffer size
4. *Scheduling* – finally, at this step the model aims to determine the target inter-request time, by calculating the time schedule for the next segment to be downloaded next.

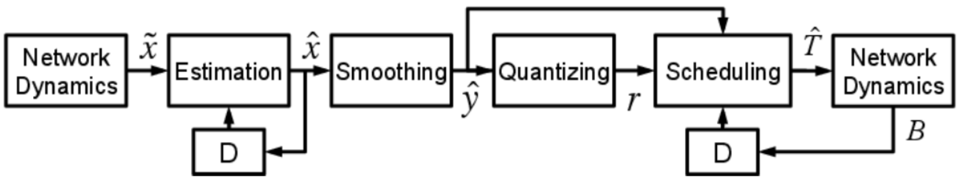


Figure 4.3-3. Block diagram for PANDA algorithm [15].

Evaluation. In order to map the performance of the PANDA, the authors used the predefined evaluation metrics, each responsible for a particular part of users-perceived QoE evaluation:

- *Buffer undershoot* – measures how much the buffer drops after a bandwidth drop as an indicator of an algorithm's ability to avoid buffer underruns.
- *Instability* – like the difference between the currently adopted video rate and the previous one, that aims to avoid frequent and drastically video bitrate changes.
- *Inefficiency* – which defines high average quality obtained during the transmission of video segments, and aims at fetching by the client as high bitrate as possible.
- *Unfairness* – aims to evaluate how fair the channel bandwidth is shared between HAS clients.

The PANDA algorithm has been shown to efficiently tackle the bitrate oscillation problem, by establishing high smoothness of the downloading stream. Additionally, the authors state the high potential and flexibility in the utilization of the PANDA algorithm in more advanced adaptive bitrate streaming combinations, which can lead to high QoE enhancements while keeping the implementation cost low.

4.3.3 Echo State Networks for Proactive Caching in Cloud-Based Radio Access Networks with Mobile Users [36]

Main idea: The work [36] has introduced a novel content distribution and mobility pattern prediction algorithm that decides which content to cache at the remote cache servers. The introduction of the echo state network for predicting the content distribution and users` mobility makes this work distinctive from conventional approaches, and particularly attractive for research.

System model: First, the authors consider a cache enabled CRAN, with a downlink transmission where the set of users U is served by the set of the remote radio heads (RRH) – R. The RRHs which hold the same content request distributions can be grouped into a virtual cluster. The users, however, have the predefined periodic mobility pattern, that is also periodically collected. At the same time, the context of each user at each time step is collected to create a set of features including content request time, week, gender, occupation, age, and device type. Based on these two sequences, the two recurrent Echo State

Networks (ESN) are applied to predict first the future content distribution and second the mobility pattern of each user. The system model is shown in Figure 4.3-4.

Evaluation: The main goal of this work is to develop an efficient caching scheme and content RRH clustering approach to highly reduce the interference and offload the traffic of the backhaul and fronthaul based on previously defined users` predictions. To achieve this goal QoS and delay optimization problems are formulated whose objective is to maximize the long-term sum effective capacity. The presented results have shown that the ESN-based approach provides significant performance gains in terms of sum effective capacity compared to the conventional approaches.

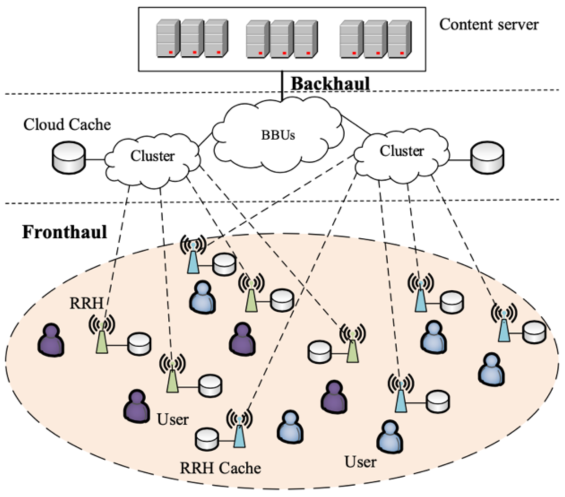


Figure 4.3-4. Proactive system model in cloud-based radio access network [36].

4.3.4 RL-Cache: Learning-Based Cache Admission for Content Delivery [35]

Main idea: Considering the content delivery methods, the cache-aware algorithms have been seen a growing interest from the research community. In work [35], the authors propose a cache admission algorithm called RL-Cache that uses a feedforward neural network to decide whether or not to admit the object, that has been recently requested to the CDN`s cache.

System model: Typically, the conventional approaches – LFU, LRU, LRFU, which are described in Section 3.9, use a small set of available features to perform the decision. However, the RL-Cache algorithm utilizes a larges set, that consists of object size, recency, and frequency of access to the video content, for future decision making. The RL-Cache is based on the typical RL paradigm, where the admission decisions represent RL actions, and cache hits constitute RL rewards. The key goal of such a model is to design a policy function of states and actions to maximize the expected return, which basically corresponds directly to maximizing the expected cache hit rate. However, in this work, the policy implementation is done by introducing a feed-forward NN, that calculates the admission probability $A(u, \omega) \in [0,1]$ as a function of fed features u of the requested content file.

Evaluation: The RL-Cache is implemented at the cache server and is used upon receiving a request for a video file. After getting the request, the CDN server applies the predefined trained NN to the features to calculate the admission probability for the request. Finally, the computed probability is rounded up to 1 or 0 to make a final decision whether to admit or not to admit the requested object into the cache. The results of the RL-cache are benchmarked against the AdmitAll, frequency-based SecondHit, and adaptive size-based Adapt-Size approaches, showing the best hit-rate performance when using the full set of its eight features.

4.3.5 Online Proactive Caching in Mobile Edge Computing Using Bidirectional Deep Recurrent Neural Network [37]

Main idea: Another work [37] proposing a novel proactive caching algorithm, that relies on Bidirectional Deep Recurrent Neural Network (biRNN) model and predicts the time-series users` requests, with an edge caching update function, that aims to maximize the cache hits. Despite the idea of reducing the traffic loads in the core network, the goal of work is to improve the service response time for active clients.

System model: The authors consider a cache-aware base station (BS) that serves multiple users under its coverage, which can be seen in Figure 4.3-5. By analyzing the content requests behavior during the off and peak times, the system in this work is designed to predict the time-variant user requests and can accordingly update the cached files at the BS proxy to increase the cache hit rate (CHR). First, the authors defined by F and C as the set of files in the remote cloud server and the local cache storage, respectively. By defining the user`s request file set $F_{req}$at a time slot $t$, the model can check if the file of this set s cached in the edge server and can directly fetch it from it. Otherwise, the downloading procedure has to be performed from the remote cloud server. The goal of this work is to predict the possible user`s request sequence to achieve a high cache hit and reduce the backhaul congestion.

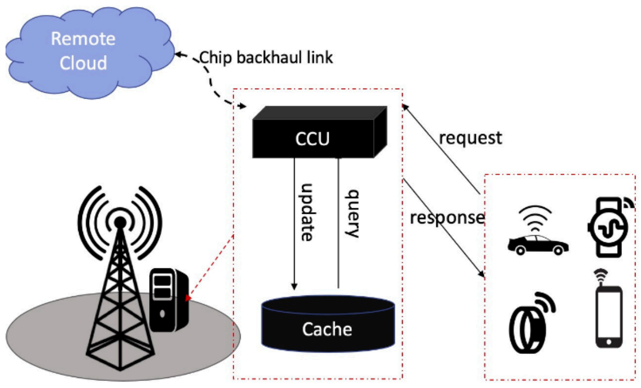


Figure 4.3-5.System model for proactive caching [37].

Evaluation: The authors proposed to implement a combination of the convolutional neural network, which is used to learn the sparse of the data, recurrent bidirectional LSTM block, and the fully connected NN block to predict the time-series requests. The models proposed, have been trained by the supervised algorithms with collected real-world data traces and have been shown to achieve enhanced cache hit ratio in comparison to other benchmarked deep learning models.

4.3.6 CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction [33]

Main idea: Another exceptional work on Video Bitrate Adaptation has been presented by Yi Sun in [33]. First, the authors provided exhaustive large-scale analysis of datasets containing essential throughput characteristics, summarizing their findings by two major aspects:

- Sessions that share similar key features (region, Internet Service Provider) display similar initial throughput values and dynamic patterns;
- The natural “stateful” behavior in throughput volatility within a chosen session is observed.

Second, the authors propose the server-side bitrate adaptation model-based throughout prediction with the data-driven learning approach – called CS2P. The CS2P approach is designed to help to select a suitable initial bitrate when a video session starts – so-called Initial Throughput, and develop adaptation approaches using throughput estimation in the so-called Midstream throughout. The system is shown in Figure 4.3-6.

System model: The CS2P is a data-driven model and therefore, the first step performed in this work is the analysis of HTTP throughput measurement data collected within the real traces of the iQIYI operational platform of client-server interaction. The features that are defined by the authors are – *Client IP*, *Internet Service Provider*, *Autonomous System* that client resides in, *Province*, *City,* and *Server*. Based on these collected features the Prediction Engine is proposed that uses cross-session stateful prediction models. Due to the fact, that throughput depends on the hidden state, e.g., the number of flows sharing the bottleneck link and link capacity, the throughout predictor in this work is defined as a Hidden Markov Model (HHM).

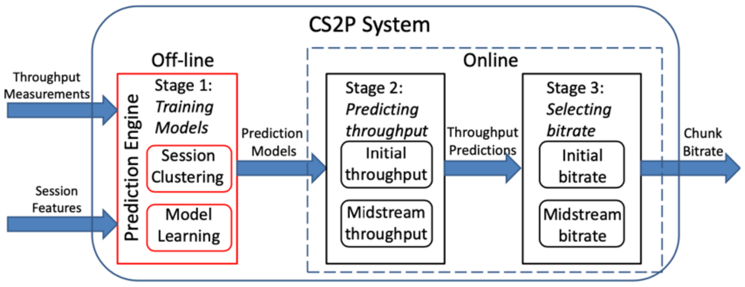


Figure 4.3-6. CS2P implementation paradigm [33]

Evaluation: The authors assessed their proposed algorithm against the following evaluation metrics:

- Absolute normalized prediction error – that is represented by error functions of the median of the per-session median, 90-percentile of the per-session median.
- QoE – which is defined as a linear combination of average video quality, average quality variation, total rebuffering time, and startup delay.
- Normalized QoE (n-QoE) – as a ratio of actual QoE and the offline optimal QoE, which is represented by given perfect throughput information in the entire future period.
- QoE – Average Bitrate – as a sum of value of the selected bitrates averaged.
- QoE – Good Ratio – as a percentage of segments without rebuffering.

In conclusion, the CS2P provides a flexible and easily deployable middle ground between complex centralized approaches [40] and decentralized bitrate adaptation algorithms [39]. It was also shown, that the trace-driven CS2P outperforms prior proposed solutions on both throughput prediction accuracy and video QoE.

4.3.7 Multipath-Based HTTP Adaptive Streaming Scheme for the 5G Network [34]

Main idea: The work in [34] has presented a multipath-based transmission scheme for HTTP adaptive streaming in 5G networks. The authors propose a collective segment request policy to establish an accurate measuring of the aggregated available bandwidth. Additionally, to prevent the reordering problem in multipath environments, the model of segment scheduling in combination with the rate adaptation algorithm for the proposed segment request policy is proposed. The particular goal of this work is to improve overall bandwidth utilization.

Providing a deep analysis of the conventional HTTP adaptive streaming, the authors define the system model by the system architecture that consists of 8 key components that are shown in Figure 4.3-7:

- The network monitor – that is responsible for available bandwidth estimation and measuring of each route and by using the predicted bandwidth of each path.

- The buffer monitor – is responsible for periodical monitoring of the client's buffer occupancy. Additionally, it is responsible for the rearrangement of the downloaded segments in each path to deliver them to the playback buffer.
- The parser – is responsible for XML-based manifest file analysis that has been received from the content server.
- The *quality module* – is responsible for defining the quality and the number of segments to request for each path with aim of enhancing users` QoE and bandwidth utilization.
- The *request controller* - is responsible for calculating the length of the request interval to improve responsiveness and determines the range of segments to request during the request interval to prevent buffer underflow and overflow.
- The *segment scheduler* – is responsible for the distribution of the order of request messages to each path.
- The *offloading controller* is responsible for checking the download status of each path and requests an alternate transmission using a partial-segment-based request policy [34].

__Evaluation__: The authors adopted the evaluation scheme that combines different aspects of users` QoE assessment for each particular block of the proposed algorithm. Mainly, the valuation relies on bandwidth utilization, which can be defined as:

$$Utilization = \frac{\sum R}{N \times R_{max}},$$

where N depicts the number of downloaded segments, R the bitrate of the requested video quality, and R_{max} depicts the bitrate of the server`s available maximum quality [34].

In conclusion, in this work, the authors show that the novel method of multipath-based transmission scheme for efficient bandwidth utilization of HTTP adaptive streaming precisely estimates the aggregate throughput and achieves a comparative improvement in the user-perceived QoE, while effectively utilizing the available network bandwidth.

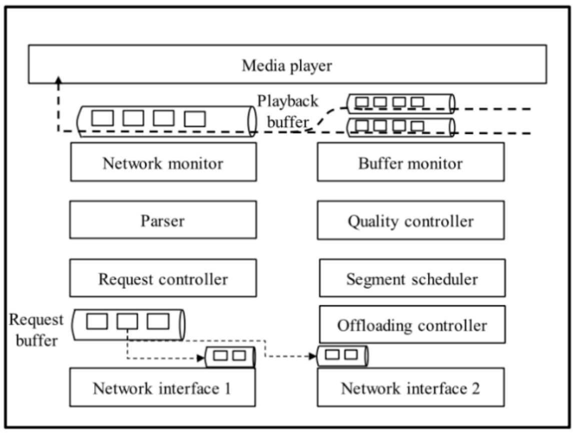


Figure 4.3-7. System Architecture of Multipath-Based HTTP Adaptive Streaming Scheme [34]

4.3.8 Summary

The works described in the previous section represent an exceptional volume of SotA techniques, that undoubtedly provide valuable innovation capacity to the content caching and DASH domains. Clearly, the system models differ one to the other with the strategically established parameters and particularly modeled design. At the same time, the problems tackled in the aforementioned works utilize the latest knowledge of the SotA algorithms and tools, that are found fit and beneficial in the implementation capabilities. Despite focusing on the standardization of the proposed novel algorithms, the main focus of the future research in such a domain is mainly – adaptation to the new generation mobile networks, high-reliability improvement, developed adjustability, and software and hardware compact realization. The main findings are summarized in table 4.3-1 for works on DASH, and in Table 4.3-2 for works on intelligent caching.

Table 4.3-1 Works focused on DASH implementation algorithms

	Works on DASH implementation			
	M. Gadaleta et al. [32]	Z. Li et al. [15]	Y. Sun et al. [33]	H. Kim et al. [34]
Large-scale analysis	Deep Learning	Throughout share	Throughput variability	DASH architecture
Type of adaptation	Throughput based	Throughput based	Throughput based	Throughput based
Multiuser		v		v
RL-based	V		v	
ML-based	V			
Metrics focus	Quality Rebuffering Smoothness	Quality Rebuffering Smoothness Fairness	Quality Rebuffering Smoothness	Bandwidth utilization
Experimental	v	v	v	

Table 4.3-2 Works focused on intelligent caching algorithms

	Works on Intelligent Caching		
	M. Chen et al. [36]	V. Kirilin et al. [35]	L. Ale et al. [37]
Large-scale analysis	Context Information	Cache Admission and Eviction Algorithms	Content Requests Patterns
Key Focus	Content Distribution Prediction	Cache Admission	Users` Requests Prediction
User Mobiliity considered	v		
RL-based		v	
ML-based	v		v
Metrics focus	Traffic Offload	Cache Hit Rate	Cache Hit Rate

4.4 Decentralized data access for massive connectivity at the network edge

Massive connectivity demands in future cellular networks pose a significant challenge for the traditional data access models due to presence of central entities for network resource control and management, e.g. mobile network operators, that might not be capable of handling vast number of connected devices alone. Decentralization of mobile data access, in turn, can eliminate the need for trusted external authorities and single-point failures as well as improve efficiency of service provisioning.

The state-of-the-art solutions relevant to decentralized mobile data access model will be categorized based on the addressed aspect. As the proposed mobile data access beyond 5G is to be deployed on top of a cellular network edge, mathematical models for describing and analyzing heterogeneous wireless **network topology** were studied first in Section 4.4.1. Next, in light of user-driven access model, the models related to the **mobile user behavior** such as mobility and video watching patterns were investigated in Section 4.4.2. To better understand potential of **blockchain for decentralized mobile data access**, an example of such solution - blockchain radio access network (RAN**)** architecture is examined in Section 4.4.3. Next, blockchain scalability challenge is addressed by **payment channel** method, namely lightning network, discussed in Section 4.4.4 that might be enabling tool for massive cellular communications. Next, inspired by solutions provided in two previous subsections blockchain usage scenario for **network service trading** in contractless mobile data access is scrutinized in Section 4.4.5 for further development of the beyond state-of-the-art decentralized data access towards massive connectivity enablement at network edge. Section 4.4.6 summarizes the presented SoA solutions in terms of used tools and KPIs, as well as discusses how these components can be relevant to the addressed problem of massive connectivity and data access.

4.4.1 Mobile network topology

4.4.1.1 Modeling and Analysis of K-Tier Downlink Heterogeneous Cellular Networks[41]

Main idea: Flexible and accurate yet analytically tractable model for K-tier downlink heterogeneous cellular networks (HCN) is proposed in [41].

System model: The modeled network consists of Poisson Point Process (PPP) distributed base stations (BS) belonging to K spatially and spectrally coexisting tiers each having various transmit power, spatial density and supported data rates. Mobile users are also modeled by PPP independently from BSs with a typical mobile user located at the origin. The wireless link between BS and mobile user is modeled as Rayleigh fading channel that is used to derive SINR expression as a function of distance from user at the origin connected to random BS.

Performance metrics: The system model is then used to derive analytical expressions for such performance metrics as coverage probability, average load per tier and average data rate achievable by mobile user under both open and closed access. Open access means that users can connect to any BSs in coverage such that the strategy is reduced to choosing the BS with highest SINR, whereas in closed access the mobile user can connect only to specific subsets of BSs belonging to home MNO so that the previous strategy will apply only to accessible tier. Clearly, the performance metrics will have lower values in the second scenario.

Key findings: The derived theoretical parameters are then evaluated against corresponding measurements acquired from simulation of an actual LTE macro-cell network with PPP distributed K-1 lower tiers. Other simulated scenarios include hexagonal grid and random PPP models. Numerical results have shown that the coverage probability in the proposed theoretical framework is accurate within 1-2dB as compared to the simulations with the PPP network providing a lower bound and grid model providing an upper bound to the actual coverage probability.

Pros/Cons: It can be concluded that the designed framework offers simpler mathematical analysis in terms of closed form expressions for performance metrics than 2-dimentional hexagonal grid model while preserving accuracy. The important limitation of the proposed framework is that it assumes uniform user distribution with BSs being available all the time.

4.4.1.2 Using Poisson processes to model lattice cellular networks [42]

Main idea: Usage of Poisson processes to model lattice cellular networks is justified in [42] by deriving and validating novel Poisson-convergence result for a broad range of stationary networks under sufficiently strong log-normal shadowing. The Poisson limit was demonstrated from the fact that propagation losses do not vary with distribution of shadowing or fading but only its moment. The minimum shadowing or fading variance for the Poisson approximation to hold reasonable is found through comparison of empirical hexagonal network and analytical Poisson model results. This model is further extended with mean energy efficiency function and its optimization.

System model: Firstly, authors demonstrate how 2D Poisson point process that models base station locations can be transformed to a point process of propagation losses experienced by users through distance-loss function and shadowing or fading variables. It was noticed that the distribution of propagation losses does not depend on the shadowing/fading distribution but only its moment of order 2/β, where β is the distance-loss exponent, such that the arbitrary shadowing can be statistically perceived as statistically constant shadowing. Next, it was demonstrated that as the variance of log-normal shadowing increases, stochastic process of propagation losses between user-station pair converges to non-homogeneous 1D Poisson process.

Key findings: In order to identify the optimum value of shadowing or fading variance for the Poisson process approximation to be accurate enough in modelling hexagonal networks, SIR cumulative distribution function (CDF) obtained from simulation of hexagonal model with shadowing is compared to the analytical SIR CDF for infinite Poisson model using Kolmogorov-Smirnov fit test. It was found that shadowing of at least 10dB makes the two CDF almost indistinguishable with critical p-value being equal to 10%. Hence, if the standard deviation of shadowing is greater than 10dB, then hexagonal cellular network can be perceived by its user as an infinite Poisson network. This serves as a quite realistic assumption for outdoor and indoor wireless communications in urban environment.

Performance metrics: In addition to the above, the authors present analytic tools to study SINR distribution in the Poisson model and use them to derive mean energy efficiency as a function of the base station transmit power. Comparison of optimal mean energy efficiency in hexagonal and infinite Poisson models also demonstrate the reasonability of Poisson approximation with strong shadowing.

In conclusion, this work serves a strong evidence for modeling any actual wireless network through Poisson processes as long as certain assumptions hold, i.e., shadowing variance is sufficiently high.

4.4.2 Mobile user behaviour

4.4.2.1 Towards Understanding the Fundamentals of Mobility in Cellular Networks [43]

Main idea: Random waypoint (RWP) mobility model defined on the entire plane is proposed in [43] to analyze handover rate and sojourn time. Classical RWP mobility model describes movement patterns of mobile nodes in a finite domain, which makes stationary spatial node distribution concentrated near its center and affect transition length as the latter has the same order of size as the domain. These two aspects might be inappropriate for describing movement in cellular networks due to different user distributions and deviations of transition lengths from domain size. Therefore, the proposed model defines node movement over the entire plane making it more appropriate model for mobility study in cellular networks. The proposed RWP model is then applied on top of hexagonal and Poisson Voronoi modeled cellular networks in order to analyze the handover rate and sojourn time.

System model: In the proposed RWP model, each node's movement is described by infinite number of quadruples containing starting and target waypoint, velocity and pause time; where waypoints designate node locations in the beginning and end of a single movement, velocity is the corresponding speed between waypoints and pause time is the immobility time spent at waypoint. Initial waypoint is set by homogeneous PPP with particular intensity, while target waypoint is conditionally defined by starting waypoint, uniformly distributed random direction angle and i.i.d. Rayleigh distributed transition length. Due to i.i.d. property of the latter, the waypoints in the proposed model form a Markov process. The velocity and pause time can follow any distribution.

Performance metrics: Using these properties, authors mathematically describe such parameters as transition time (found by division of transition length by velocity), period time (sum of transition time and pause time), direction switch rate (defined by inverse of period time), spatial node distribution (probability that the moving node resides in some measurable set during single movement with and without pause time). Transition length and direction switch rate statistics are then compared to those extracted from Classical RWP and synthetic truncated Levy Walk (constructed from real mobility trajectories) simulations.

Key findings: It was demonstrated that the proposed RWP model has closer behavior to Levy Walk model than classical RWP model, inferring better suitability for mobility simulation in future cellular networks that have ever-smaller cells than the classical RWP. Despite similar behaviors, the authors note that the statistical differences between proposed RWP and Levy Walk models cannot be considered insignificant. For example, the transition length cannot be modeled by inverse power-law distribution as in Levy walk model, because its mean will be infinite causing analytical problems such as infinite number of expected handovers in a movement period.

Use case: The two important parameters for studying cellular mobility are handover rate, which refers to number of handovers per unit time, and sojourn time, which is the time that a mobile node resides in a cell. These metrics are analyzed through proposed RWP model application in cellular networks modeled by hexagonal cell and Poisson-Voronoi tessellation. In hexagonal network, the cells are modeled as hexagons with BSs in the center, whereas in Poisson-Voronoi network, the BSs are PPP distributed in 2D plane and serve mobile users located within its Voronoi cell.

The *handover rate* is given by division of expected number of handovers during one movement period by the corresponding time, where the expected number of handovers is found by averaging over the spatial distribution of target waypoint and area covered by neighboring cells in hexagonal model or Poisson-Voronoi tessellation distribution in the corresponding network. The resulting expression shows that the handover rate is inversely proportional to square root of cell size in both scenarios.

The analytical and simulation results for number of handovers were shown to match, except the approximate analytical result in hexagonal network. When Poisson-Voronoi, hexagonal exact and approximate handover rates were compared, the latter model underestimated the rate. Lastly, number of handovers calculated analytically from hexagonal and Poisson Voronoi models as well as estimated from simulation of real-world data of macro-BS deployment in a cellular network were compared. The proposed Poisson-Voronoi model is shown to lie very close real-world data simulation result and exact analytic result for hexagonal network model, whereas the approximate analytic result in hexagonal model underestimated number of handovers. It can be concluded that Poisson Voronoi model seems to be as accurate yet more tractable than exact hexagonal model.

The *sojourn time* is given by averaging expected transition time (eliminating pause time) and probability distribution of cells within coverage area. For hexagonal model with constant velocity, the sojourn time is proportional to the square root of cell size, which is reciprocal to handover rate. The comparison of upper and lower bounds for sojourn time in hexagonal network as well as its counterpart in Poisson-Voronoi tessellation revealed that these bounds are quite tight, whereas Poisson-Voronoi model yields smaller sojourn time than hexagonal upper bound.

Pros/cons: In conclusion, the study revealed that Poisson-Voronoi model is about as accurate in terms of mobility evaluation as hexagonal model predicting higher handover rate and lower sojourn time. The proposed RWP model can be further extended by taking into consideration other important mobility characteristics such as temporal and spatial dependency of the mobility pattern. The authors also emphasize its suitability in performance evaluation of various wireless protocols through application of the tractable model in theory or simulation.

4.4.2.2 Measurement and Modeling of Video Watching Time in a Large-Scale Internet VoD System [44]

As video content starts to constitute major part of mobile data traffic (predicted to account for 74% of cellular traffic in 2024), it is important to understand and assess its main characteristics. Therefore, video watching behavior estimated and evaluated in [44] will be extremely useful for designing future on-the-fly data access model discussed in Section 3.11.2 as it is an important metrics for investigating user watching behavior and therefore can help to describe cellular data usage.

Main idea: In this paper, the authors perform large-scale measurement of user watching data on one of the most popular commercial Internet Video-on-Demand (VoD) systems in China over three weeks period in order to describe watching time distribution and its correlation with different video features. Performance evaluation of the proposed model versus common data-mining methods in terms of normalized mean square and absolute errors has demonstrated its feasibility and accuracy.

Methodology: The measurement data used for video watching time analysis is provided by PPLive from their log servers that collected video sessions logs from PPLive client program installed on user devices. Each video log contained its type, length and popularity and the ones with playback interruption were excluded from analysis in order to focus on clients' natural watching behavior.

Performance metrics: The two metrics for the proposed analysis are video watching time and watching finish ratio defined as watching time divided by video length. Also, video's system workload is defined as product of mean watching time and its popularity. All the performance metrics under study are averaged per each video session. The correlation between these metrics and other video features is calculated using Pearson Correlation Coefficient (PCC), whereas the goodness of fit for hypothesized distribution is quantified using Kolmogorov-Smirnov(K-S) distance. Different models' accuracies are compared in terms of normalized mean square and absolute errors (NMSE and NMAE).

Key findings: *Type-based* mean model assumes that videos of the same type have the same mean watching finish ratio. When compared to the baseline global mean model, which does not consider video type, in terms of NMSE and NMAE, the type-based model shows considerable improvement suggesting strong dependency between video type and its watching time, finish time ratio and system workload. Skew-normal distribution provides a good fit for watching finish ratio of for cartoon and drama serial, info, show, sports, and game videos, whereas movie and news videos do not fit that model very well showing relatively large K-S distances.

Video duration has been shown to affect average finishing ratio linearly with late-drama and sports video types indicating the highest and lowest correlation coefficients, respectively. Type-duration model evaluation has shown that user's video watching finish ratio linearly decreases with video length. Also, different video types experience significantly different user watching finish ratios, even if their video length is the same. The watching time under this model has shown that when video length is below a certain threshold, a longer video usually has longer user watching time, whereas when video length exceeds the threshold, longer video has shorter user watching time. Overall, the inclusion of video duration information further improved the model accuracy for all performance metrics in terms of NMSE and NMAE values.

Video popularity analysis has revealed that watching finish ratios of most video types increase logarithmically with their popularity, except for news video type where watching times have indicated to decrease logarithmically with their popularity. Comparison of type-popularity model with the type-based mean model NMSE/NMAE have shown minor improvements in watching finish ratio and time but not for system workload, which is why the latter will not include effect of video popularity in final derivations.

Using the above observations, authors derive *integrated model* to characterize correlation of watching finish ratio of specific video type to both video length and video popularity in terms of multiple linear regression model. Similarly, the integrated model for watching time and system workload is derived. NMAE/NMSE performance analysis of both has yielded considerable improvements as compared to previous analytical models considering single variable.

Lastly, the performance of the integrated model is compared to that of two commonly used data mining methods: regression tree and random forests. The resulting NMAE/NMSE values have shown slight difference (0.01) in favor of mining methods indicating good accuracy of proposed integrated mathematical model within data-mining-based models. The proposed models have simple and elegant forms and can be easily applied in various scenarios including user engagement information that are necessary for designing future cellular network traffic and workload.

4.4.3 Blockchain for decentralized mobile data access

4.4.3.1 Blockchain Radio Access Network (B-RAN): Towards Decentralized Secure Radio Access Paradigm [45]

Main idea: The utility of blockchain radio access network (B-RAN) model proposed in [45] is investigated by designing decentralized, self-organized and scalable framework capable of network access control and authentication among inherently trustless network nodes followed by evaluation of such performance metrics as throughput and latency. Some motivations of using blockchain technology for network access include provision of adequate economic stimulus to network participants, minimization of overhead costs associated with centralized access mechanisms and establishment of trust among participating nodes for improved cooperation.

System model: Under the proposed scheme, wireless nodes can act both as access providers or users forming self-organized open market for exchanging network related services reaching agreement through smart contracts. When user equipment (UE) acting as access consumer requests service from access point (AP) acting as service provider, they first need to reach agreement on availability of their assets, e.g., spectrum resources from AP side and UE balance as indicated on Figure 4-4.1. Once consensus is achieved and the contract is signed by both parties, it is sent to mining network for verification of signers' sufficient assets. Then, it is aggregated together with other transactions in the newly created block to be added and confirmed to blockchain. Once newly verifying blocks are built on top of it, the block with pending smart contract will be posted on-chain and the AP-UE pair will exchange their assets (i.e., AP gets paid and UE obtains time limited access to service).

Due to high power consumption, traditional Proof-of-Work consensus mechanism is replaced by Proof-of-Device (PoD) method that allows the devices to vote on the new generated blocks based on their unique identifiers, e.g., the international mobile equipment identity (IMEI) for UEs and the cell global identity (CGI) for BSs. This consensus mechanism is more resilient to attacks since it is very unlikely that a single party will own more than half of the devices required to control the whole blockchain. The risk of double-spending attack was shown to increase with shorter delay requiring fewer confirmations for a new block. Less delay is desirable in the proposed protocol, therefore finding optimal degree point between security and latency is essential in B-RAN.

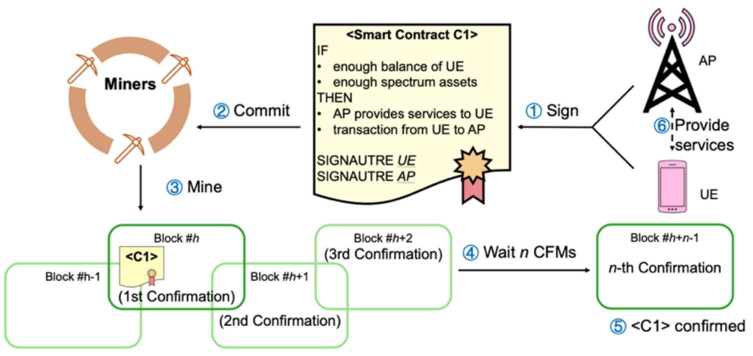


Figure 4.4-1 AP-UE access establishment in B-RAN [45]

Performance metrics: The B-RAN framework is evaluated in terms of different performance metrics such as throughput, latency and probability of successful double spending attack obtained from simulation of randomly located APs and UEs that follow asset exchange process described above.

Key findings: Throughput versus traffic load was analyzed for 5 different scenarios: centralized scheme, multi-operator network, selfish and blockchain access. The centralized scheme, where a single center with high computational power is handling the network, outperforms other schemes in terms of served data rate although in reality this model is unrealistic. Blockchain scenario is the next highest throughput scenario outperforming the remaining schemes owing to its decentralization and network-level trust. Another analysis has yielded that larger block size is associated with larger throughput but is marginally limited due to specific traffic load and the network size. Latency tends to increase with number of confirmations however is not affected by traffic load. Probability of successful double spending is inversely proportional to latency indicating a trade-off between the two.

Pros/Cons: Blockchain serves as a promising candidate for managing network assets exchange in decentralized self-organized radio access networks, however challenges such as mining energy efficiency, block spreading, scalability, latency, reliability and payment scheme choice are to be addressed to enable blockchain-based decentralized networking.

4.4.4 Blockchain scalability solution: Payment channels

4.4.4.1 The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments [46]

Main idea: Lightning network (LN) [46] is one of the methods based on payment channels designed for proliferating Bitcoin network to address blockchain scalability issue mentioned in Section 3.10.3. Its main idea is to reduce on-chain transactions through establishment of trading channel between payer and payee, where multiple off-chain transactions are performed at a higher rate that are later aggregated and posted on-chain at a deferred date.

System model: There are three main stages in the proposed framework: payment channel establishment, off-chain trading through channel and their closure. When two nodes want to open payment channel, they need to deposit a particular number of coins that should be greater than the total amount of planned transaction in the channel. Once this action is posted on-chain, bi-directional payment channel between two parties is opened and the two nodes can start exchanging coins with each other. In case one of them cheats, that party will be penalized by withdrawal of all funds from the channel that will be sent to the second node. During channel closure, the amount of tokens on both sides of the channel are submitted to the main chain to update balance of both participants. Therefore, the transactions on the payment channel are aggregated in balance updates and the process produces only 2 on-chain transactions. Such method drastically increases transactions throughput of blockchain network while keeping the block size unchanged.

The authors extend this algorithm to a Payment Channel Network (PCN) where two parties can conduct off-chain transaction without having direct payment channel between them. Instead, they can transfer off-chain payments through “multi-hop” payment route consisting of multiple trading channels and intermediary nodes. It should be noted that such approach still lacks fully developed routing mechanism that will guarantee PCN availability via appropriate routing mechanism.

Pros/Cons: To conclude, Lightning Network provides elegant solution to blockchain scalability issue. Compared to conventional bitcoin network that would require 24 GB block size to enable 7 billion people making two transactions per day, the proposed model would need 133 MB blocks to connect the same amount of people making 2 payment channels per year. Apart from addressing bitcoin scalability, it can be used in latency sensitive cases allowing almost instant payments as well as to reduce payment fee through enablement of off-chain micropayments. Yet, the protocol has important limitations: both nodes belonging to establishing and maintaining the payment channel have to be simultaneously online. Furthermore, it was noticed that LN has low success rate for high value transactions leaving room for protocol improvement.

4.4.5 Blockchain for mobile service trading

4.4.5.1 Contract-less Mobile Data Access Beyond 5G: Fully-decentralized, high-throughput and anonymous asset trading over the Blockchain [26]

Main idea: Mobile data access and trading model for future cellular networks based on blockchain paradigm is proposed in [26]. Motivated by increasing complexity of 5G and beyond networks management and rapid growth of MEC/RAN integration use cases, the study focuses on the enablement of decentralized high-throughput exchange of network- and blockchain-level assets among network participants. It addresses blockchain challenges, namely consensus, scalability (in terms of transactions throughout) and anonymity, which are of substantial importance in the scope of mobile data access model in 5G and beyond, integrating payment channels, smart contracts, Delegated-Proof-of-Stake (DPoS) consensus algorithm, novel payment relay and mixing services. Moreover, due to its prevalence in future mobile data traffic pattern (according to predictions), wireless video content sharing and trading are considered in combination with previously mentioned components resulting in a fully personalized, user-centric yet confidential and anonymous mobile video access paradigm.

System model: The deployment scenario of proposed framework can be divided into three stages discussed in Section 3.11.2: service discovery and pairing, service negotiation and parameterization, online service management and charging. The service domains include user, control and blockchain related interactions. User domain spans video delivery and corresponding payment issuance/reception. Control domain includes service control functionality, e.g. processes relevant to charging or payment aggregation, whereas blockchain domain holds the public ledger and corresponding interactions. Further details of the proposed protocols are omitted for the sake of conciseness, only providing overview of crucial concepts in the context of this discussion.

Payment relay is the component that allows the proposed cryptocurrency platform to have sufficient off-chain transaction throughput and facilitates trust establishment between user-server pair acting as an escrow between them. Leveraging the concept of payment channels, it allows to aggregate payments between payer-payee pair in a similar way. However, unlike payment channel that aggregates micropayments per user-server pair session and thus requires two on-chain transactions per new mobile video session, the payment relay performs additional on-chain transactions corresponding to inbound(U-R) and outbound(R-S) channel balance updates occurring during that video session (PSU). Although this leads to increase in number of on-chain transactions, such model is more trustworthy for service providers as it allows timely update of outbound channel balance as compared to balance update in payment channel that will depend on video session length. In addition, payment relay can aggregate pending micropayments to a particular server originating from different users before agreed deferred time denoted as relay delay. Therefore, under optimal parametrization of payment channel balance and relay delay, the proposed payment relay framework outperforms P2P payment channel model.

It is further extended to a variant without outbound channel balance updates (SU) as well without both outbound and inbound payment channel updates (U). These two provide even better performance in terms of supported transaction throughput, however, require more trust between payees and payers. It is also possible to include the effect of simultaneous presence of inbound and outbound channel towards network participant, such that both inbound and outbound transactions are aggregated and result in 1 on-chain balance update transaction instead of 2 updates per each channel.

Comprehensive simulation results that were obtained from deployment of different scenarios discussed above on top of a multi-tier heterogeneous mobile data network consisting of 6.5 billion potential users, 0.5 billion server addresses in all server scenario and 1k cellular MNOs. Number of new video sessions are Poisson distributed, whereas video duration is exponentially distributed. The micropayments are made per specific period of video time, e.g., user pays per m seconds of video. Transaction throughput was quantified and compared against number of users, relay delay, user balance credit, network relay ratio and new sessions per second.

Key findings: The main observations are as follows. *Transactions per second (TPS)* grows linearly with number of users. PSU model has noticeable performance improvement when relay delay approaches mean service time per session. When the outbound balance credit is sufficiently large, PSU outperforms P2P payment channel model reaching 12-times decrease in TPS for the same number of users. Obviously, the proposed PSU, SU and U variants exploit the credit balance better than conventional models.

Analysis of *TPS versus relay delay* yielded that the latter only affects PSU variants decreasing TPS linearly. It should also be noted that higher relay delay demotivates the servers to trust the relay and affects the liquidity of servers. Therefore, it is important to optimize it.

TPS graphs against UE balance indicated 4-fold reduction in TPS for PSU variant opposed to P2P channel model when relay delay is equal to 6 hours. Increasing this value to 120 hours leads to 8-fold performance gain in favor of PSU variant. All payment frameworks except the one without channel balance updates attain flat performance when the balance credit is equal to 10-15 hours.

Network relay ratio did not affect conventional models without payment relay, whereas PSU and SU variants had slight increase in TPS. Conversely, in U variant TPS decreased because users act as servers consuming their balance more slowly. Lastly, TPS is directly proportional to the number of new sessions per second in traditional variants without payment relay, whereas the ones with payment relay scale more slowly due to one hop star topology with relay.

Future work: In the context of the proposed solution, future research directions include extension of the proposed framework evaluation to mathematical analysis and its validation through the simulation results from its full-scale implementation.

4.4.6 Discussion of current SotA and future research directions

The discussed SoA solutions are summarized in Table 4.4-1 in terms of used methodology and KPIs.

Analysis of solutions and corresponding performance evaluations carried out in the scope of cellular networks (1-3) has shown the vast usage of mathematical tools related to stochastic processes. Poisson Point Process was the most commonly used stochastic geometry tool for modelling cellular networks, thus it will be referred to in designing a baseline system model for future mobile networks.

The study of video watching behavior used regression analysis to evaluate collected data (4). Since the study dates back to 2013, newer data sets need to be collected for estimation of video related parameters that are relevant in nowadays and future mobile video streaming.

Observing methodological tools for blockchain related solutions (5-7), it is clear that these frameworks mostly rely on algorithm descriptions and simulations, thus requiring mathematical analysis similar to the used tools described in 1-4, especially if these approaches are to be used on top of heterogeneous mobile networks. Therefore, future work will be focused on analytical framework development utilizing such mathematical tools as stochastic geometry and queuing theory in order to study the feasibility and scalability of the proposed mobile video trading scheme.

Table 4.4-1 Summary of SotA solutions, corresponding tools and KPIs

#	Scope	Study	Methodological tools	Key performance metrics
1	Network topology	[41]	Poisson Point Process, Rayleigh fading, hexagonal network model	Coverage probability, average load per tier, average data rate, fraction of active users
2	Network topology	[42]	Poisson Point Process, Kolmogorov-Smirnov test, hexagonal network model	SINR, SIR, energy efficiency
3	User mobility	[43]	RWP mobility model, Poisson Point Process, Rayleigh distribution, Poisson-Voronoi tessellation, hexagonal network model	Handover rate, sojourn time, transition time, pause time, direction switch rate, spatial node distribution
4	Video watching behavior	[44]	Pearson correlation coefficient, Kolmogorov-Smirnov distance, normalized mean square/absolute error, R, regression tree, random forests data mining models	Video watching finish ratio, video watching time, system workload
5	Blockchain RAN	[45]	Algorithmic framework, simulation	Network throughput, latency, block size, probability of successful double spending
6	Payment Channel	[46]	Algorithmic framework	Block size, payment fee
7	Blockchain backed mobile service trading	[26]	Algorithmic framework, simulation	Transaction throughput, relay delay, channel balance credit

5 Methodological tools

The purpose of this section is to present methodological tools that are relevant to WP 2. These tools include platforms or testbeds available at partner universities (e.g. POLITO) as well as mathematical frameworks such as stochastic geometry, queuing theory and machine learning. These are defined in terms of their building blocks, capabilities, and relevance to the project objectives.

5.1 POLITO testbeds/tools relevant to MEC

The MEC testbed at POLITO consists of two interconnected blocks as shown in Figure 5.1-1, namely the edge host and the mobile terminal. The mobile terminal consumes the services offered by the mobile applications. The edge host provides the computation and mobile connectivity to the mobile applications, such that the services are readily available for the mobile terminal. The connectivity between the edge host and the mobile terminal is provided through a heterogeneous RAN formed by integrating 3GPP LTE and IEEE 802.11p. Both 3GPP LTE and IEEE 802.11p technologies are implemented using Software Defined Radio (SDR).

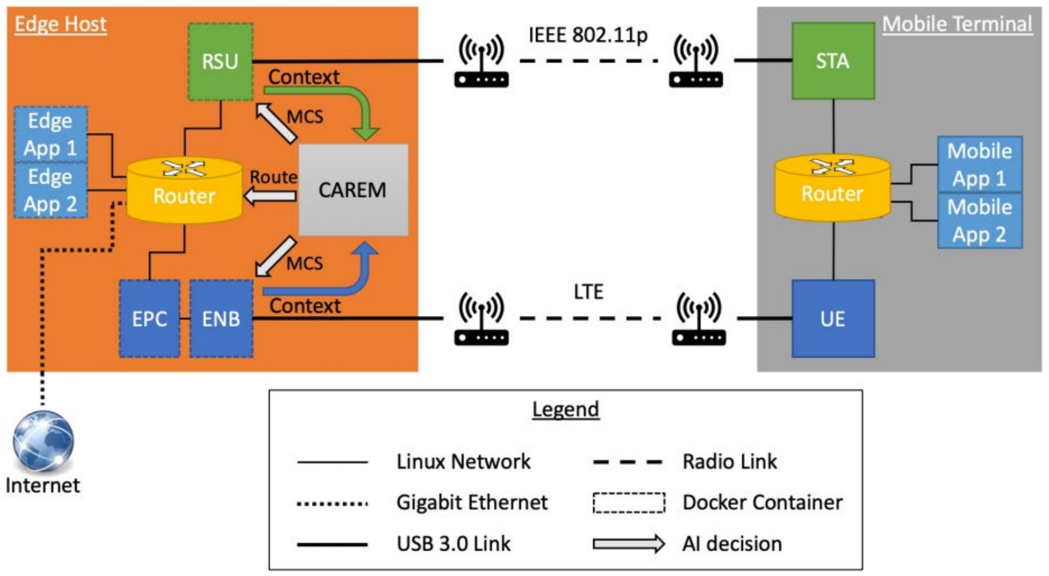


Figure 5.1-1 MEC Testbed at POLITO

The LTE RAN built based on srsLTE, an open-source SDR LTE stack implementation offering EPC, eNodeB, and UE applications. The LTE RAN supports up to 20 MHz bandwidth channels and transmission modes 1 through 4, all using FDD configuration, and it is compatible with LTE release 9. On the other hand, the 802.11p transceiver is implemented using the WiME project's GNU Radio flowgraph, and it is compatible with commercial IEEE 802.11p products.

The critical module of the edge host is the CAREM framework proposed in [12], responsible for controlling the operation of the heterogeneous RAN. The CAREM framework has an algorithm responsible for periodically selecting the appropriate link and the MCS to be used by the link chosen for downlink packet transmission. Both SDR solutions expose a tun/tap interface to which an IP address is allocated to communicate with the host operating system network stack. To those exposed interfaces, a router is connected to steer traffic over radio links, the host applications, and the internet, according to the link selected by CAREM.

The srsLTE eNodeB has been modified to run a dedicated thread responsible for listening and applying the MCS selected by the CAREM framework to the uplink and downlink transmission of a particular UE. An XMLRPC server block has been added to the 802.11p GNU Radio flowgraph for the same purpose, exposing a remote procedure call interface to dynamically set the MCS to be used. The SDR applications also connect the environment variables such as mean and variance of SNR, and buffer occupancy status at the MAC layer.

5.2 Stochastic geometry for modelling and analysis of future cellular networks

Stochastic geometry (SG), also referred to as geometric probability, is a mathematical tool dealing with random spatial patterns. SG aims to characterize random collection of point in 1D, 2D and higher dimension planes. Random nodes in this framework are usually described as point processes (PP). Therefore, point process theory is an important sub-field of stochastic geometry and definitions of the most common point processes are provided in Section 5.2.1 based on their detailed discussion in [47]. Their typical application scenarios and performance metrics are presented in Section 5.2.2 and Section 5.2.3, respectively. The last subsection 5.2.4 briefly discusses the relevance of stochastic geometry for modelling and analysis of future cellular networks.

5.2.1 Point processes

Point process is simply a collection of random point lying on some space. One-, two- and d-dimensional Euclidean spaces are denoted as R, R^2 and R^d and in the context of wireless networks 2D space is used most. The most frequently deployed PPs in wireless communications are Poisson point process (PPP), binomial point process (BPP), hard core point process (HCPP) and Poisson cluster process (PCP).

Homogeneous or uniform PPP is a point process on R^d with intensity λ, such that number of points inside every compact set B N(B) a Poisson distributed random variable with mean equal to $\lambda * |B|$. Moreover, the number of points in disjoint sets B_1, B_2, etc., are independent random variables. General PPP on R^d with intensity measure Λ is a PP where every compact set $B \subset R^d$ contains Poisson distributed random variable with Λ(B) representing its mean on a specific set. Similarly to uniform PPP, the number of points are independent given that the sets B are disjoint. The examples of homogeneous(left) and non-uniform(right) PPP models are provided on Figure 5.2-1. It can be observed that unlike uniform PPP, the general PPP shows on the left shows variation in intensity with repetitive patter across x-axis and decreasing along y-axis.

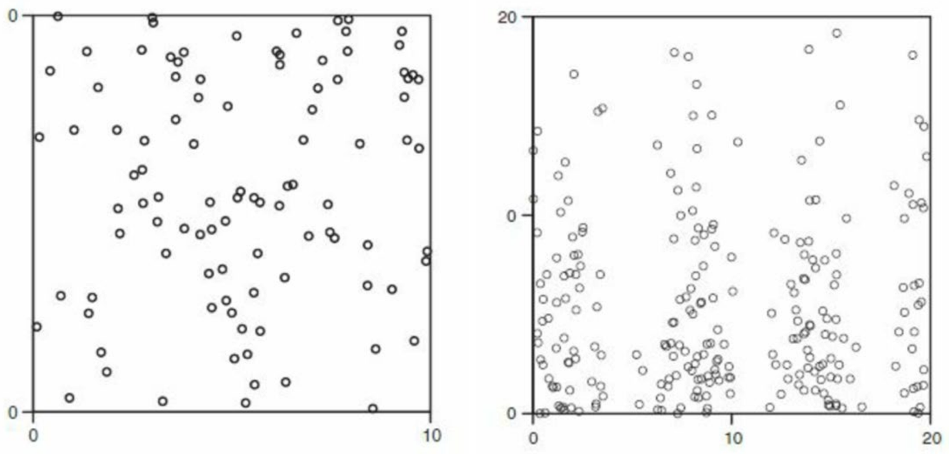


Figure 5.2-1 Realizations of uniform (λ = 1) and non-uniform (λ(x, y)=2exp(−y/6)(1 + sin x)) PPP [47]

BPP is another important class of point process that model random spatial pattern by placing fixed number of N identically and independently distributed points on a compact set $B \subset R^d$. In uniform BPP, the point process $\Phi = \{x_1, ..., x_n\} \subset R$ is seen as random vector uniformly distributed on B^n. Conversely, in general BPP, each point has a probability distribution function, and the intensity measure of the process is given by $\Lambda(A) = n\int_A f(x)dx$. The number of points in Borel subsets of B is binomially distributed with parameters n =

Φ(B), p = |A|/|B| and intensity measure $\Lambda(A)=n\frac{|A\cap B|}{|W|}$, where A represents Borel subset. Contrary to PPP, the number of points in Φ(A) and Φ(B/A) are not independent. However, the important implication is that conditional PPP with fixed number of points N is equivalent to BPP with N nodes.

HCPP is a repulsive point process where no two points at a distance closer than predefined hard core parameter r_h can exist at the same time such that $\|x_i - x_j\| \geq r_h$ must be satisfied for all x_i, x_j belonging to PP where $i \neq j$.

PCP model is used to characterize clustered random spatial distributions. It is formed from a parent PPP such that each x_i point belonging to PP is replaced with cluster of points M_i that contains identically and independently distributed points in spatial domain.

5.2.2 Usage scenarios

PPP is typically applied when modelling a network consisting of infinitely many random and independent nodes coexisting in a finite or infinite service area. For example, users belonging to cellular networks or nodes in large-scale ad hoc networks are frequently represented by PPP providing tractable yet accurate analytical framework for evaluation of important performance metrics. Even when applied to provide approximation of planned infrastructure-based networks and coordinated spectrum access networks, PPP allows tight bounds for performance parameters.

In scenarios with known number of nodes and finite service area, **BPP** is preferred for network abstraction. This PP can be applied to the case where known number of sensors are randomly distributed in a specific area, e.g. in military applications.

HCPP is found to be useful in modeling scenarios with minimum spatial separation requirements caused by physical limitations, specific network planning or MAC layer behavior (e.g., CSMA CW).

Recently, **PCP** that is suitable for analytically describing network nodes clustered according to specific behavior has been proposed in designing analytical frameworks of HetNets (e.g., in [50]).

5.2.3 Performance metrics

Stochastic geometry provides a powerful analytical framework for evaluation of various important performance parameters in heterogeneous wireless networks. Ability to derive interference statistics is perhaps one of the most meaningful implication of stochastic geometry. It allows to derive other essential metrics such as coverage probability, SINR, transmission capacity, spectral efficiency, frequency reuse, energy efficiency and many more in terms of design parameters such as transmit power and channels.

5.2.4 Relevance

According to [49], stochastic geometry is the only available mathematical tool that provides a rigorous yet tractable analytical approach to the modeling, analysis, and design of heterogeneous and cognitive multi-tier cellular networks. Although the baseline SG models are very simplified, most of the research on cellular networks performance evaluation builds up on this analytical paradigm.

As the cellular networks become more and more heterogeneous and congested, their topologies start to converge to the ones modeled via point processes. For example, locations of actual LTE base stations can be approximated by PPP of the same density as indicated on Figure 5.2-2. According to [48], the actual deployment lies between two extreme cases of perfectly regular grid (hexagonal or square) model and completely random PPP model. In fact, the latter presents a lower tight bound for coverage probability in such topology and the discrepancy between the actual and PPP models is decreases as the SINR threshold increases meaning that under specific assumptions one can rely on SG for modelling HCN.

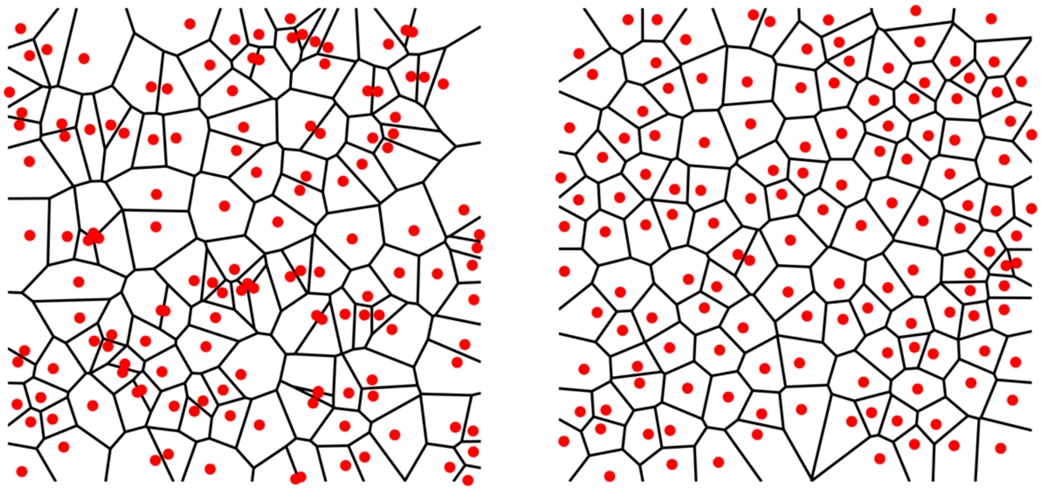


Figure 5.2-2 PPP(left) and Actual LTE(right) base stations over 40x40km area [48]

In the scope of this work, SG can be a powerful tool for designing and evaluating performance of novel mobile data access models on top 5G and beyond networks enabling simple yet accurate modelling of corresponding cellular topologies. Essentially, elaboration of underlying network topology on top of which the novel data access algorithms are to be built is going to be a first step in the system model attempt. The resulting topology and performance metrics can be further used to describe user mobility that will eventually help us to evaluate feasibility of the proposed mobile data access model.

5.3 Queuing theory for modelling and analysis of blockchain transactions

Queuing theory is a field of applied probability that deals with the analysis of waiting lines. Queues exist almost everywhere in our daily lives hence applications of queuing theory span a wide range of cases such as analysis of lines in a supermarket or computer networks. The main concepts and terms are defined in Section 5.3.1, while Section 5.3.2 briefly explains Kendall notation. Performance measures and relevance of this tool to current work are presented in Sections 5.3.3 and 5.3.4, respectively, based on [51].

5.3.1 Basic terminology

The three main components that constitute the subject of queuing theory are customers, queues and servers. **Customers** are generated by an input source, i.e. customer or calling population having infinite or finite size, with specific distribution that describes times between customer arrivals in the queue. **Queue** is the waiting line joined by customers or incoming requests upon arrival to the system. It can be of finite or infinite length depending on modeled situation. The customers wait in the queue until they are selected for service by a **server** or service mechanism that can be single or multiple in a system. Basic queueing system consisting of customers, queue and servers is depicted on Figure 5.3-1.

The way customers are chosen from queue for further processing by server depends on **queueing discipline**. The most common queuing discipline is First Come First Server (FCFS or FIFO) where customers who came earlier are served in the order they landed the queue, e.g., buyers in supermarkets. Other disciplines include Last Come First Served (LCFS or LIFO), Random Service (RS or SIRO), Round Robin, Priority with and without Preemption. LCFS is self-explanatory – the customers in the tail(back) of the queue are served earlier than the ones in the head (front) and such discipline is typically encountered in computer stacks. Priority disciplines differentiate between classes of customers, e.g, distinguishing between first, business and economy class passengers on a plane when seating them.

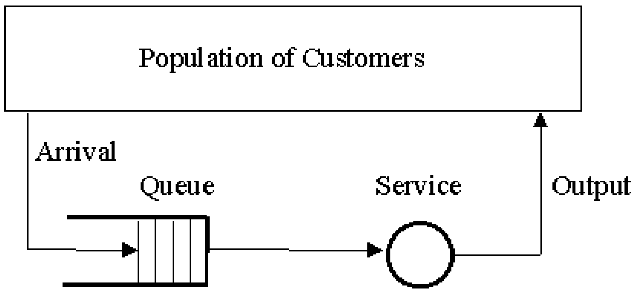


Figure 5.3-1 Simple representation of queueing model [51]

5.3.2 Kendall's notation

There is a wide variety of possible queuing systems depending on distribution of interarrival and service times, number of servers, capacity of system and serving discipline. In order to distinguish these systems, they are classified using Kendall's notation that describes queuing systems in terms of six characteristics denoted as A / B / m / K / n/ D, where each symbol refers to the following:

A – distribution of interarrival times,

B – distribution of service times,

m – number of servers,

K – system capacity, i.e. maximum number of customers in the system,

n – population size, i.e. number of possible customers or input sources,

D –queueing discipline.

The main patterns for interarrival and service times are decrypted as follows:

M – Markovian (Poisson) arrival process with exponential distribution of service times,

E_k – Erlang distribution of intervals or service duration with k phases,

D – deterministic arrivals or constant service times.

G – general distribution with known mean and variance.

Moreover, if the population size and the capacity are infinite, queuing discipline is FIFO, then corresponding symbols are omitted in Kendall's notation. Typical queueing system M/M/1 denotes such model where customers arrive from an infinite customer population according to Poisson distribution to one exponential server with FIFO discipline.

5.3.3 Performance measures

Queueing theory helps to identify the statistical properties of the queue using information about customer arrival and service processes. The time between incoming requests is called **interarrival times**, whereas the interval needed to process that request at server is called **service time**. These are commonly handled as independent random variables with distributions mentioned in Section 5.3.2.

The performance metrics of interest include queueing time, number of customers in the system, number of waiting customers, server utilization, mean response time, idle and busy times of the service mechanism. The amount of time which a customer waits in the queue for is called the **queueing time**. **Number of customers** in the system includes number of waiting (in a queue) and served customers. **Utilization** is the fraction of time that the server is busy, whereas **mean response time** is the average time spent by customer in the system. The latter us typically found using Little's law.

Little's law relates average number of customers(N) in the queueing system with average customer arrival rate(λ) and mean response time or average service time(T), such that **N = λT** in steady state. This theorem holds for different queues including G/G/1 and even service disciplines other than FCFS.

5.3.4 Relevance

In the scope of blockchain based service charging for future mobile data access models, queueing theory can be found useful in modelling the incoming transactions in the system as arriving requests to the queue, whereas the amount of time it takes to process these transactions can be referred to as service time. For example, authors in [52] proposed to model transaction-confirmation process in Bitcoin network as queueing system with batch service $M/G^B/1$, where batch service refers to the ability of server to process B customers at the same time. Using joint distribution transactions number in the system and the elapsed service time, they were able to derive mean transaction-confirmation time.

Similarly, it is planned to model transactions generated in the proposed blockchain based mobile data access model as arrival process, whereas the payment relay will be treated as batch server. This will help us to evaluate systems transactions throughput by finding out number of customers in the system.

5.4 CTTC testbeds/tools relevant to MEC

With the increase in service verticals in 5G and beyond and instability in network requirements making static decisions on providing reliable service to the user becomes a hectic process. Making dynamic decisions instantly based on changing requirements of users is important with the help of AI/ML. AI/ML algorithms like federated and reinforcement learning helps to have a distributed control over the network without causing much of pressure on single server/cloud, but it does have some disadvantage when the users are moving as seen in Section 4.2.1 as task offloading drains may drain the network traffic and cause delay in the service to be achieved. More survey is yet to be done to decide on using appropriate AI/ML algorithms.

5.5 Machine learning tools/platforms/IDE for MEC

Supervised learning*:* Supervised learning is a machine learning technique that takes labeled data as input during the training phase. The labeled data is nothing but a collection of data objects (typically as vectors) and their corresponding output (label). During the training phase leveraging the labeled input data, supervise learning learns a model that maps the input data to the target output variable. Once trained, the optimal model predicts the output of those data objects whose label is unknown. Examples of supervised learning algorithms include decision trees, support vector machine, k-nearest neighbor, etc.

Unsupervised learning: Unsupervised learning is a machine learning technique that takes data without labels as input. Instead, it allows the algorithm itself to detect hidden and unknown patterns present in the data. Unsupervised learning uses two main methods called principal component analysis and cluster analysis. When learning is to discover similar groups present in the data, cluster analysis can be used. However, when we need to determine the distribution of data in the space, principal component analysis can be used.

Reinforcement learning: Reinforcement learning is a type of machine learning algorithm with an agent who continuously interacting with the environment and learns to take actions that result in a maximum reward. Reinforcement learning maps the situations into actions in such a way that we get maximum reward. Essentially, it is a closed-loop problem because the learning system's actions influence its subsequent inputs. Moreover, the learner is not told which steps to take, as in supervised learning, but instead must discover which actions yield the most reward by trying them out. In the most exciting and challenging cases, actions may affect the immediate reward and the following situation and, through that, all subsequent rewards.

TensorFlow: TensorFlow is the machine learning framework that Google created and used to design, build, and train deep learning models. TensorFlow libraries can be used to perform numerical computations through data flow graphs. In these graphs, nodes represent mathematical operations, while the edges represent the data, which usually are multidimensional data arrays or tensors that are communicated between these edges. In addition to deep learning models, TensorFlow also has support for other machine learning models.

5.6 Machine learning tools/platforms/IDE for MEC

Machine Learning (ML) applications have overgrown their implementation popularity in most of the industrial and academic domains over the past years and currently are proved to be successfully dealing with a plethora of challenging problems from physical layer channel equalization to network and application layers add-ons. The ML tools have been shown successfully applied in many MEC-related domains such as content caching, adaptive bitrate streaming, federated learning, content popularity estimation, mobility prediction, and clustering. The neural network (NN), as a particular example of ML technique, consists of a vast number of nonlinear units – neurons and dense synaptic connection links between them, which makes the architecture of a NN – a universal function approximator. Eventually, more and more challenging tasks appear in the domain, which requires exceptional computational approaches, that NNs can easily provide. Consequently, two main forces currently keep the development of artificial neural networks steady and triumphant: a variety of available datasets from highly diverse domains and low-cost access to computational devices, that can process these data [53]. The major types of NNs that are currently flourishing in a variety of industrial tasks are feedforward NN (FNN), recurrent NN (RNN), and convolutional neural networks (CNN) [54].

5.6.1 Feed Forward NN

One of the simplest but prevalent types of NNs is the feedforward neural network (FNN), which has been exhaustedly studied and analyzed so far. The simple architecture of FNN consists of three key parts: the input layer, the hidden layer, and the output layer, which are all consists of bunch of activation neurons or perceptrons. As long as the perceptrons, are connected unidirectionally, creating coupling from the left side to the right, the network is called feedforward and can be defined by determining the state of a network:

$$\begin{aligned} \mathbf{x} &= f_1\left(W^{in}\mathbf{u}\right), \\ \mathbf{y} &= f_2(W^{out}\mathbf{x}), \end{aligned} \tag{1}$$

where f_1 and f_2 are perceptron activation functions, the W^{in} and W^{out} are connection weights matrices, **u** is the input information vector, while **x** is a state of the network`s hidden layer and **y** is the calculated output. In order to build the model for a particular problem, the main task here is to find the best, most suitable, values of $W^{in}$ and $W^{out}$ weights of the network`s architecture, which is usually referred to as training [54]. The FNNs are exceptionally simple and straightforward ML tools, that can be successfully applied for deep classification tasks or even regression predictive tasks. On the other, the memoryless property of the FNN has led to the introduction of the recurrent type of NN, which can easily handle long sequences and are highly efficient in prediction tasks.

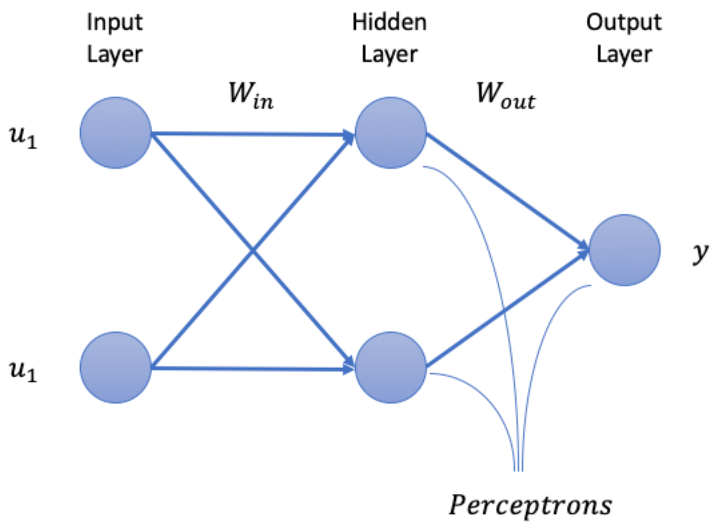


Figure 5.6-1 A three layers feedforward neural network (FNN). Input layer ***u*** *=(u_1 , u_2), a hidden layer with perceptrons and the output layer* ***y****. The W^{in} and W^{out} are the input and output weight matrices accordingly*

The FNNs are typically used as a function approximation tools for classification and regression tasks. In [32] the FNNs have been found successfully applied for estimating the Q value in Q-learning algorithms.

5.6.2 Recurrent Neural Networks

Opposite to the feedforward neural networks, where the computation procedure is implemented statically and without memory utilization, the recurrent NNs are designed to use memory to accumulate the essential processing information from previous steps. Another distinctive feature of RNNs is feedback connectivity. The network`s hidden layer consists of densely interconnected individual neurons and therefore inside the recurrent type of the networks the information flow is not done in one direction, as it is in FNNs. Furthermore, the data is undergone the circulation within the hidden layer, which is typically called the recurrent layer, and a simple recurrent network has at least one synaptic connectivity cycle – loop (see Figure 5.6-2). Combining these essential recurrent properties with high network flexibility, this type of network is an extremely powerful tool for dynamic sequential processing applications [55].

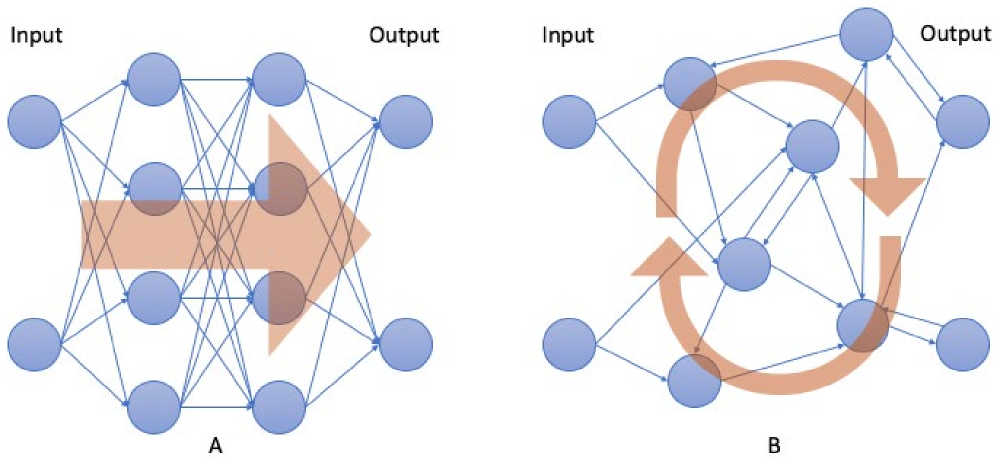


Figure 5.6-2 The architecture of an (A) feedforward and (B) recurrent types of neural network

Being very well known for the predictive ability of the model and highly relaxed complexity level, the recurrent type of the NN – called echo state network – has been applied in [36] for pro-active caching to estimate the future content distribution and users` mobility pattern.

5.6.3 Long short-term memory

One of the currently highly popular approaches to tackle sequential problems is the application of short-term memory (LSTM) networks. Generally, the RNNs can store information from previous in the form of activations, thus the basic architecture is limited by the defined short time window – usually called "short-term memory". On the other hand, the long short-term memory (LSTM) is a special type of recurrent neural network, which can learn the long-term dependencies along with the short-term ones. The general idea of the LSTM design is a combination of a memory cells, that maintain their conditions over the time, and additional gates that control the dataflow inside the cells.

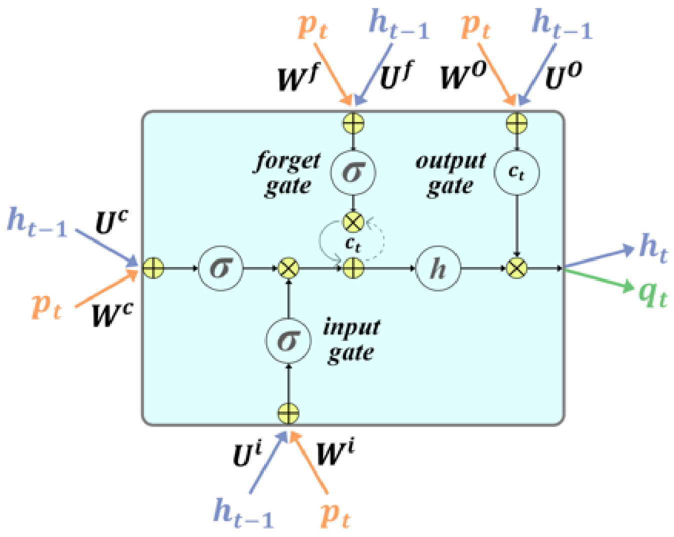


Figure 5.6-3 Detailed structure of the LSTM cell

The design of the LSTM neural network is conceptually similar to the simple RNN network, however, the major difference between their structure is the substitution of the network`s neurons with the LSTM units. A single LSTM unit is composed of a memory cell and a set of multiplication gates: a forget gate, input gate, and output gate. The design is shown in 5.6-3.In that architectural design, the output $\mathbf{q}_t$, as well as the state $\mathbf{h}_t$ and the content $\mathbf{c}_t$ of the cell are defined by the current input $\mathbf{p}_t$, and the previous state $\mathbf{h}_{t-1}$ under the management of these three gates accordingly. The cell`s content, state, and all the outputs of the gates can be calculated by:

$$\mathbf{f}_t = \sigma\left(\mathbf{W}^f\mathbf{p}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f\right),$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}^i\mathbf{p}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i\right),$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o\mathbf{p}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o),$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}^c\mathbf{p}_t + \mathbf{U}^c\mathbf{h}_{t-1} + \mathbf{b}^c\right),$$

$$\mathbf{h}_t = \mathbf{o}_t \tanh \odot(\mathbf{c}_t).$$

here, the outputs of the forget gate, input gate, and output gates are $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t$ respectively. The $\mathbf{c}_t$ parameter depicts the content and $\mathbf{h}_t$ represents the state of the LSTM cell. Typically, the gate activation function σ is a logistic sigmoid function and $\odot$ is the element-wise dot product. The parameters of the LSTM network that required to be optimized during the training procedure are $\mathbf{W}$, $\mathbf{U}$**,** and $\mathbf{b}$**.**

The LSTM modification of the recurrent NNs was found successfully applied for Q value estimation in the Q-learning algorithm [32]. Additionally, the LSTM was applied in [37] for proactive caching in MEC with an upgrade of using the LSTM cells taking into account not only the past sequence, but also the future sequence as an input.

5.6.4 Convolutional Neural Networks

The convolutional neural network has become an essential and paramount technique in the image, video, and natural language processing (NLP) applications. Due to its key model property of extracting the features – the CNNs are able of detecting information in different matrix locations with exceptional accuracy. In the majority of applications, CNNs are used for processing 2D or 3D data, like images or videos. On the other hand, the one-dimensional (1D) CNNs (1D–CNN) are getting popular in applications sequence processing is required [56]. The 1D-CNN are found highly successful in dealing with temporal one-dimensional data such as time series of sensors data, audio signals, and NLP series [57]. Additionally, a single-dimensional convolution layer can be applied as a pre-processing step in most of the signal processing applications for reorganizing the original sequences and extracting higher-level features.

The CNN layer has several hyper-parameters to be optimized: number of filters, or kernels, the size of the kernel, and the activation function, which sometimes can be avoided. The feature extracting is performed by applying a number of different kernels, that represent the hidden layer activation units, to the original data sequence. The kernels with a particularly defined size are sliding on the raw input data and performing numerical multiplication. The simple procedure example can be seen in Figure 5.6-4. The output of the 1D convolution layer is represented by the number of kernels obtained during the transformation.

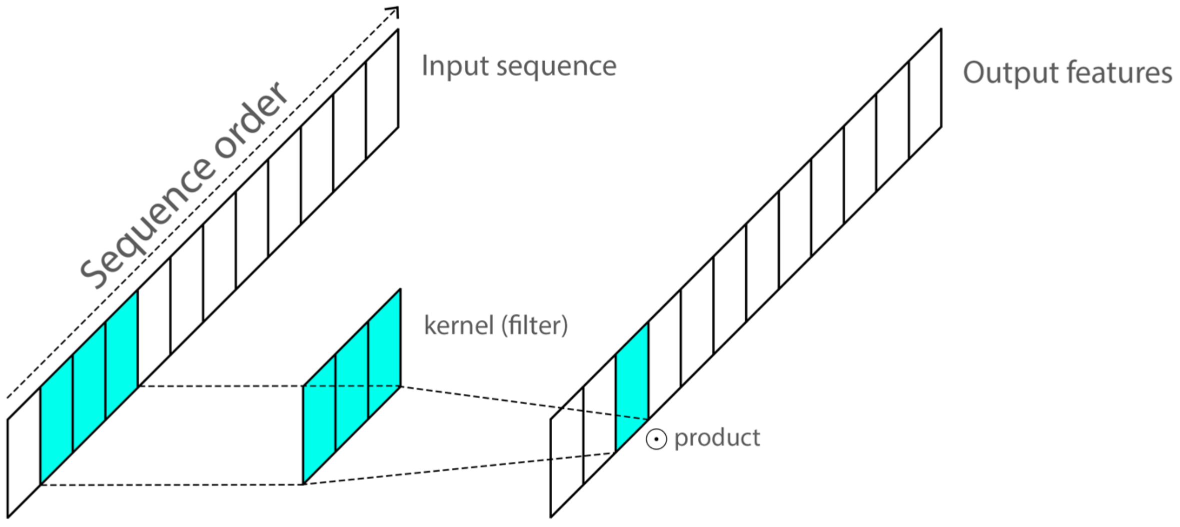


Figure 5.6-4 Convolving process of a 1D-convolution layer

The squashing function that represents the convolution procedure is defined by equation X:

$$x_j^l = f_{act}\left(\sum_i x_j^{l-1} * k_{ij} + b_j^l\right), \tag{1}$$

where x_j^{l-1} is the input vector, k_{ij} represents the i-th trained convolution kernels, x_j^l is the output feature maps, and b_j^l is bias. The layer activation function is represented as f_{act} and is usually set to be a rectified linear unit (ReLU) with function $\max[0, x]$ or leaky rectified linear unit (LeakyReLU) with $\max[\alpha * x, x]$, where α is the negative slope coefficient.

The CNNs have been successfully used in [37], in combination with recurrent NN to extract valuable features from time sequence input and predict the time series requests.

5.6.5 Reinforcement Learning

Another approach that is very often met as an optimal solver in the MEC domain is Reinforcement Learning (RL) techniques, which are typically utilized when there is a lack of available data in the model. RL is defined as learning what to do – by mapping the faced situations to available actions – to maximize the value of the regard signal [58]. Typically, the learner does not know which actions to take but instead has to discover which of the applied actions lead to the most rewarded state by trying them. The trial and error search and delayed reward-feedback are the two key distinguish mechanisms of any RL algorithm.

The major elements that define the RL algorithm are a policy, a reward signal, a value function, and sometimes a model of the processing environment. The policy defines the way that the learning agent behavior at each time step, or in other words is a mapping function that takes space and provides the corresponding action. A reward signal defines the aim of the RL algorithm. The agent's goal is to maximize the total reward over some time. Therefore, the reward signal defines the good events – rewards themselves, or the bad events – penalties to avoid those events. The value function specifies what is good in the long run. Which is defined as a total reward that an agent can expect to accumulate over some finite period. Finally, the methods for solving RL problems can be model-based, where the model of the environment is predefined, or model-free, where the environment is completely unknown and the algorithm is purely based on trial-and-error learners.

The RL uses the formal framework of Markov decision processes (MDP), to define the interaction between the learning agent and its environment in terms of states, actions, and received reward signals. The simple one-step agent-environment interaction in MDP starts from the agent receiving the current state S_t, which can be predefined by a set of the possible states $S_t \in \mathbf{S}$ and a recently obtained reward R_t at time step t. The agent`s function is based on the current state S_t define the action A_t, typically chosen from the predefined set of possible actions $A_t \in \mathbf{A}(s)$. The action then interacts with the environment and the outcome is given by a new state S_{t+1} and a produced reward signal R_{t+1}. The agent-environment interactions can be seen in Figure 5.6-5.

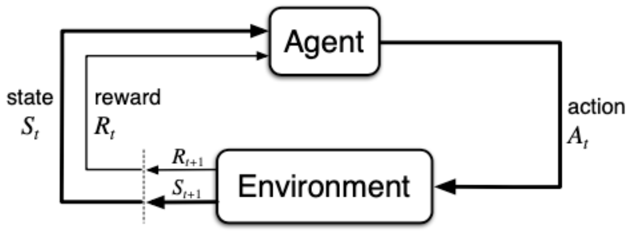


Figure 5.6-5 The MDP paradigm of agent-environment interaction

The RL as a tool have been shown in applications for efficient DASH adaptation algorithms, where they effectively learn the constantly changing environments by executing the action, which can be represented as quality of consumed video [32]. Another work, that has been successfully applied for content delivery domain – is [35], where authors propose RL for efficient content caching. Additionally, the RL algorithms were successfully applied to provide additional security in mobile edge caching [59].

6 Conclusions

This document acts as a summary of the efforts that have been done towards Work Package 2 Task 2.1. It provided an introduction to MEC technology as a key enabler for 5G-and-beyond cellular standards. Then, background on more specific enabling techniques such as MEC clustering, 5G positioning, adaptive video streaming and blockchain was presented. State of the art review revealed past and future research directions towards WP2 objectives, whereas discussion of potential methodological tools shed light on how specific open research questions can be tackled. Finally, the document presented the methodological tools and platforms available through the beneficiaries.

7 References

[1] ETSI GS MEC 003,“Mobile Edge Computing (MEC); Framework and Reference Architecture”, 2019.

[2] ETSI GS MEC 002. “Multi-access Edge Computing (MEC); Phase 2: Use Cases and Requirements”, 2018.

[3] Mach, Pavel, and Zdenek Becvar. "Mobile edge computing: A survey on architecture and computation offloading." *IEEE Communications Surveys & Tutorials* 19.3 (2017): 1628-1656

[4] ETSI GR MEC 017., “Mobile Edge Computing; Deployment of Mobile Edge Computing in an NFV Environment, February 2018.

[5] ETSI GS NFV-MAN 001., “Network Functions Virtualisation; Management and Orchestration”, December 2014.

[6] ETSI GS NFV-IFA 014., “Network Functions Virtualisation; Management and Orchestration; Network Service Templates Specification”, October 2016.

[7] ETSI GS NFV-IFA 011., “Network Functions Virtualisation; Management and Orchestration; VNF Packaging Specification”, October 2016.

[8] ETSI GS MEC 010-2., “Multi-access Edge Computing (MEC); MEC Management; Part 2: Application Lifecycle, rules and requirements management”, November 2019.

[9] Quoc-Viet Pham, Fang Fang, Vu Nguyen Ha, Mai Le, Zhiguo Ding, Long Bao Le, Won-Joo Hwang, “A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art,” *PREPARED FOR IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*, [online] Available: arxiv.org/pdf/1906.08452v1.pdf

[10] ETSI White paper No.28., “MEC in 5G networks,” ISBN No. 979-10-92620-22-1, First edition June 2018.

[11] O-RAN Alliance White paper, “O-RAN use cases and deployment scenarios, Towards open and smart RAN,”, February 2020

[12]A. Tootonchian, A. Panda, C. Lan, M. Walls, K. Argyraki, S. Ratnasamy, S.Shenker, “ResQ: Enabling SLOs in network function virtualization”, USENIX NSDI, Washington USA, 2018, pp. 283-297

[13] J. Khalid, E.Rozner, W.Felter, C.Xu, K. Rajamani, A. Ferreira, A.Akella “Iron: Isolating Network-based CPU in Container Environment”, USENIX NSDI, Washington USA, 2018, pp. 313-328

[14] Y. Wang, V. Friderikos, “A Survey of Deep Learning for Data Caching inn Edge Network”, Special Issue The Future of Network Softwarization: A Network Function Virtualization Approach, 7(4), 43, Informatics, 2020.

[15] Z. Li et al., "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," in *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719-733, April 2014, doi: 10.1109/JSAC.2014.140405.

[16] MoveNetworks. (Jun. 2013). [Online]. Available: http://www.movenetworks.com/history.html

[17] Y. Sani, A. Mauthe and C. Edwards, "Adaptive Bitrate Selection: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2985-3014, Fourthquarter 2017, doi: 10.1109/COMST.2017.2725241.

[18] C. Zhou, X. Zhang, L. Huo and Z. Guo, "A control-theoretic approach to rate adaptation for dynamic HTTP streaming," 2012 *Visual Communications and Image Processing*, 2012, pp. 1-6, doi: 10.1109/VCIP.2012.6410740.

[19] A. Masood, T. -V. Nguyen and S. Cho, "Deep Regression Model for Videos Popularity Prediction in Mobile Edge Caching Networks," *2021 International Conference on Information Networking (ICOIN)*, 2021, pp. 291-294, doi: 10.1109/ICOIN50884.2021.9333920.

[20]Dinh C Nguyen, Pubudu N Pathirana, Ming Ding, and Aruna Seneviratne. “Blockchain for 5g and beyond networks: A state of the art survey,” *Journal of Network and Computer Applications*, page 102693, 2020

[21]Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. “Solutions to scalability of blockchain: A survey.” IEEE Access, 8:16440–16455, 2020

[22]S. Nakamoto et al., “*Bitcoin: A peer-to-peer electronic cash system*,” 2008.

[23]W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "*A survey on consensus mechanisms and mining strategy management in blockchain networks*," IEEE Access, vol. 7, pp. 22 328-22 370, 2019.

[24]G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” Ethereum project yellow paper, vol. 151, pp. 1–32, 2014.

[25]C.Cachin, “Architecture of the hyperledger blockchain fabric,” in Workshop on distributed cryptocurrencies and consensus ledgers, vol. 310, 2016.

[26]D. Xenakis, A. Tsiota, C. Koulis, C. Xenakis and N. Passas, "Contract-less Mobile Data Access Beyond 5G: Fully-decentralized, high-throughput and anonymous asset trading over the Blockchain," *IEEE Access*, doi: 10.1109/ACCESS.2021.3079625.

[27]A. Manousis, R.A Sharma, V.Sekar and J.Sherry, “Contention-Aware Performance Prediction For Virtualized Network Functions”, ACM Sigcomm, Newyork City, USA, 2020, pp. 270-282, doi: https://doi.org/10.1145/3387514.3405868

[28]Dongyu Wang*, Xinqiao Tian, Haoran Cui, Zhaolin Liu, “Reinforcement Learning-Based Joint Task Offloading and Migration Schemes Optimization in Mobility - Aware MEC Network,” China Communications, August 2020

[29]Nessrine Hammami, Kim Khoa Nguyen and Mohamed Cheriet, “Joint Radio and Computing Resource Allocation in 5G Open Radio Access Network,” 2019

[30]Umer Majeed, Sheikh Salman Hassan, Choong Seon Hong, “Cross-Silo Model-Based Secure Federated Transfer Learning for Flow-Based Traffic Classification*,” International Conference on Information Networking (ICOIN)* 2021, IEEE 978-1-7281-9101-0/21/$31.00 ©2021

[31]Sridharan Natarajan, Tarun Khandelwal, Mohit Mittal, *“*MEC Enabled Cell Selection for Micro-operators based 5G Open Network Deployment*,”* IEEE 978-1-7281-5178-6/20/$31.00 ©2021

[32] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, «D-DASH: A Deep Q-Learning Framework for DASH Video Streaming,» *IEEE Transactions on Cognitive Communications and Networking*, v. 3, № 4, pp. 703-718, 2017, doi: 10.1109/TCCN.2017.2755007.

[33] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “CS2P: Improving video bitrate selection andadaptation with data-driven throughput prediction,” Proceedings of the 2016 ACM SIGCOMM Conference, Florianopolis, Brazil, page 272–285, 2016, doi: 10.1145/2934872.2934898.

[34] H. Kim and K. Chung, "Multipath-Based HTTP Adaptive Streaming Scheme for the 5G Network," in *IEEE Access*, vol. 8, pp. 208809-208825, 2020,

[35] V. Kirilin, A. Sundarrajan, S. Gorinsky and R. K. Sitaraman, «RL-Cache: Learning-Based Cache Admission for Content Delivery,» *IEEE Journal on Selected Areas in Communications*, v. 38, № 10, pp. 2372-2385, 2020, doi: 10.1109/JSAC.2020.3000415.

[36] M. Chen, W. Saad, C. Yin, and M. Debbah, «Echo State Networks for Proactive Caching in Cloud-Based Radio Access Networks With Mobile Users,» *IEEE Transactions on Wireless Communications*, v. 16, № 6, pp. 3520-3535, 2017, doi: 10.1109/TWC.2017.2683482.

[37] L. Ale, N. Zhang, H. Wu, D. Chen and T. Han, «Online Proactive Caching in Mobile Edge Computing Using Bidirectional Deep Recurrent Neural Network,» *IEEE Internet of Things Journal*, v. 6, № 3, pp. 5520-5530, 2019, doi: 10.1109/JIOT.2019.2903245.

[38] S. Chikkerur, V. Sundaram, M. Reisslein and L. J. Karam, "Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison," in *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 165-182, June 2011, doi: 10.1109/TBC.2011.2104671.

[39] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, 2015.

[40] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-Scale Control Plane for Video Quality Optimization. In *Proc. USENIX NSDI*, 2015.

[41]H. S. Dhillon, R. K. Ganti, F. Baccelli and J. G. Andrews, "Modeling and Analysis of K-Tier Downlink Heterogeneous Cellular Networks," in *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 3, pp. 550-560, April 2012, doi: 10.1109/JSAC.2012.120405.

[42]B. Błaszczyszyn, M. K. Karray and H. P. Keeler, "Using Poisson processes to model lattice cellular networks," 2013 Proceedings IEEE INFOCOM, 2013, pp. 773-781, doi: 10.1109/INFCOM.2013.6566864.

[43]X. Lin, R. K. Ganti, P. J. Fleming and J. G. Andrews, "Towards Understanding the Fundamentals of Mobility in Cellular Networks," in *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1686-1698, April 2013, doi: 10.1109/TWC.2013.022113.120506.

[44]Y. Chen, B. Zhang, Y. Liu and W. Zhu, "Measurement and Modeling of Video Watching Time in a Large-Scale Internet Video-on-Demand System," in *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 2087-2098, Dec. 2013, doi: 10.1109/TMM.2013.2280123.

[45]X. Ling, J. Wang, T. Bouchoucha, B. C. Levy and Z. Ding, "Blockchain Radio Access Network (B-RAN): Towards Decentralized Secure Radio Access Paradigm," in *IEEE Access*, vol. 7, pp. 9714-9723, 2019, doi: 10.1109/ACCESS.2018.2890557.

[46]J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments", Jan 2016. [Online] https://lightning.network/lightning- network-paper.pdf.

[47]Martin Haenggi. *Stochastic geometry for wireless networks*. Cambridge University Press, 2012.

[48]J. Andrews, Jeffrey & A. Gupta and D. Harpreet. “A Primer on Cellular Network Analysis Using Stochastic Geometry”. 2016.

[49]H. ElSawy, E. Hossain and M. Haenggi, "Stochastic Geometry for Modeling, Analysis, and Design of Multi-Tier and Cognitive Cellular Wireless Networks: A Survey," in *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 996-1019, Third Quarter 2013, doi: 10.1109/SURV.2013.052213.00000.

[50]M. Afshang and H. S. Dhillon, "Poisson Cluster Process Based Analysis of HetNets With Correlated User and Base Station Locations," in *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2417-2431, April 2018, doi: 10.1109/TWC.2018.2794983.

[51]Willig, A., 1999. *A short introduction to queueing theory*. Technical University Berlin, Telecommunication Networks Group, 1999.

[52]Kawase, Y. and Kasahara, S. “Priority queueing analysis of transaction-confirmation time for Bitcoin” in *Journal of Industrial & Management Optimization*, vol 16(3), 2020

[53] D. Brunner, B. Penkovsky, I. Fischer., Tutorial: Photonic neural networks in delay systems., 2018.

[54] E. Charniak, Introduction to Deep Learning (The MIT Press), The MIT Press; Illustrated edition., January 29, 2019.

[55] D. P. Mandic, J. A. Chambers., Recurrent Neural Networks For Pre-diction., 2001.

[56] C. C. Aggarwal, Neural Networks and Deep Learning, Springer International Publishing, 2018.

[57] Tsironi, Eleni & Barros, Pablo & Weber, Cornelius & Wermter, Stefan. , "An Analysis of Convolutional Long-Short Term Memory Recurrent Neural Networks for Gesture Recognition," Neurocomputing. , vol. 268, 2017.

[58] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. Second edition, A Bradford Book. The MIT Press. 2018

[59] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, ‘‘Security in mobile edge caching with reinforcement learning,’’ IEEE Wireless Commun., vol. 25, no. 3, pp. 116–122, Jun. 2018.

[60] T. M. Ayenew, D. Xenakis, N. Passas, and L. Merakos, «Cooperative Content Caching in MEC-enabled Heterogeneous Cellular Networks», IEEE Access, accepted, 2021.